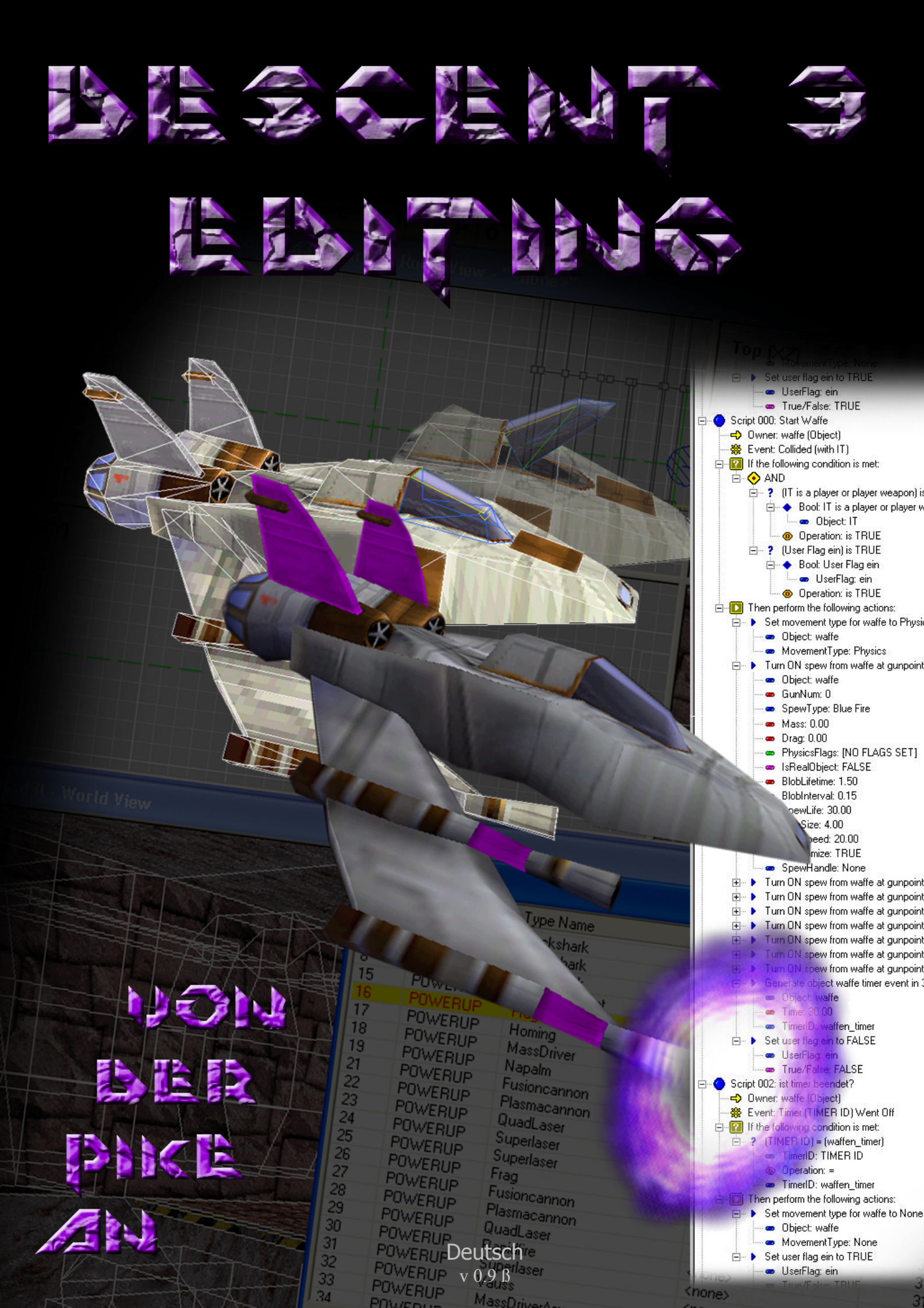
DESCENT 3 EDITING



VON DER PIKE AN

To the
Community.

INHALT

- **Ein Kurzes Wort vorab...**
- **Abschnitt Reisberg – Das Programm**
- **Abschnitt A – Einen Level von Grund auf erstellen**

001 - Erklärung des Editors 20
002 - Der erste Raum....23
003 - Den Raum ändern....27
004 - Texturing....29
005 - Der erste Level....31
006 - Einen zweiten Raum hinzufügen....34
007 - Reaktor-Raum....37
008 - Reaktor....40
009 - Lathe einen Raum....43
010 - Das Bend Tool....45
011 - Basics....48
012 - Ein einfacher Raum....51
013 - Ein Level anfangen....54
014 - Texturierung....56
015 - Beleuchtung....58
016 - Den Level testen....62
017 - Neue Räume anfügen....64
018 - Objekte &Player Starts....66

- **Abschnitt B – Vertiefung in D3Edit**

019 - Shortcuts einsehen 69
020 - Das Geometrie-Paneel 70
021 - Quicktest mit D3Edit einrichten 84
022 - Editoreinstellungen und -profile sichern....87
023 - Standard - Mod's 90
024 - Tools Menü anwenden 93
025 - Objekt Info Dialog (OI)....94

- **Abschnitt C – Fortgeschrittenes Bauen**

026 - Bau einer hochwertigen Shell 103
027 - Multiple Layers 106
028 - Multiple Extrusion....110
029 - Multiplanare Extrusionen kombinieren 114
030 - Raum Teilen....118
031 - Vertexoperationen an Faces....121
032 - Segmente anwählen....122
033 - Benden – Grundlagen 124
034 - New Style Bend, v39....126
035 - Die Bend-Funktion 129
036 - Runderneuerte Bend-Funktion in v40....131
037 - Bend: Mathematik....133
038 - Benden, Objekte inklusive....135
039 - Biegung ohne Bend....137
040 - Lathe – Vertiefung 138
041 - Extrude – Vertiefung 141
042 - Extrude: Zero....147
043 - Spieglein, Spieglein....154
044 - Raum im Raum 157
045 - Eine Sphäre machen....159
046 - Komplexität ganz einfach....167
047 - Primitives....178
048 - Felsformationen erzeugen....184
049 - Tunnel im Höhlen-Stil 186

- **Abschnitt D – Debug**

050 - Konkave Faces 194

051 - Non-Planar Faces....195
052 - T-Joint....197
053 - T-Joints: Zweite Version....198
054 - Verifiziere Dein Level....202
055 - Verifiziere Deine Mine....207

❖ Abschnitt E – Terrain

056 - Terrain-Implementierung....214
057 - Außenwelt erstellen....217
058 - Außenwelt – Update....223
059 - Spielerstart in der Außenwelt....225

❖ Abschnitt F – Texturen

060 - Eigene Texturen....230
061 - Custom Texturen Anwenden....232
062 - Custom Texturen: Tipps....236
063 - Texturen kachelbar machen....239
064 - Teiltransparente Texturen....242
065 - Animierte Texturen - Screenshots....247
066 - Animierte Texturen - Selbstgemacht....254
067 - Texturen @ 256x256....255

❖ Abschnitt G – Sounds

068 - Sounds abspielen....257
069 - Sound im eigenen Descent3-Level....260
070 - Sound: Probleme vermeiden....264

❖ Vorgriff: DALLAS zum Laufen kriegen

❖ Abschnitt H – Türen & Objekte

071 - Doors Tutorial....273
072 - Eigene Doors....276
073 - Sound für die eigene Tür....282
074 - Eigene Objekte 1....283
075 - Eigene Objekte 2....285
076 - Eigene Objekte 3....289
077 - Custom Robots Part 1....293
078 - Einführung & Übersicht....296
079 - Die Powerup GAM....298
080 - Die Weapon GAM....299
081 - Eine simple Primärwaffe....303
082 - Die Schiffs-GAM....307
· OOFEdit-Exkurs....309
083 - OOFEdit: Terminologie....311
084 - OOFEdit: Features....312
085 - OOFEdit: Erste Schritte....313
086 - OOFEdit: Ein Powerup....316
087 - OOFEdit: Turrets....318
088 - OOFEdit: Animation....321
089 - Ship Tutorial....325

❖ Abschnitt I – Scripting

090 - Scripting: Hello World....330
091 - Scripting: Einführung....333
092 - Neue DALLAS Suchfunktionen....335
093 - Teleporter....343
094 - Create Lighting....344
095 - Wind skripten....346
096 - Musikregionen....347
097 - Spew!....350
098 - Glasbruch mit Scherben....352
099 - Einen Schalter skripten I....356
100 - Einen Schalter skripten II....359
101 - Scripten einer Inventory-Cloak....362
102 - Reactor Gamma: Script Companion....366

❖ Abschnitt J – Designing

103 - Level Design Rules....369
104 - Single Player Level Building....372
105 - Single Player....374
106 - Anarchy....376
107 - Anarchy....378
108 - Team Games....380
109 - When Great Levels Die....382
110 - Eye Candy....389
111 - Waffenbalance....392
112 - Warum Missionen bauen?....394

❖ Abschnitt K – Single Player

113 - Level-Ziele....397
114 - Roboterzentren....398
115 - Pfade....400
116 - BNode-Tutorial....404
117 - BNodes setzen....406
118 - Player-Respawnpunkte - toprog....410
119 - Loadscreen einbinden....413
120 - Levelbriefing....415

❖ Abschnitt L – Meisterklasse

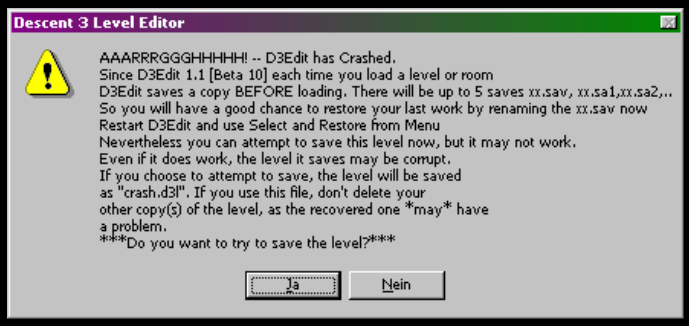
121 - Türen optimieren....421
122 - Dynamisches Licht....425
123 - Gruppendateien....428
124 - Customs, nochmal....432
125 - Bauen mit Objekten....433
126 - Mercenary-Bots verwenden....438
127 - Schiffswahl in Singleplayer - toprog....441
128 - Scripting & Matcen - toprog....445
129 - Räume teilweise skalieren....450
130 - Zusammengesetzte Objekte inprog....452

❖ Anhang – Nützliche Infos

Liste aller D3 Tools....456
Haus der Spiegel....461
Ingame-Hilfen fürs Testen....462
D3-Edit Limits....464
Anwendung von D3 Levelinfo....465
Dokumentation der Türen in Descent 3....468
Custom-Objekte in D3Edit sichtbar haben....493
Das Gam-Format: The Specification File....494
Texturenkatalog - das überleg' ich mir aber noch....516
Shortcut-Liste....518

Ein Kurzes Wort vorab...

Der Zweck dieses Werkes ist es, ein möglichst vollständiges Bild zu vermitteln wie man für Descent 3 Levels baut, welche nicht nur gut und fehlerfrei aufgebaut sind, sondern auch ansprechend gestaltet.



Diesen Dialog werdet Ihr noch öfter sehen.... es ist wohl eine gute Idee, sich anzugewöhnen öfters mal zwischenzuspeichern.
Angenehmer Nebeneffekt, man macht das dann bei allen Programmen, ist ja auch kein Fehler ☺

Ich habe diese Datei anfangs aus den Tutorials zusammenkompiliert, welche sich auf www.descent3fischlein.de befinden, da ich es für mich als praktischer erachte eine einzelne File zu haben, die ich durcharbeiten kann. Ich habe alle überarbeitet sowie durchnummeriert, fehlende Infos hinzugefügt, auch Grafiken, so der Erklärung dienlich. Die Dateien welche zur jeweiligen Section gehören sind ebenfalls entsprechend nummeriert. Nachdem im Netz doch noch ein paar andere Tutorials existieren, hab ich beschlossen einfach alle verfügbaren hier einzugliedern und zu übersetzen falls nötig, sodaß ein umfassendes Werk entsteht in dem ich jederzeit nachspicken kann.
Dadurch ergibt es sich allerdings, daß das Eine oder Andere Thema mehrmals behandelt wird.
Weiter hinten stehende Themen setzen das Wissen der vorherigen üblicherweise voraus.
Bei Übersetzungen habe ich mich bemüht möglichst am Original zu bleiben, wenn möglich auch den Witz mitzunehmen, wo aber ganz einfach Begriffe vorkommen die nicht 1:1 übersetzbar sind, habe ich versucht den Sinn des Textes zu erhalten.

Die Schöpfungsrechte verbleiben natürlich bei den jeweiligen Autoren!

Es gibt in diesem Dokument Konventionen, welche die einfachere Verwendung unterstützen sollen:

`Dateiname.ext`	Wörter in dieser Schriftart verweisen auf Dateien bzw. Dateinamen und deren Pfade.
Strg+Alt+Entf	Tastenkombinationen; Groß- & Kleinbuchstaben sind gleichbedeutend, **Shift** wird extra angegeben. Bei Ziffern ohne zusätzliche Angaben: immer die vom Nummernblock!
To Current	Buttons die zu klicken sind oder auch Titel von Dialogfeldern.
File->New...	Diese Schriftart weist auf eine Programm-Menüauswahl hin, oder etwas das vom Editor angezeigt wird; etwa in der Statuszeile oder Dialogfeldern.
`ls -AGhl`	Dies zeigt Ein- oder Ausgaben an, auch Code oder Listen.
Es ist wies ist	Anmerkungen von mir werden in 8 Punkt geschrieben.
Tx	Damit sind die Buttons in der Toolbar gemeint.
Achtung:	Infos in rot umrandeten Boxen solltet Ihr ernst nehmen - wirklich!

Die verwendeten Editoren sind 1.1 [Beta 10 | Atan P.0_1_39(p)] - kurz v39 genannt; und die 1.1 AV 40, kurz v40 genannt, welche viele Neuerungen, Verbesserungen und mehr Bequemlichkeit bringt.

Also laßt jucken Leutz,

RAGN RAL
Member of
DM

Abschnitt Reisberg – Das Programm

Ragil Ral

Ihr kennt das Märchen vom Schlaraffenland? Gut!
Denn wie bei allen tollen Sachen muß auch hier ein Hindernis überwunden werden, bevor das eigentliche Erlernen, Probieren und Bauen stattfinden kann. Wer diese Sektion überspringt kanns zwar trotzdem schaffen, aber es wird signifikant länger dauern - es sei denn man hat anderweitig erworbene Kenntnisse, die hier greifen.

Um was gehts überhaut?

Wer dies liest, der will für Descent 3 Missionen erstellen und lernen wie das jetzt geht, verdammtnocheins. Natürlich möglichst schnell 😁 das ist ja klar.

Voraussetzungen

Man braucht eigentlich nur grundlegende Computer-Kenntnisse sowie ein wenig Englisch (oder die Wörterbücher am Schreibtisch), der Rest steht hier drin und begann mit dieser Zeile. Des weiteren ist ein wenig Sitzfleisch gefragt, da es sich mit einmaligem Durchkauen eines Themas nicht bewendet, wenn man etwas wirklich lernen will; wer ein Musikinstrument spielt wird wissen was ich meine... Übung macht den Meister, auch hier. Probiert herum. Und um des Himmels Willen, übt übt übt! Das heißt aber nicht daß Ihr immer dasselbe machen sollt, sondern daß Ihr das Gelernte einfach ausgiebig ausprobiert. Als sehr praktisches Untensil erwies sich ein Taschenrechner. Natürlich braucht Ihr auch den D3Edit.

Grundelemente

In Descent 3 ist alles aus Polygonen, Faces genannt, aufgebaut die 'Edges' als Kanten haben und 'Vertices' als Eckpunkte. Auch Objekte wie unsre Pyros bestehen aus ihnen (sind eigentlich Raum-Dateien, denen man 'Leben' eingehaucht hat). Daher ist der Editor auch auf diese spezialisiert, und alles was man mit dem Editor macht (Von Objekten wie Bots, Powerups oder Scripting mal abgesehn) ist, diese Grundelemente derart zu manipulieren daß am Ende ein möglichst geiler Level rauskommt. Die einzelnen Elemente erhalten eine Durchnummerierung welche mit 0 beginnt.

Besorgt Euch immer den aktuellsten Editor! Es gibt ihn im Descentforum, unter „D3Edit - Development - Forum / D3-Editing Tools“:

http://www.descentforum.de/forum/viewtopic.php?t=2577&sid=a48c24638cf76c6af4640f26e72988e2

Ihr bekommt dort auch andere Tools, die Ihr später noch brauchen werdet.

Current & Marked

‚current’ ist ein wenig schwer zu übersetzen: ‚aktuell’, ‚derzeit gültig’, ‚gegenwärtig’. Damit bezeichnet man ein ausgewähltes (angeklicktes) Element, und bevor ein Element bearbeitet werden kann muß es zumindest current sein. Current unterscheidet sich von markiert insofern, als daß bei Mausklick in eine aktive Ansicht jenes Element current wird, das unter dem Mauscursor liegt, entsprechend dem Betriebsmodus. Markierte Elemente bleiben hingegen markiert, auch bei Wechsel des Arbeitsmodus.

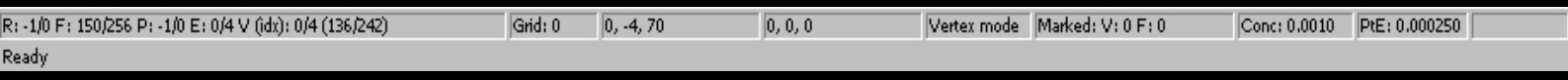
Es sei noch darauf hingewiesen, daß das Spiel (als Maschinenprogramm, das es ja ist) eigentlich nur mathematische Operationen mit dem macht, was Ihr als Level erstellt. Daher greifen auch mathematische Gesetzmäßigkeiten, und es ist von Vorteil diese zu kennen. Dadurch ist es leichter Fehler zu vermeiden, oder zu verstehen wie etwa ein Vertex funktioniert, oder warum es Fehler gibt.
Die einzelnen Buttons, Menüs und was da sonst noch alles so herumliegt lernt Ihr nachher in Folge kennen. Für derweil werden wir nur den groben Aufbau von D3Edit besprechen.

Allgemeine Tips zum D3Edit

Ich habe während des Machens dieser Datei gelernt, daß man besser nachdenkt BEVOR man IRGENWAS macht. Das Dialogfeld in der Einleitung habt ihr vielleicht gesehen, und ihr wollt es sicher nicht genau dann sehen, nachdem ihr eine Stunde lang kreativ wart, aber nix gespeichert habt. Bevor Ihr irgendeinen Button klickt, eine Menü-Auswahl ausführen wollt oder an den Grundementen des Levels ernsthaft herumfingert, speichert Euer Zeug. Es wird auch immer wieder darauf hingewiesen werden. Seit v40 legt der Editor Sicherheitskopien an, genannt `levelname.sav` und `levelname.sa1` bis `levelname.sa5`. Je höher die Nummer in der Endung, desto jünger. Sie werden beim Öffnen eines Levels erzeugt.

Die Statuszeile

Sie befindet sich im unteren Teil des Programmfensters und zeigt Euch allerlei Informationen, etwa in welchem Betriebsmodus ihr seid, wo Euer Cursor im Editor ist etc.,etc. Seit v40 kann man hier einen Wert für eine bestimmte Funktion setzen.



Was jetzt was bedeutet werdet Ihr sukzessive erfahren. Macht sie euch zum Freund!

Einheiten

Im Editor gibt es ein Einheitensystem. Wie groß so eine Unit (=Einheit) jetzt tatsächlich ist weiß ich auch noch nicht genau (bisherige Tests lassen mich glauben daß eine Einheit etwa ein Yard sein könnte), aber sie sind maßgebend (naja, sind ja Einheiten ☺) Steht man mit dem Pyro vorne direkt an einer Wand an, beträgt das Sichtfeld etwa acht Einheiten Breite. Das angezeigte Koordinatengitter (Grid) zeigt Euch wo Ihr Euch befindet, seine Gitterkonstante bestimmt die Werte, mit denen im Editor die Elemente manipuliert werden. die Gitterkonstante ändert Ihr entweder über die **PgUp**/**PgDwn**-Tasten oder über das Kontextmenü, immer für die jeweilige Orthogonalansicht. Zusätzlich könnt Ihr die Gitterkonstante auch direkt über **1** bis **7** (Zahlenreihe über den Buchstaben) auswählen.
Ihr könnt mit **G** das Koordinatengitter ganz ausschalten und dadurch Verts, Faces und alles andere um 0.1 Einheiten bewegen, was allerdings Schwierigkeiten bringen kann. Vermeidet es, Konstruktionen zu bearbeiten, welche kleiner als 1 Unit sind.

Verhalten beim Editieren

Seid nicht sparsam. Was zuviel ist kann entfernt werden, aber wenn mal wirklich irgendwo irgendwas fehlt, ist meist Geduld gefragt; das gilt allerdings nicht in allen Belangen, manchmal muß man auch aufpassen wo man was dazutut; darauf wird dann aber (meist ☺) hingewiesen. Speichert Euch Eure Arbeit auch öfter unter anderen Namen, sodaß Ihr immer wieder auf ältere Dateien, also ältere Versionen, zurückgreifen könnt. Hin und wieder kommt einem eine Idee, die aber mit dem ausgebauten Raum und/oder Level nicht (mehr) realisierbar ist, dann ist sowas echt praktisch. Wenn der Level dann ganz fertig ist, kann immernoch der Großeinsatz der **Del**ete-Taste stattfinden.

Modes

D3Edit kennt vier verschiedene Betriebs-Modi, in denen jeweils bestimmte Angelegenheiten geregelt werden, und während des Levelbauens muß man oft zwischen ihnen hin- und herwechseln. Dies geht über Tastenkombinationen oder die Buttons in der Toolbar.

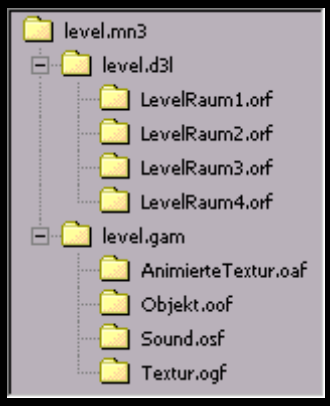
Hotkey	Button	Modus	Zweck
Ctrl-R		Vertex Mode	Erstellen und Bearbeiten von Vertices, Erzeugung von Faces
Ctrl-F		Face Mode	Erstellen von Faces, deren Bearbeitung und Texturierung
Ctrl-G		Object Mode	Für das Hantieren mit Objekten wie Powerups oder Bots
Ctrl-H		Path Mode	um 'intelligenten' Objekten bestimmte Vorgaben zu machen

Per **Tab** schaltet man zwischen dem Vertex Mode und dem Face Mode um, war man zuerst in einem der anderen Modes, bringt **Tab** einen in den Vertex Mode.

Tatsächlich überschneiden sich die Modes in ihren Möglichkeiten ein wenig, aber das paßt schon

Dateien

Ein Descent 3-Level besteht aus verschiedenen Ressourcen, nicht nur dem Level.
Zunächst mal die Dateiformate: Räume bekommen die Datei-Endung `.orf`, die Level bekommen `.d3l`; es gibt dann noch `.oof` für Objekte, `.ogf` und `.oaf` für Texturen und sogar `.dll`, beim Scripting; es gibt auch noch ein Format für Audio (`.osf`) und Einiges mehr. Das fertige, spielbare Level dann hat - wie Ihr wahrscheinlich wißt - `.mn3`; diese Datei ist ein Kompilat aus allen Ressourcen die zum Level gehören, und wird mittels eines Werkzeugs erstellt daß mit D3Edit mitkommt.



Hier links ist die Struktur einer Leveldatei grob nachgebildet.

> *Tip:*
> Beim Testen empfiehlt es sich, zuvor alle Missionen im D3-Missionsverzeichnis wegzuschieben. Einfach gleich im Missionsverzeichnis einen Ordner anlegen, und alle da reinschieben, sodaß dann nur Euer Level im Missionsverzeichnis liegt. Descent 3 lädt dadurch wesentlich schneller, und Ihr findet Euren Level auch gleich.

Im Editor erstellt Ihr Räume aus den genannten Grundelementen, und setzt diese dann zu Levels zusammen, eventuell mit Außenwelt; das wird dann die Leveldatei. Die einzelnen Räume können separat verspeichert werden, was praktisch ist - so erstellst Du Dir zum Beispiel eine Leuchte, gleich mit den richtigen Texturen, speicherst die, und kannst sie dann jederzeit überall einbauen wo Du Licht brauchst. Es gibt auch Raum-Dateien zum herunterladen, grabe ein wenig in den Foren. Und übrigens, auch die Türen sind eigene Räume, genauso wie alle Objekte!

Aber eben nicht nur die Leveldatei; man kann eigene Sounds, Texturen, Objekte ins Level bauen. Am Ende wird das alles in eine Datei reingeschaufelt, und das ist dann eben die `.mn3`, das spielbare Level. Man kann auch über DALLAS gewissermaßen 'Leben' in ein Level programmieren, sogar das Verhalten der Bots bestimmen; dann kommen noch eine `.dll` und eine `.msg` dazu. Die KI, welche ins Spiel reinprogrammiert wurde hat Schnittstellen, die über DALLAS angesprochen werden können. Wer mehr über die Spiele-KI wissen will, dem empfehle ich 'Developing an Artificial Intelligence Engine', von Michael van Lent und John Laird, entstanden am Artificial Intelligence Lab, University of Michigan; die Jungens haben an der Descent 3-KI mitprogrammiert. (`tools/coding/BotsInDescent3&QuakeII.zip`)

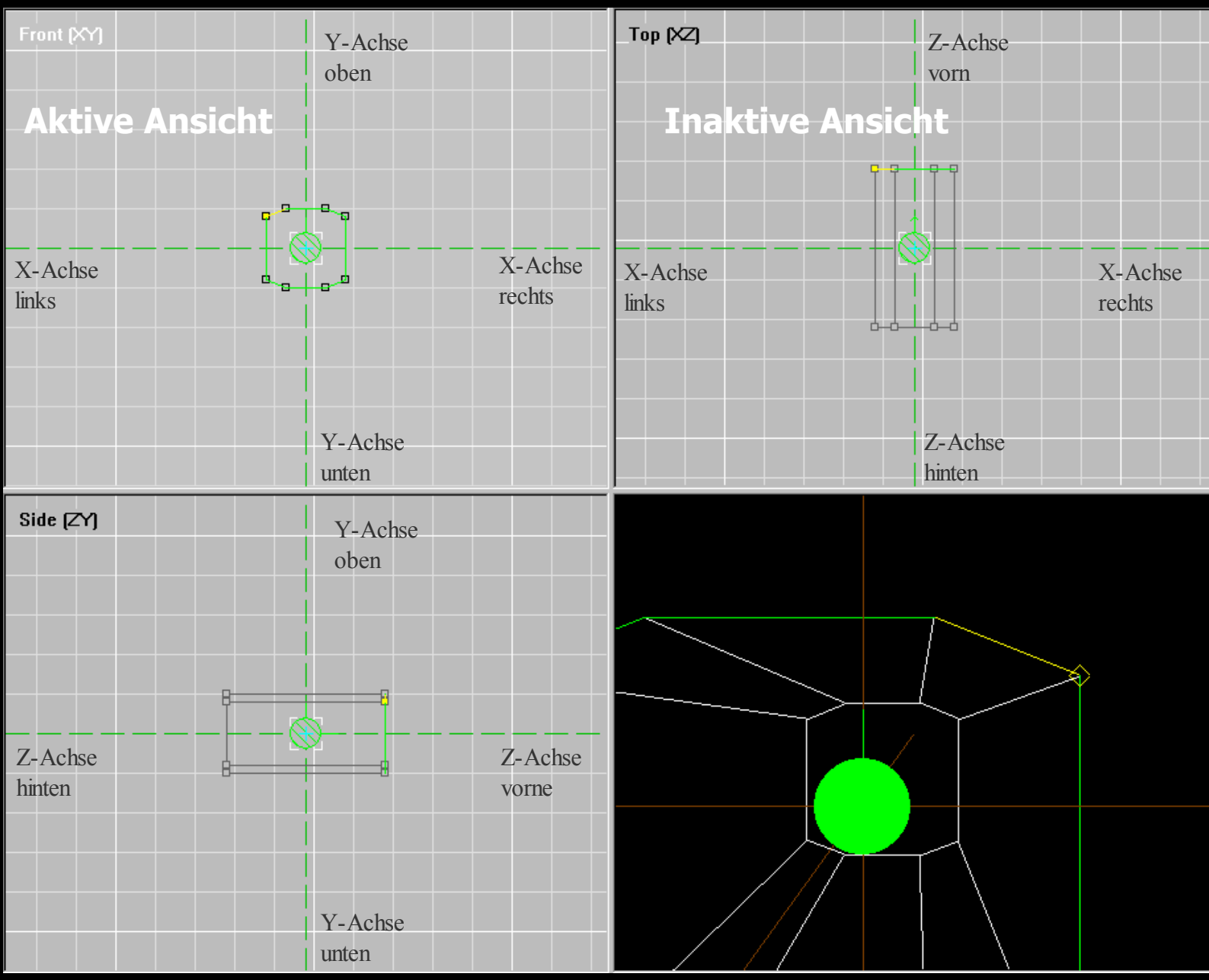
Bereit für den Sprung?

Geht auf File->New->New Level - Default Room. Damit erstellt der Editor ein (sogar spielbares!) gültiges Level; dieses ist ideal um Dinge auszuprobieren, da man ja nix verliert wenns mal schiefgeht.

Die Ansichten

Es gibt zwei- und dreidimensionale Ansichten, und die unterscheiden sich, aber nicht nur in dem was sie zeigen, sondern auch in ihrer Funktion. Die 2d-Ansichten heißen eigentlich korrekt 'Orthogonale' Ansichten ([lat. orthogonium < griech. orthogonion] (Geom.): Rechteck ☺) und sie zeigen immer nur zwei Achsen; dafür gibts aber dann gleich drei davon. Sie bieten nur die Linien-Ansicht. Die 3d-Ansichten haben hingegen drei Anzeigearten, Linien, Texturiert und Linien, sowie Texturiert.

2d-Ansichen



Um in einer Ansicht zu arbeiten, müßt ihr sie erst aktivieren, dazu einmal hineinklicken; sie wird dann etwas heller und die Bezeichnung wird weiß. Der nächste Klick hinein wird bereits was bewirken.
Hier macht man die Konstruktionen, fügt Elemente ein. Ihr seht das Gitter im Screenie unten, dies ist das Koordinatensystem des Editors. Es kann mit **G** ein-/ausgeschaltet werden, und bestimmt die Inkremente in denen Bewegungen oder Veränderungen ablaufen. Inkrement heißt hier, daß bei Bewegen oder Drehen von Elementen dies mit bestimmten fixen Werten abläuft (Tabelle). Man kann auch den Ursprungspunkt des Gitters ändern, mit dem Button links. Aber Vorsicht in den 2d-Ansichten, **Ctrl-Klick** in eine aktive Ansicht versetzt das Bezugszentrum ebenfalls! Ihr könnt es mit dem Button aus dem Konstruktionspaneel (Öffnen mit R)aber wieder zurücksetzen.Die Ansichten sind immer Vom Spieler-Startpunkt aus gesehen.

Inkrementeliste			
Gridsize Units	**lineares Inkrement Units**	**Winkelinkrement Grad**	**Winkelinkrement Quadrant-Anteile**
1	eingestellte Gridsize	1,4063	Quad/64
2		2,8125	Quad/32
5		5,625	Quad/16
10		11,25	Quad/8
20		22,5	Quad/4
50		45	Quad/2
100		90	Quad
Grid aus	0,1	entsprechend eingestellter Gridsize	

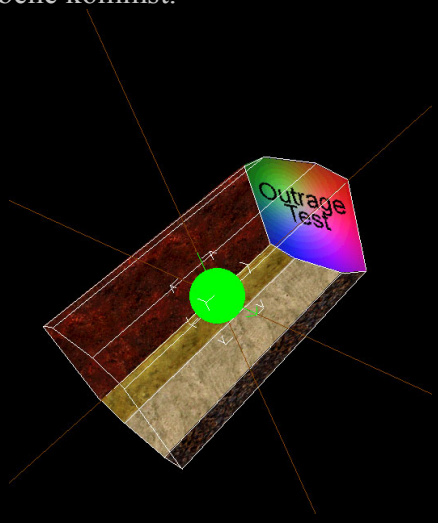
3d-Ansichten

Die 3d-Ansicht zeigt den Raum/das Level, und gibt einen Voreindruck wie es aussehen wird. Sie zeigt Dir eine perspektivische Ansicht Deiner Arbeit. Hier manipuliert man an dem Geschaffenen herum: man wählt Bereiche aus und texuriert sie, oder bearbeitet sie in einer der 2d-Ansichten weiter. Man kann aber auch hier in beschränktem Umfange Änderungen an der Geometrie vornehmen.

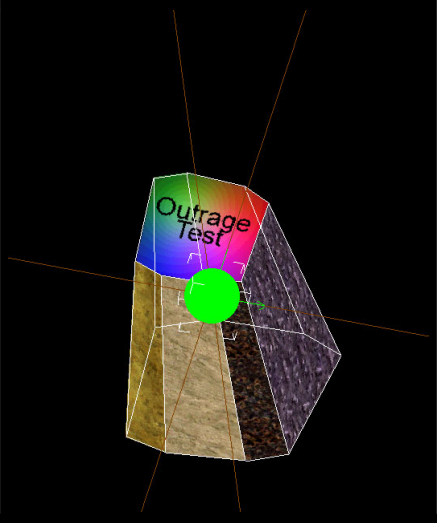
Bewegen in der 3d-Ansicht - Immer dorthin wo man will

Mit **Ctrl-Klick**en und halten/ziehen in einer 3d-Ansicht rotiert ihr diese. Wenn Ihr schon herumprobiert habt, werdet Ihr gemerkt haben daß es manchmal garnicht einfach ist das ins Blickfeld zu kriegen was man haben will.
Bewegt man die Maus horizontal, wird horizontal rotiert, bewegt man die Maus vertikal, wird entsprechend vertikal rotiert. Bezugspunkt ist hier immer die Anzeige!
Das braucht ein wenig Übung, ist aber wie Fahrradfahren.

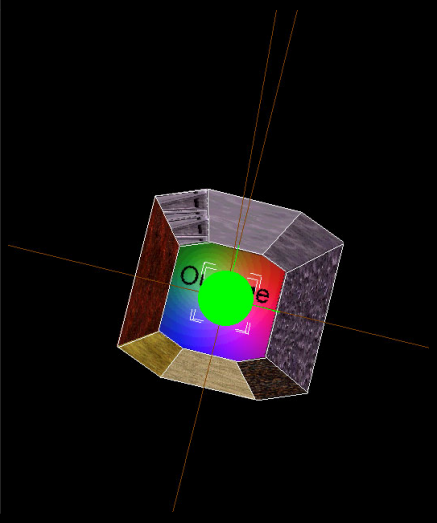
Alle Achsen schauen irgendwohin. Bewege die Maus horizontal und rotiere damit die Anzeige, bis Du in eine Ebene kommst.



Eine Ebene schaut nun zu Dir. Bewege die Maus nun senkrecht.



Nun schaust Du direkt auf eine Achse und siehst das Achskreuz, hier Y-Z .



Das ist nur ein einfacher Raum, doch mit zunehmender Komplexität muß man in den 3d-Ansichten immer genauer manövrieren.

Mit **Shift-Klick**en/halten und ziehen verschiebt Ihr die Ansicht.

Es gibt noch andere Navigationsmöglichkeiten, diese finden sich in den Kontextmenüs; lernt die Tastenkürzel dafür. Am praktischten finde ich **Strg-Shift-C** ☺

Die Ansichten ☺

Es gibt hier die World View, die Weltansicht und die Room View, die Raumansicht. In der Weltansicht seht Ihr den gesamten Level, mit den eingesetzten Objekten und je nach Einstellung unterschiedlich vielen Räumen (Einzustellen in File->Settings).

Die World View

hat nur die 3d-Ansicht. Man kann hier texturieren und Räume dazubauen, auch die Behandlung des Außenterrains findet hier statt. Habt ihr mehrere Räume im Level, so zeigt die Room View immer den aktuellen (=current) Raum.

Die Room View

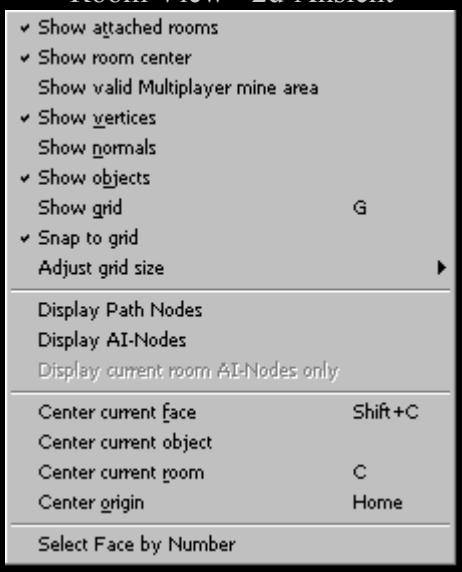
zeigt Euch entweder eben den Raum der in der Weltansicht current ist, oder einen separaten Raum, an dem Ihr arbeitet. Nur hier könnt ihr an der Geometrie umfangreiche Veränderungen vornehmen oder Objekte einsetzen.

Zur vollen Kontrolle aber braucht Ihr nun noch Kenntnis über die...

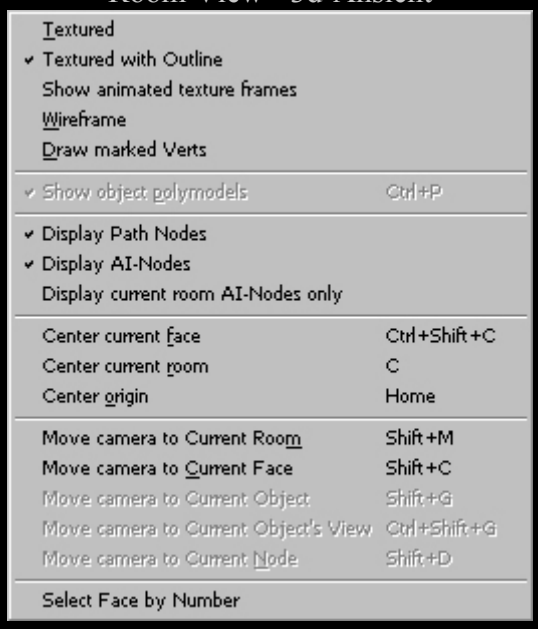
Kontextmenüs der Ansichten

Damit könnt Ihr Dinge anzeigen lassen oder verbergen, bei den 3d-Ansichten könnt Ihr die Anzeigeart bestimmen. Beachtet besonders auch Center ... und Move camera to..., das holt Euch Eure Sachen wieder auf den Schirm, falls Ihr mal totally lost sein solltet. Center zentriert die 3d-Ansicht auf das entsprechende Objekt, das heißt Ihr rotiert/pannt dann um dieses. Move camera holt es Euch nur ins Blickfeld. Zusätzlich gibt es auch noch **AltGr-C**, das resettet Euch Die Ansicht und zentriert auf den current Raum.

Room View - 2d-Ansicht



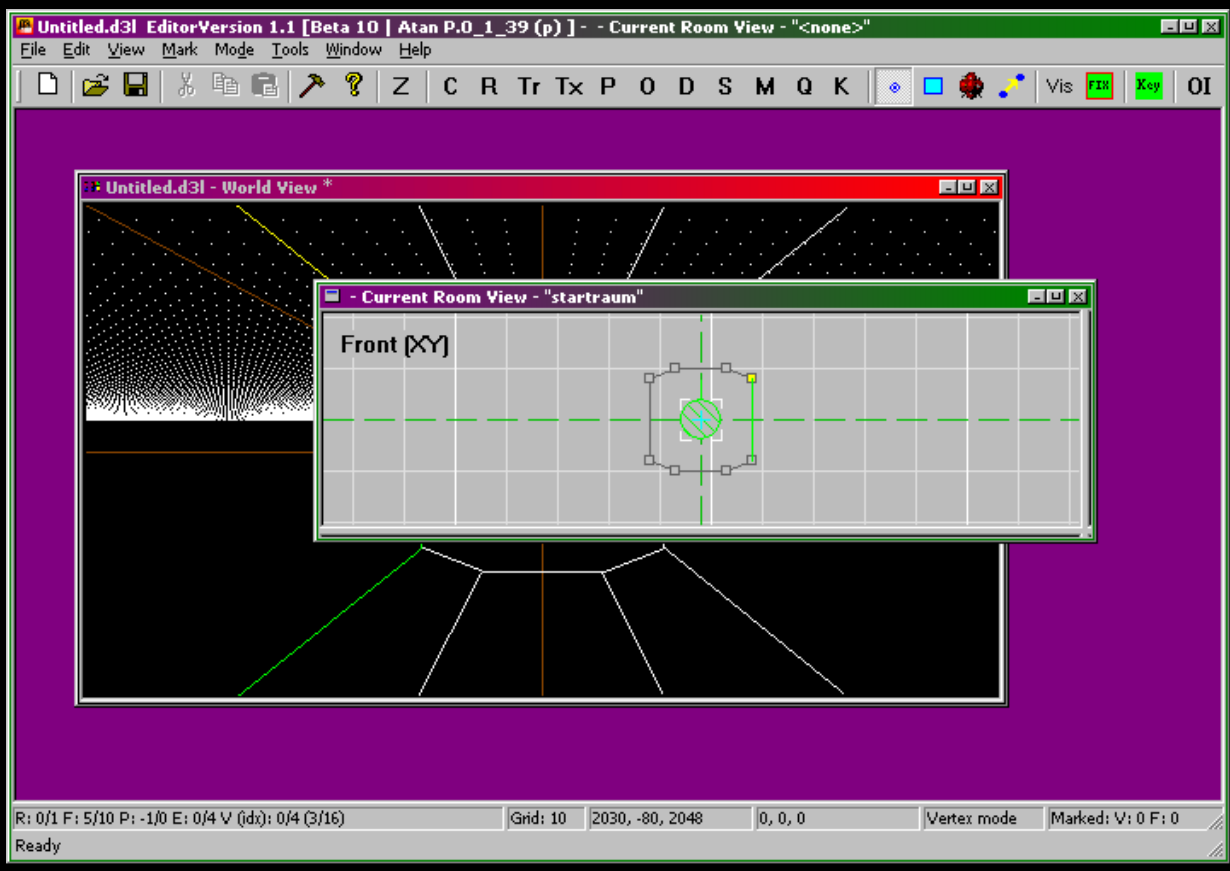
Room View - 3d-Ansicht

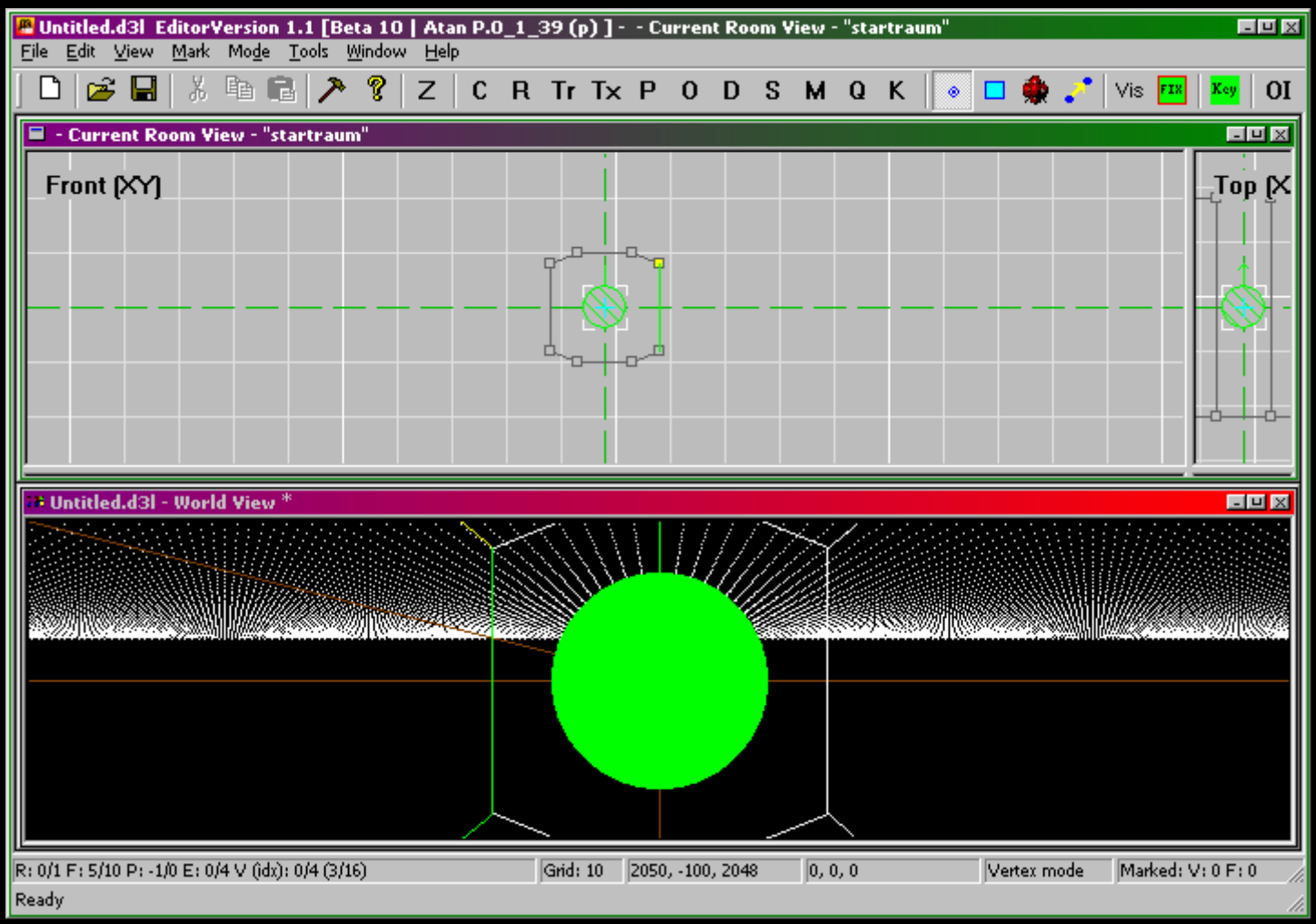


World View

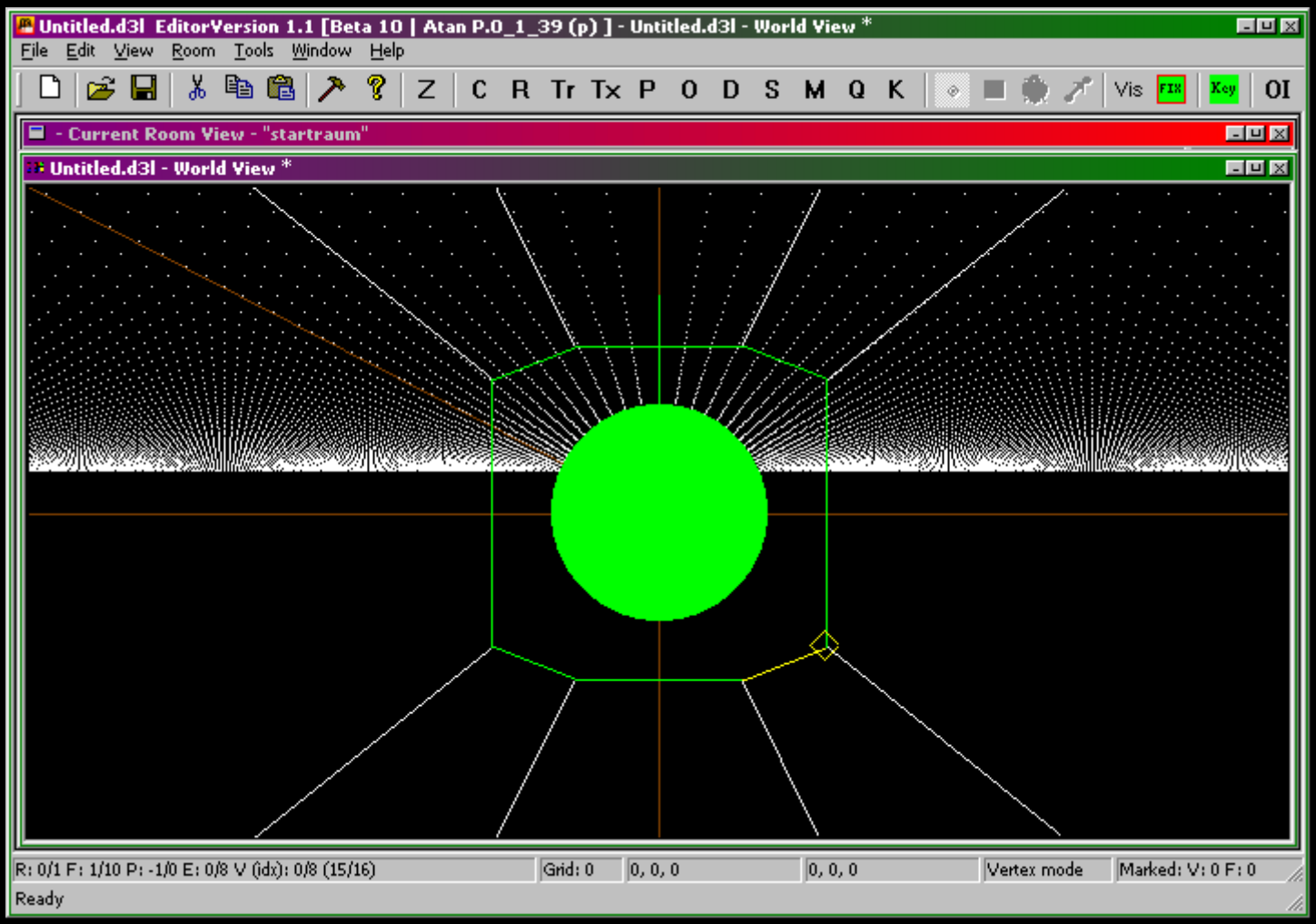
Noch ein Tip zu den Ansichten, denn Ihr werdet bald merken daß so ein Monitor recht wenig Platz bietet:
Unter Window gibt's die Auswahl Tile, was die offenen Dateifenster gleichmäßig im Fenster des Programmes anordnet.

Dabei wird das Aktive Fenster (hier mit der grünen Titelleiste) als oberstes angeordnet.

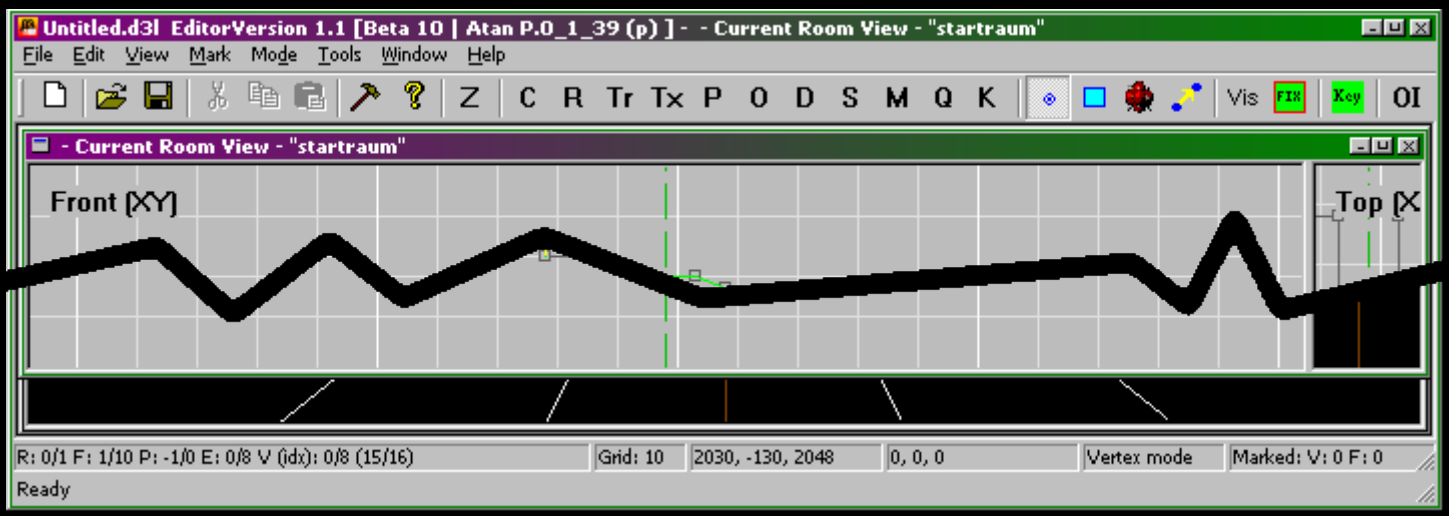




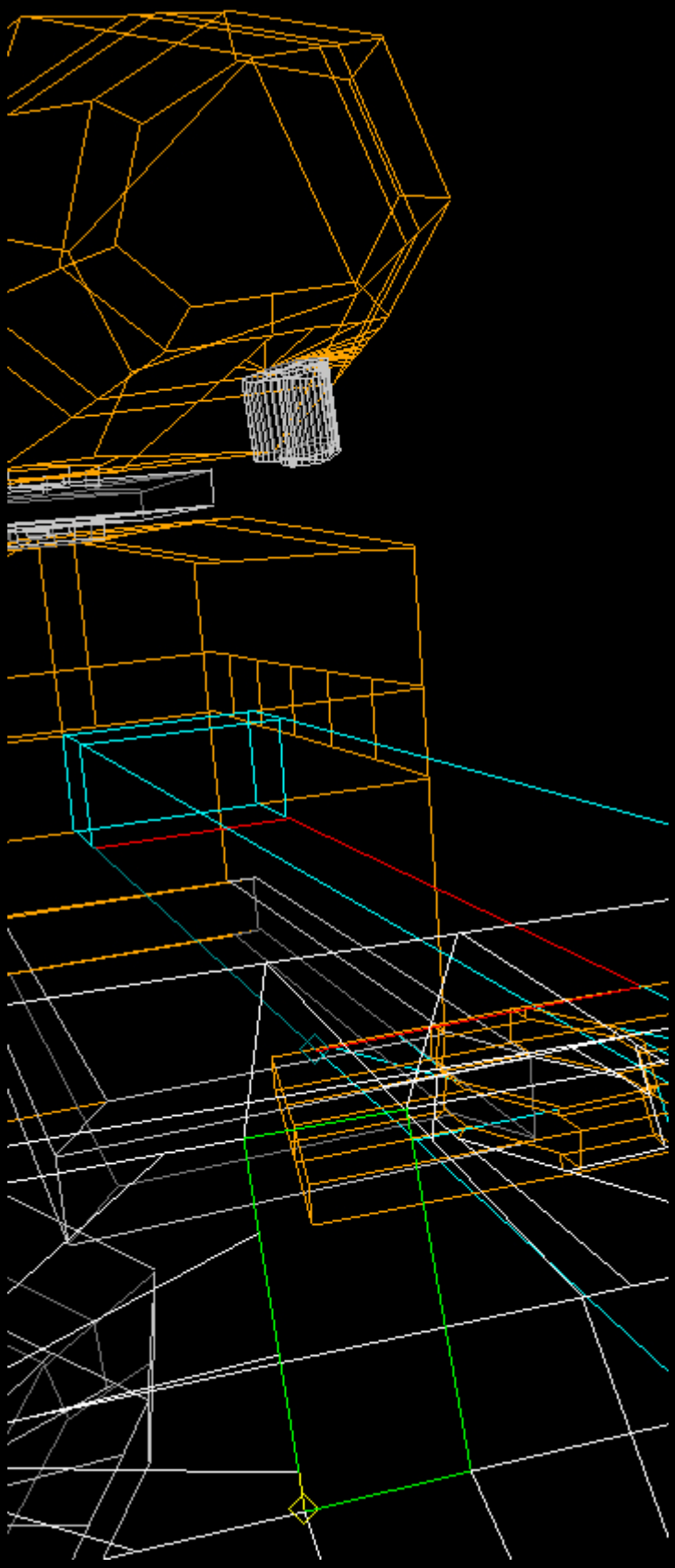
Dadurch kann man die Fenster so aufziehen, daß die Titelleiste des Fensters mit dem Raum zuoberst ist, und die der Weltansicht darunter.



Das hat den Vorteil, daß man immer sehen kann in welchem Raum man sich befindet bzw. welchen Namen er hat, da bei Wechsel in einen anderen Raum - in der Weltansicht - die Raumansicht stets den aktuellen (current) Raum und dessen Namen anzeigt. Zusätzlich kann man dann die Raumansicht dann etwas kürzer ziehen, sodaß man mit einem Klick wieder in der World View ist. Hat man mehr als zwei Fenster offen, muß man sich mit **Strg-Tab** behelfen, damit schaltet man durch die Fenster.



World View - Farbcode der Wireframe-Ansicht



Bewegt man sich im Wireframe Mode in der Weltansicht, so haben die Linien der Leveldarstellung unterschiedliche Farben. Diese bedeuten:

Orange
Dies kennzeichnet selektierte Räume.

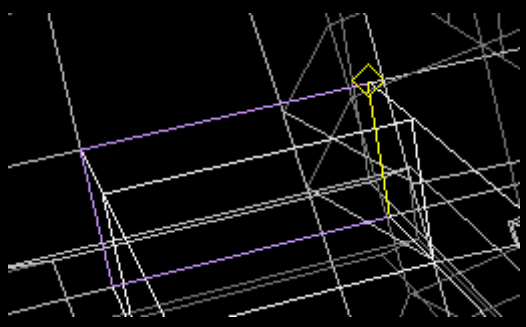
Nicht-ganz-so-weiß
die Standarddarstellung.

Türkis
Markierter Raum.

Rot
Markiertes Face des markierten Raumes

Weiß
Current Room; dieser wird dann in der Room View angezeigt. Grün das Current Face im Current Room, mit Edge und Vert (die Raute) in gelb.

Lavendel
werden Portale angezeigt, wenn sie angeklickt sind (rechts).

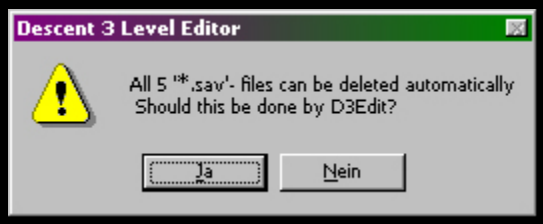


Update v40: die `.sa*`-Dateien

Seit Version v40 legt D3Edit beim Öffnen eines Raumes oder Levels eine Kopie desselben an, bis zu sechs Stück. Diese Dateien heißen `<levelname>.sav` und `.sa1` ....`sa5`. In dieser Reihenfolge steht auch das Alter der Sicherheitskopien, je höher die Nummer, desto jünger die Datei (rechts).

Dateiname	Erstellt	Geändert
terrprob.d3l	20.07.2008 14:08	01.10.2008 15:41
terrprob.gam	23.07.2008 14:03	31.08.2008 12:51
terrprob.qpd	26.08.2008 17:15	31.08.2008 13:41
terrprob.sa1	01.10.2008 15:52	31.08.2008 12:42
terrprob.sa2	01.10.2008 15:52	01.10.2008 13:38
terrprob.sa3	01.10.2008 15:52	01.10.2008 15:16
terrprob.sa4	01.10.2008 15:52	01.10.2008 15:21
terrprob.sa5	01.10.2008 15:52	01.10.2008 15:25
terrprob.sav	01.10.2008 15:52	31.08.2008 12:40
wide01.orf	23.07.2008 14:18	23.07.2008 14:18

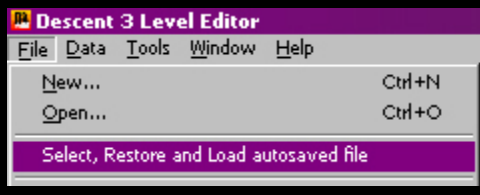
Wurde die maximale Anzahl an Sicherheitskopien erreicht, weist D3Edit darauf hin (unten links) und bietet daraufhin an, die Sicherungskopien automatisch zu löschen (unten rechts);



Wenn Ihr die Sicherungskopien behalten wollt, müßt ihr sie jetzt wegkopieren oder verschieben. Wenn Ihr sie verschiebt, müßt Ihr bei der Löschabfrage Nein klicken, da sonst D3Edit eine Fehlermeldung ausgibt und den Level oder Raum nicht öffnet.

Diesen Dialog bekommt Ihr angezeigt, wenn Ihr bei der Lösch-Abfrage 'Nein' klickt.

Wenn denn dann mal was schiefgegangen ist, Wählt File->Select, Restore and Load autosaved file und folgt den Anweisungen die dann kommen.



Es kann sein daß D3Edit wissen will ob die restaurierte Dateifassung ein Raum oder ein Level war, dann kommt dieses Dialogfeld:

gebt hier dann die entsprechende Endung an.

Damit gibt es erstmals ein Sicherheitsnetz.

Abschnitt A – Einen Level von Grund auf erstellen

Dies ist Descent3Fischleins Übersetzung von Schplurg's Walkthrough-Tutorials ‚The Level'.

001	**Erklärung des Editors**	Das soll eine kurze Erklärung zum Editor sein.	Schplurg	20
002	**Der erste Raum**	Hier wird gezeigt wie man "vertices" platziert, "faces" erstellt und die "extruding" Funktion erklärt.	Schplurg	23
003	**Den Raum ändern**	Texturen hinzufügen und den Raum verändern.	Schplurg	27
004	**Texturing**	Den Raum wird ein besseres Aussehen verliehen und Licht - Texturen hinzugefügt.	Schplurg	29
005	**Der erste Level**	Eine Erklärung wie man das Licht in den Raum berechnet und man sieht den Level das erste mal von innen.	Schplurg	31
006	**Einen zweiten Raum hinzufügen**	Zeigt wie man einen zweiten Raum hinzufügt und wie man einen Energietanker baut.	Schplurg	34
007	**Reaktor-Raum**	Hier könnt Ihr sehen wie man einen Raum auf eine andere Weise erstellt und zum Level hinzufügt.	Schplurg	37
008	**Reaktor** *ohne Script!*	Hier könnt Ihr sehen wie man mit dem Lathe Tool arbeitet, Dinge positioniert und Player Startpunkte hinzufügt.	Schplurg	40
009	**Lathe einen Raum**	Es wird wieder mit den Lathe Tool gearbeitet, nur daß ein Raum damit erstellt wird und dieser dann mit Hilfe der Snap Funktion Positioniert wird.	Schplurg	43
010	**Das Bend Tool**	Hier wird erklärt wie man gebogene Tunnel erstellt und wie man diese dann in den Level einbaut.	Schplurg	45

Dieser Walkthrough wurde von Robo verfaßt und ist etwas kürzer als Schplurgs ‚The Level'.

011	**Basics**	Grundbegriffe & die Ansichten	Robo	48
012	**Ein einfacher Raum**	Erster Start des Editors, erster Raum	Robo	51
013	**Ein Level anfangen**	Einen Level korrekt erstellen	Robo	54
014	**Texturierung**	Dem Raum Texturen verpassen	Robo	56
015	**Beleuchtung**	Lichter einstellen & berechnen	Robo	58
016	**Den Level testen**	Eine `.mn3` erzeugen und testen	Robo	62
017	**Neue Räume anfügen**	Erklärung der BOA & Räume attachen	Robo	64
018	**Objekte &Player Starts**	Essentiell für ein Level	Robo	66

Toc

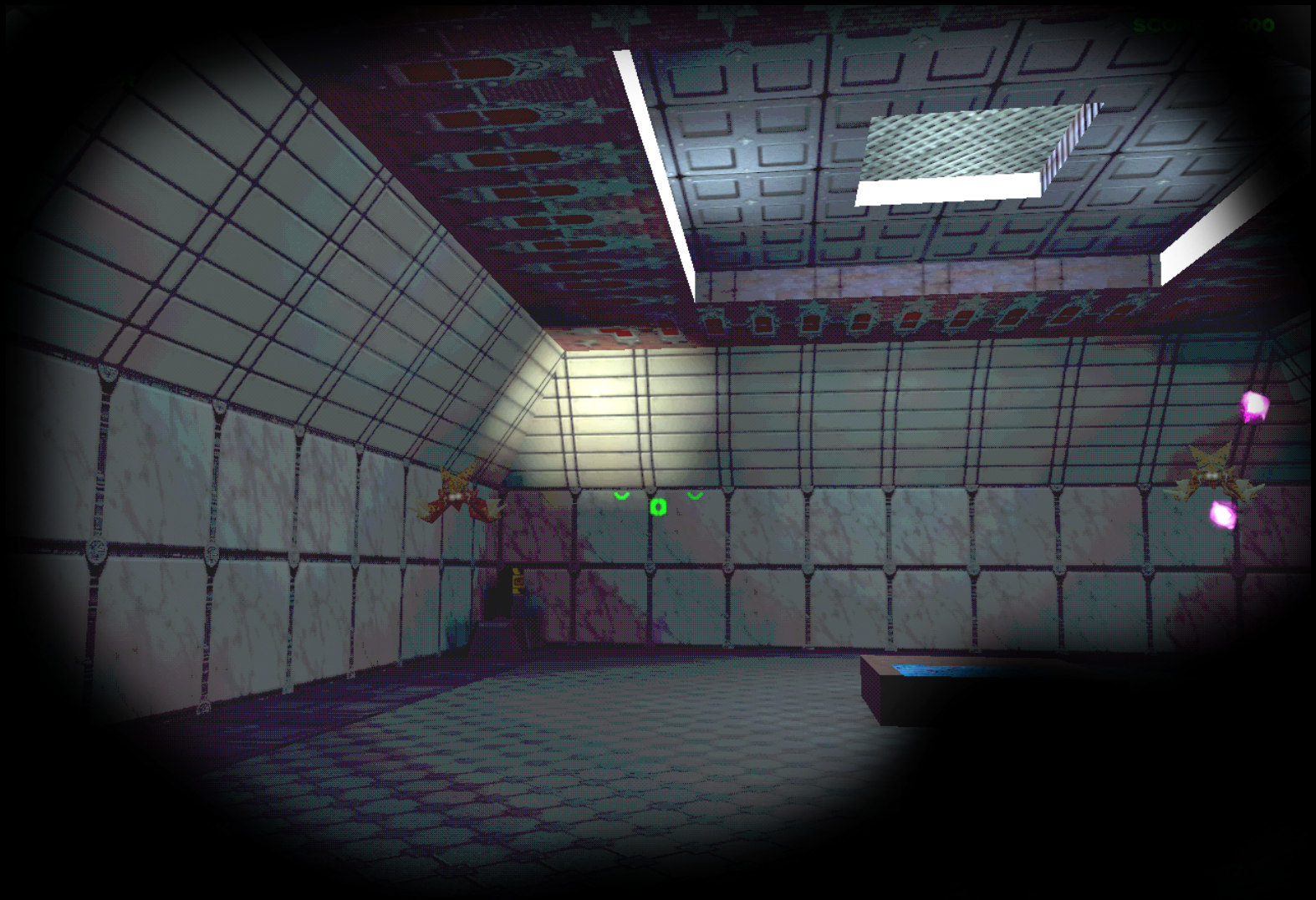
= THE LEVEL =

BY SCHPLURG

Dies ist ein kompletter Walkthrough, in dessen Verlauf ein vollständiges Descent3-Level entsteht. Descent3Fischlein hat die zehn Tuts ins Deutsche übertragen.

Es empfiehlt sich, daß alles GENAU so ausgeführt wird wie beschrieben, da ansonsten (bestenfalls) unvorhergesehene Resultate zu erwarten sind. Ich selbst hab' es mehrmals durchgearbeitet, bis ich ein ‚flüssiges' Gefühl entwickelt habe; dabei habe ich auch einiges an sehr seltsamen Fehlern produziert – Verify Mine: fehlerfrei, aber dann durch die Wände ausm Level fliegen. Interessant war auch, daß ich mal irgendwo zwischen ‚007 - Reaktor-Raum' und ‚008 - Reaktor' den Boden (im Wortsinne!) verloren hab' - keine Ahnung wo - und diesen im Nachhinein wieder einzubauen hatte. Jetzt, einige Monate später, weiß ich natürlich was ich falsch gemacht hab...

Wem es hier in den ersten fünf Tuts zu schnell geht, dem empfehle ich Robo's Walkthrough (ab Seite 47), welcher grundsätzlich dasselbe behandelt, jedoch etwas mehr in die Tiefe geht.



Packt euch noch'n paar fiese Bots mit rein, dann ist das Testen kurzweiliger

001 - Erklärung des Editors

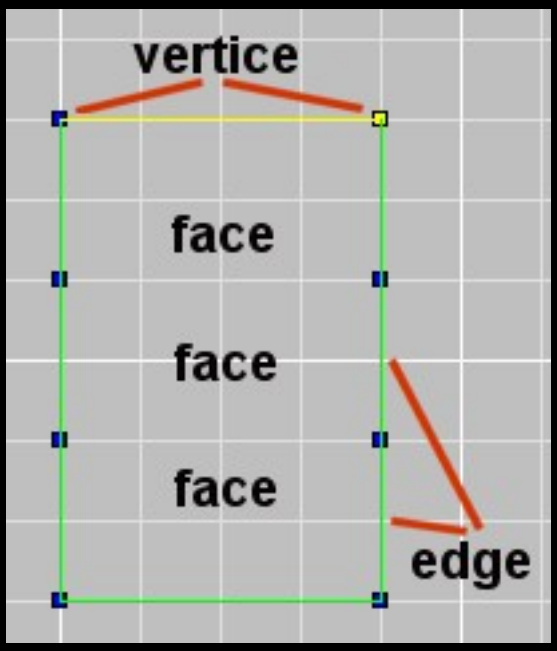
Schplurg

Ich gebe nicht wortwörtlich wieder, was Schplurg von Gameedit in englisch geschrieben hat!
(Descent3Fischlein)

Der Editor ist ein sehr gutes Programm, mit den man durchaus hochwertige Levels erstellen kann. Allerdings muss man ihn erst kennen lernen. Ich möchte hier den D3Edit einmal kurz beschreiben.

Faces

Ein Face ist zweidimensional und besteht aus 3 oder mehreren vertices. Stellt euch einen Raum vor in dem Ihr sitzt, schaut euch eine Wand an, sie hat 4 vertices und 4 edges. Es ist allerdings auch möglich ein Face mit mehrere vertices und edges zu erstellen.

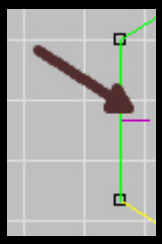


Beispiel:

Hier seht Ihr eine Art, wie ein Face aussehen kann. Ein Face braucht nicht nur 4 vertices zu haben. Es ist außerdem zu empfehlen, das man mehrere vertices ein Face hinzufügt oder gleich von Anfang an mit mehreren vertices arbeitet, weil man dann noch das Face splitten und die Texturen unterschiedlich gestalten kann. Dazu allerdings später. Die vertices müssen ausserdem auf der selben Ebene liegen, da man sonst ein "Nonplanar" Face erzeugt.
Ein Face ist eine Serie von vertices und edges, es gibt ein erstes, zweites, drittes .....

Shell

Ein Shell ist eine Gruppe von mehreren Faces, die den Raum abschließen, so das man nicht aus dem Level oder dem Raum hinausfliegen kann (das Level von aussen sehen). Das wäre dann ein "bad shell", dieses wird in D3Edit unter "Verify Mine" angezeigt. Wie man diese "bad shells" korrigiert, werde ich Euch unter einen anderen Thema erklären.



Normals

Dazu gibt es meiner Meinung nach nicht viel zu sagen. Eine "Normal" ist ein kleiner magentafarbener Strich, der angibt, von welcher Seite aus man ein Face sehen kann. Ein "Normal" sollte immer in das Innere des jeweiligen Raumes zeigen, da man sonst aus den Raum hinausfliegen kann.

Rooms

In jedem Descent3 Level gibt es wenigstens einen Room (sollten mehrere sein). Die Rooms sind mit Hilfe von "Portals" miteinander verbunden. Man kann die Rooms auch mit Hilfe von "Doors" verbinden, was eventuell auch besser ist.

Objects

Objects sind powerups, robots, player starts, barrels, soda machines. Diese kann man im Level überall platzieren.

Textures

Textures sind Bilder, mit denen Ihr Euer Level gestalten könnt. Mit Light Textures könnt Ihr Euer Level effektvoll ausleuchten, man muss allerdings überlegen, wo man eine Lichtquelle anbringt so daß es ein guter Effekt wird.

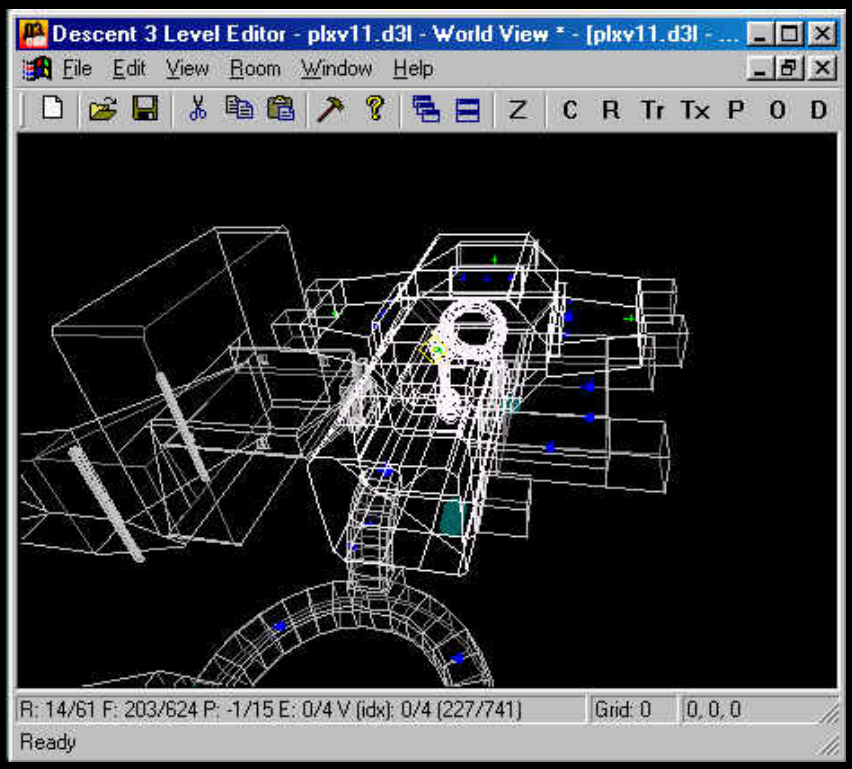
Modes

Vert mode - (**Strg+r**) - Vertices im Grid platzieren, auswählen, markieren, kopieren & einfügen, Vertices im Grid bewegen.

Face mode - (**Strg+f**) - Faces einfügen & löschen, auswählen, markieren, kopieren / einfügen, Faces im Grid bewegen

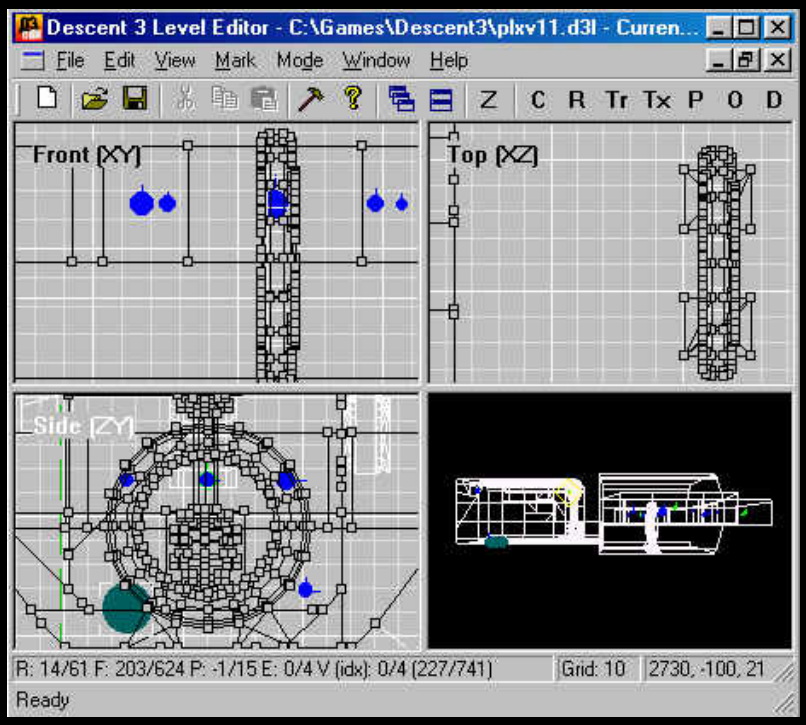
Object mode - (**Strg+g**) - Objekte einfügen, löschen und herumbewegen

Path mode - (**Strg+h**) – Pfade für Roboter anlegen, siehe Section 24.



Ansichten in D3Edit

World View Das ist die Ansicht, in der man den ganzen Level sehen kann (Bild rechts). Man kann es in alle Richtungen bewegen, neue Räume erstellen, Räume löschen und mit Textures ansehen. Das Bild ist mein Level Planet X. Wenn man in einen Raum klickt, wird er weiß hervorgehoben und die Ansicht wechselt in **Room View** zu den angeklickten Raum. Wenn ich jetzt im Menü View->Display Current room view anwähle dann seht Ihr das Bild links. Ich habe die Normals in dem Bild nicht angezeigt.



Room View oder Raum editieren. Hier sieht man 4 Fenster, einmal die Front von oben, von der Seite und die vierte ist ähnlich der von World View, außer daß man nur den aktuellen Raum sieht. Hier erstellt man Räume, formt sie, texturiert sie und fügt Objects ein.

Der Grid

Das ist der Raster im Hintergrund. Man kann die Rastergrösse verändern, wenn man einen sehr grossen Raum bearbeitet oder Vertices um eine bestimmte Masseinheit verschieben will; und zwar unter View->Adjust grid size/rotation angle -> auswählen oder unter Custom... eigene Werte eingeben.

Die Achsen

Man hat eine X, Y und Z Achse. Diese drei Achsen braucht man um einen dreidimensionalen Raum zu erstellen. Hier ein Bild:

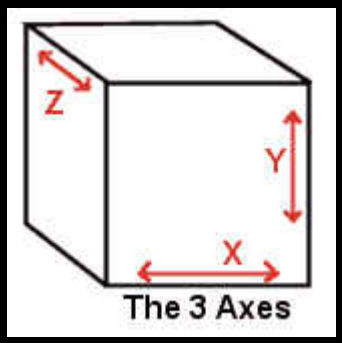


Bild Quelle: Gameedit

So das sollte fürs Erste reichen, die diversen Funktionen von D3Edit werden einzeln beschrieben.

002 - Der erste Raum

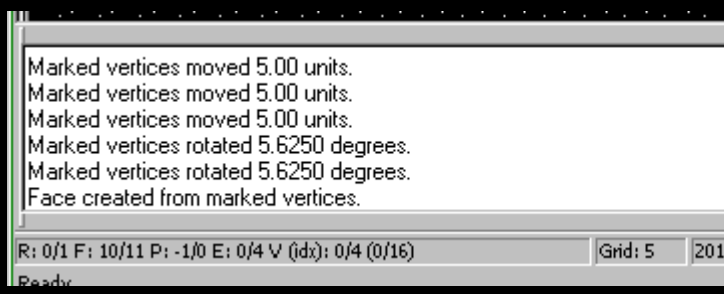
Schplurg

Hier erstellen wir nun Schritt für Schritt den ersten Raum. Ihr solltet eine Bildschirmauflösung von 1024x768 Pixel oder höher eingestellt haben, damit Ihr genug Platz zum Arbeiten habt.

Öffne D3Edit und Du müsstest folgendes sehen:

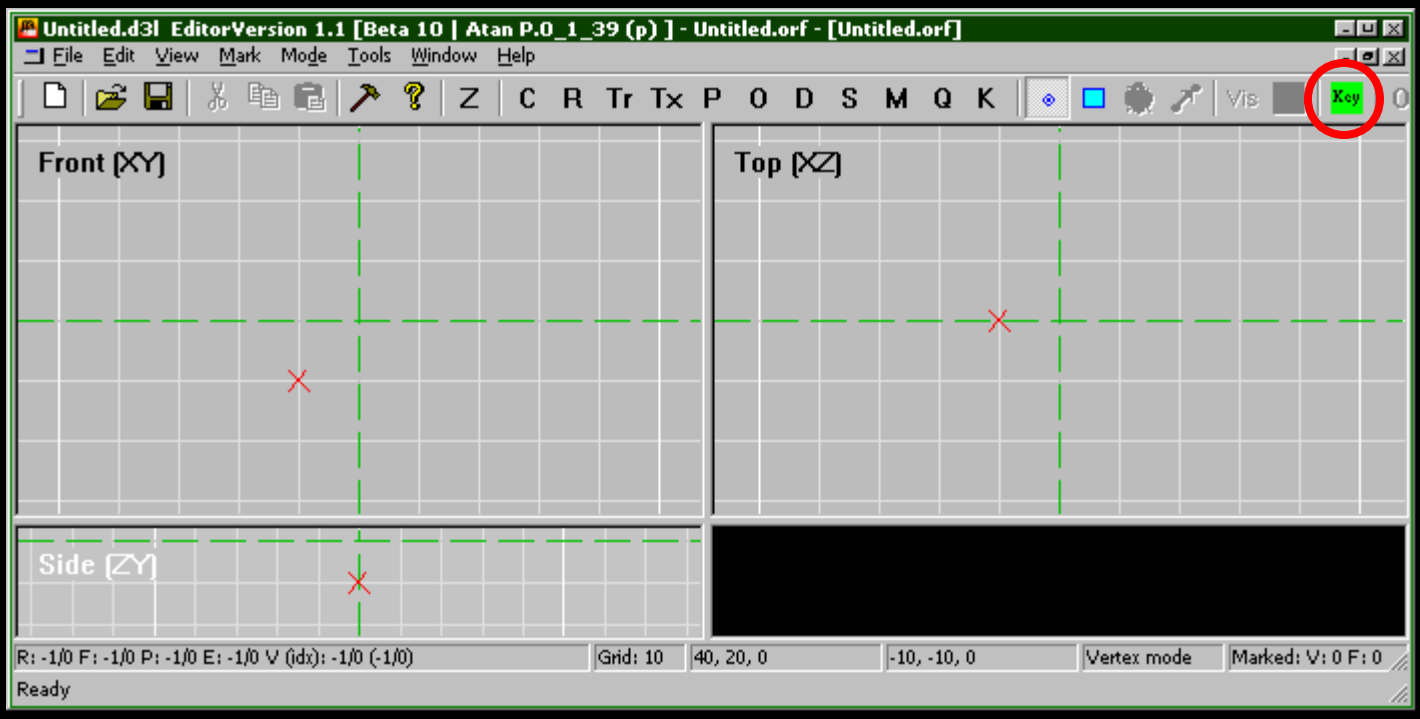
Wenn es irgendwelche Menüs gibt, die auf dem Hauptbildschirm geöffnet sind, können diese geschlossen werden, indem Ihr die Buttons in der Toolbar verwendet (hervorgehoben). Diese Buttons öffnen folgende Menüs:

Tx - Texture Menü
P - Path Menü
O - Object Menü
D - Doorway Menü
S - Sound Menü
Tr - Terrain Menü
R - Geometrie Menü (extruding, lathing, bending...)
Q - Quicktest-Menü, eine relativ neue Einrichtung um den Level schnell & ohne viel Aufwand zu testen
C - Camera Slewer Das ist die Menüleiste unterhalb der Standard Menüleiste (oben im Bild zu sehen). Diese braucht man eigentlich so gut wie gar nicht, denn man kann das auch mit Tastenkombis machen, dazu aber nach und n Muß noch behandelt werdenach.
Z - Schaltet den Z - Puffer ein oder aus. Die Texturen werden besser dargestellt, man braucht dazu aber mehr Systemleistung!
K – Ein Hilfsmittel, um über Auswahlkritereien zu bestimmen was an- oder abmarkiert werden soll (später mehr dazu). Man kann damit beispielsweise gezielt Faces auswählen, welche Fehler haben, oder auch T-Joints.
M – Messages, hier seht ihr was ihr zuletzt gemacht habt, bzw. das Programm (rechts).



Und los!

Wähle in Menü File->New... New Room und klicke auf OK. Jetzt hat sich der Room View geöffnet, maximiere das Fenster. Du siehst jetzt, das was im folgenden Bild dargestellt ist. Das Rote X ist immer die Einfügemarke und die grün /weiss gestrichelten Linien stellen den Mittelpunkt des Levels dar. Unten in der Statuszeile sieht man, welcher Grid eingestellt ist (Grid: 10), daneben wird die Position des cursors angegeben (-10, -20, 0) ein weiteres Feld daneben die Position der Einfügemarke (-10, -10, 0).



An dieser Stelle folgt erst einmal eine Hotkey - Liste , die werden wir jetzt brauchen.

Gibt's aber auch nochmal in komplett am Ende der File (Seite 518), und kann auch über den rot eingekreisten Button direkt im D3Edit eingesehen werden.

View Control
cycle through windows = ctrl + tab

pan = Shift + left mouse
zoom = Shift + Ctrl + left mouse
zoom = mouse wheel

scroll = left a
scroll right = d
scrolling = arrow keys
zoom in = w
zoom out = s

grid smaller = [
grid larger =]

Modes select
face ctrl + f
vertex ctrl + r
object ctrl + g
path = ctrl + h

place vertex (vertex mode) = ins
place face (face mode) = ins
place face (vertex mode) = shift + ins

Selecting next...
vertex = v
face = f
room = r
portal = p
select previous above items add shift + (key)

Mark items in vertex, face, and object modes
mark selected = space
unmark all = u
invert markings = i
mark all = m
mark worldview = m

Marking multiple items
Dragging a rectangle around vertices and faces
will mark all that lie entirely within the rectangle.
alt + drag to unmark

faces:n = flip marked face(s)

Moving marked items (vertices and objects in 2D views)
move = shift + arrow key

...numpad keys
left = 4
right = 6
up = 8
down = 2
twist left = 1
twist right = 3
objects rotate left = 7
objects rotate right = 9

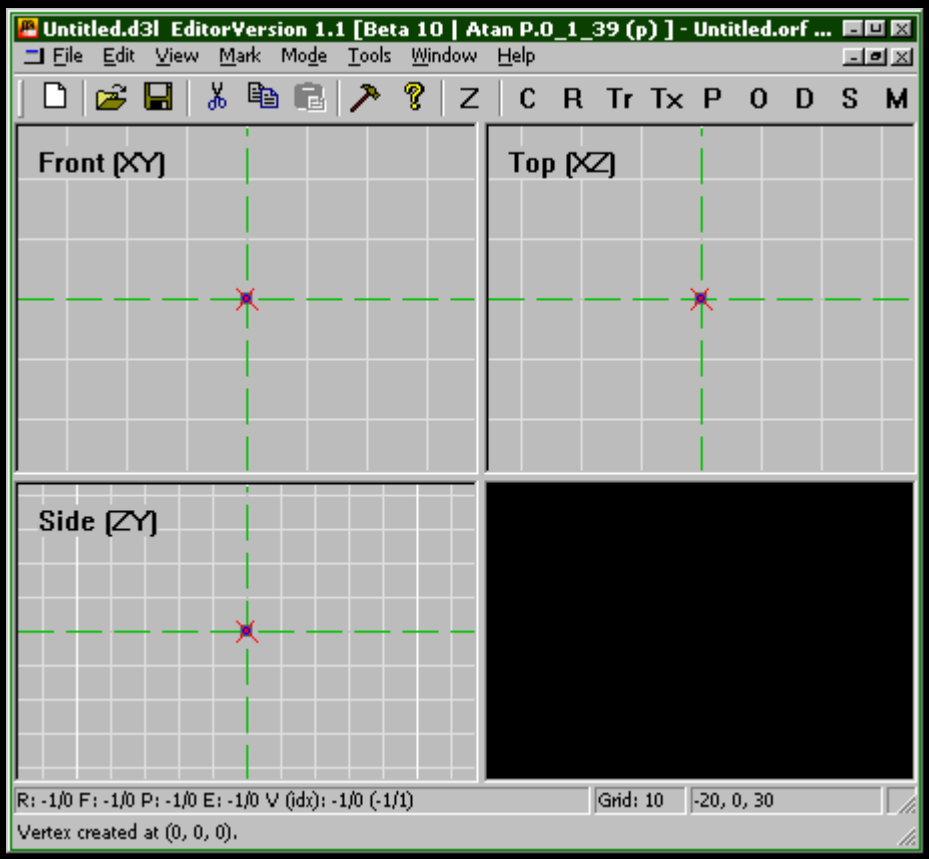
Editing
copy face = ctrl + c
paste face/verts = ctrl + v
paste on top (of copied item) = ctrl + shift + v

Moving placed rooms
rotate = alt + mouse
slide = alt + ctrl + mouse

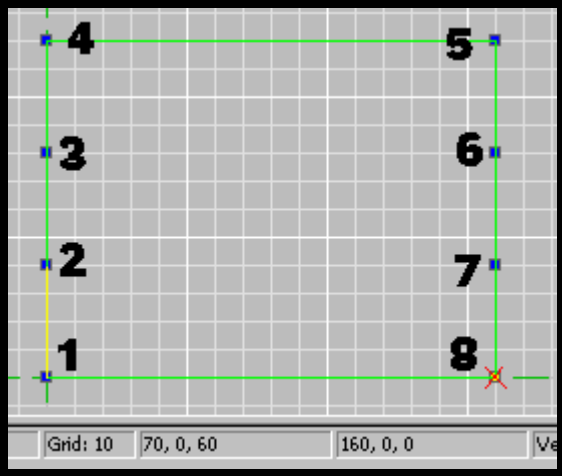
stop moving = alt

Other
save = ctrl + s

texture stretch = <
texture stretch = >

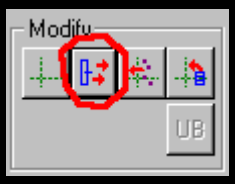


Jetzt wechseln wir zum Vertex Modus (**Ctrl+r**), setzt die Einfügemarke auf den Mittelpunkt bei "Top (XZ)", wie im Bild links. Nachdem das erledigt ist, drücke die Taste "**Einfg**" oder "**Ins**". Nun enthält der Raum den ersten Vertex, wiederholt das, bis Ihr so viele Vertices wie im Bild unten habt (auch die Reihenfolge). Wenn das vollbracht ist, müssen alle Vertices markiert sein ! (Taste **M** oder Mark->mark all)

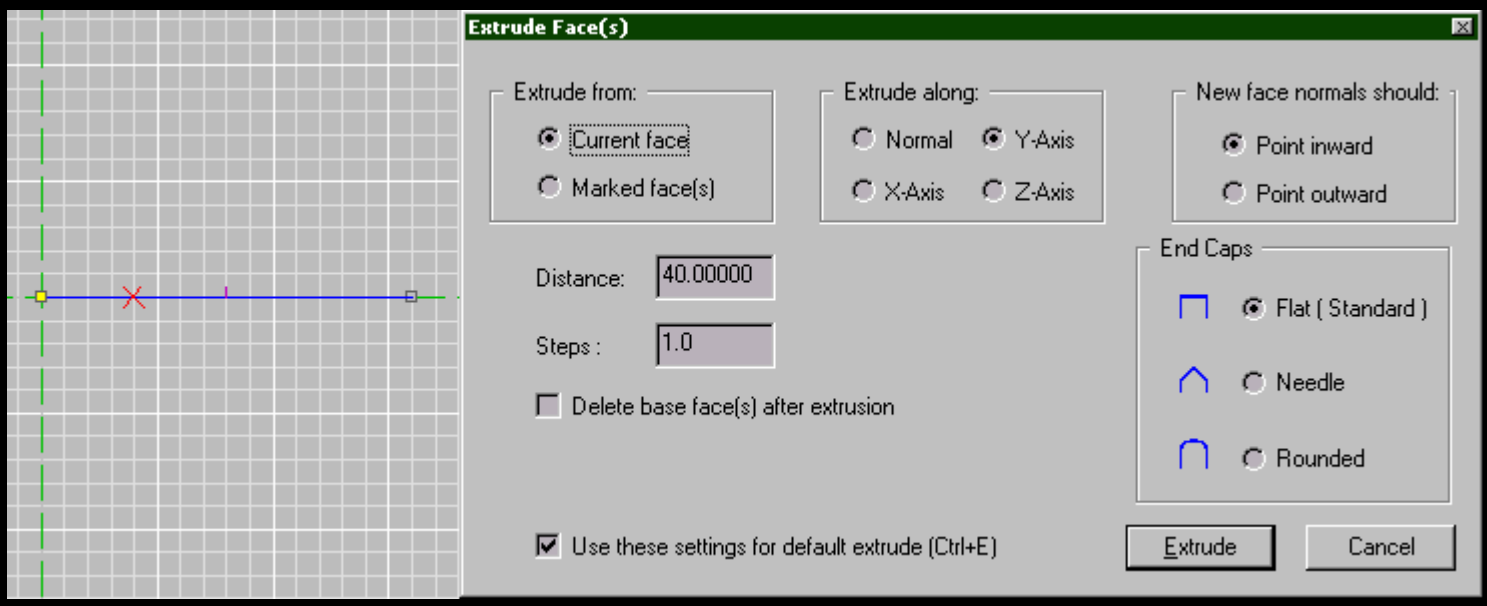


Jetzt in den Face Modus wechseln (**Ctrl+F**) und wieder "**Einfg**" oder "**Ins**" drücken. Schon habt Ihr das erste Face erstellt. Das Face ist jetzt das "**current face**".

Extrude



Als nächstes machen wir aus dem Face einen Raum. Drücke in der Toolbar den R-Button, ein Menü öffnet sich. Klicke nun auf die Schaltfläche, wie im Bild links gezeigt, es öffnet sich folgendes Dialogfeld wenn Ihr einen Editor älter als v40 habt:

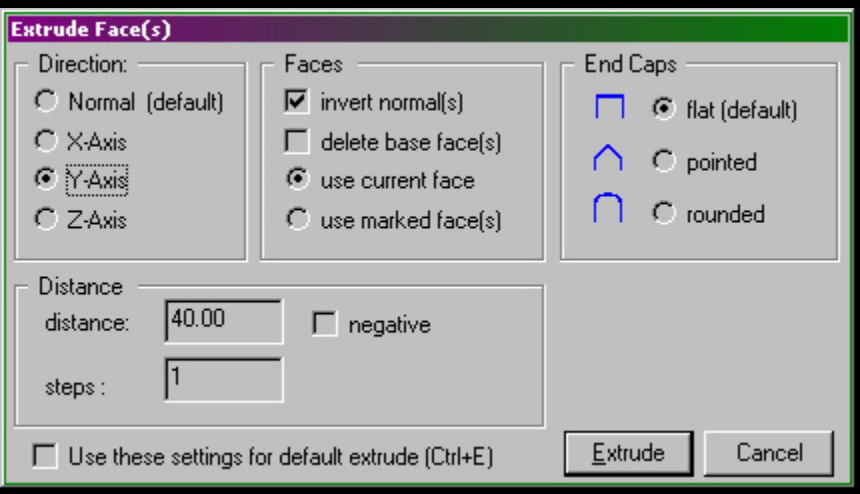


"Extrude from" hier könnt Ihr wählen um welches Face es sich handelt und wie hoch der Raum werden soll ("Distance"). Beachte, daß hier über positive bzw. negative Werte bestimmt wird in welche Richtung!
"Extrude along" in welche Achse soll der Raum extrudiert werden.
"New face normals should" in welche Richtung die Faces zeigen sollen.
"Delete base face(s) after extrusion" bestimmt, ob das Ursprungs Face gelöscht werden soll oder nicht.
"Use Settings as Default" kann man ein Häkchen setzen oder auch nicht; ist es gesetzt werden diese Werte für die Standard-Extrusion (**Strg+e**) verwendet.

Übernehmt die Einstellungen vom Bild und klickt auf die Schaltfläche Extrude. Der erste Raum ist von der Geometrie gesehen fertig.

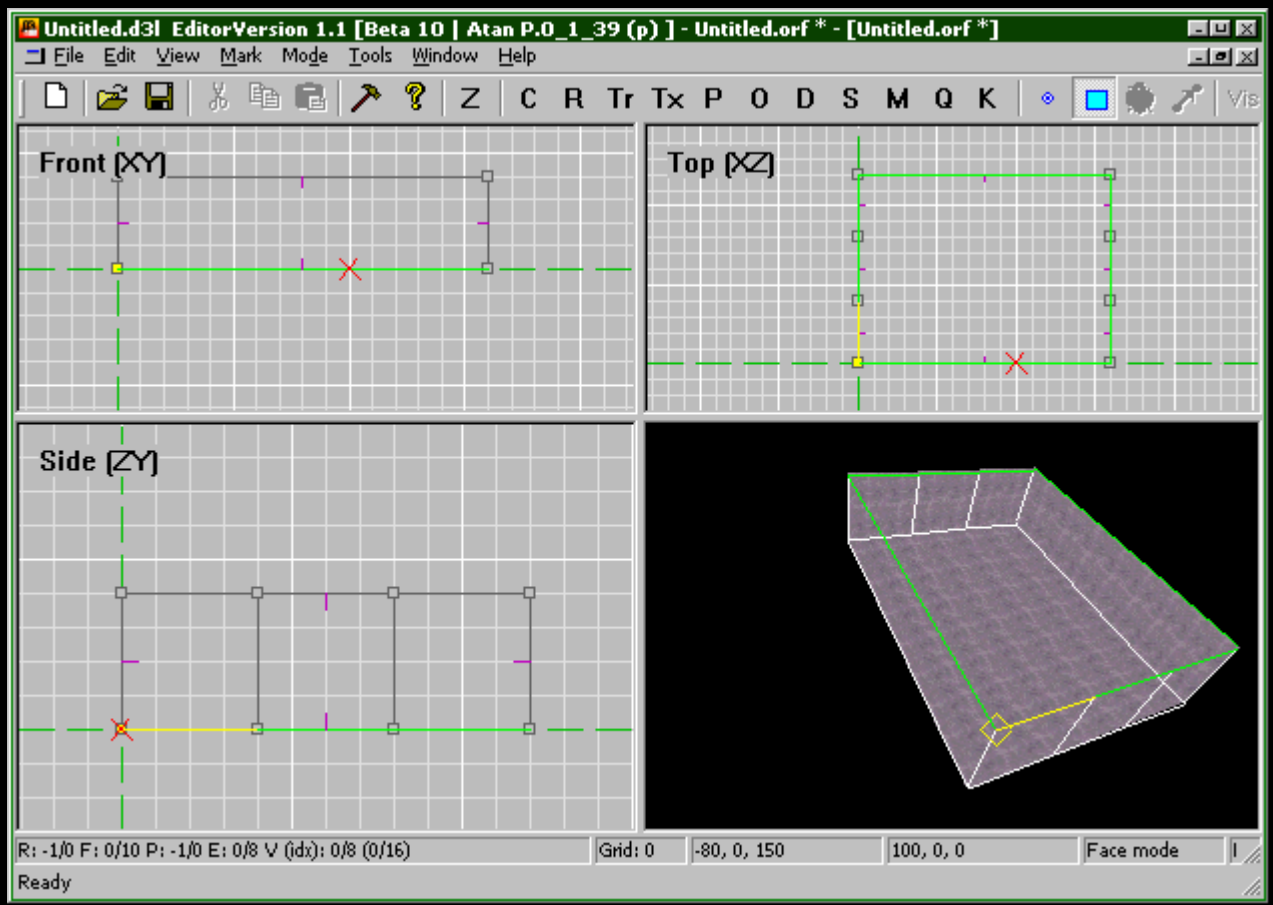
Update für die v40:

Der Extrude-Dialog wurde im Rahmen der Überarbeitung der Extrude-Funktion ebenfalls modernisiert, nehmt dann die Einstellungen aus diesem Bild.



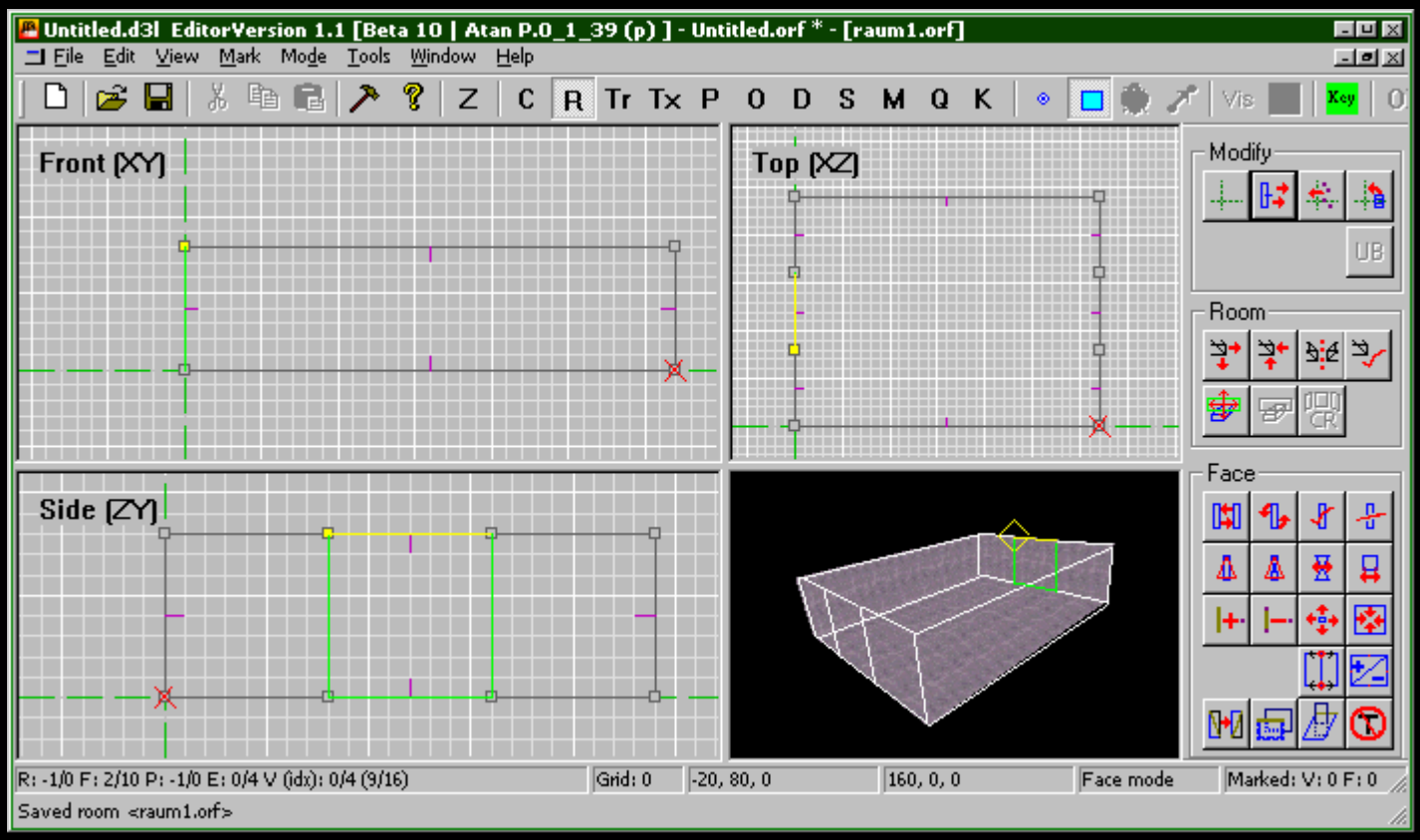
"Direction" Hier gebt Ihr an wohin die Extrusion gehen soll.
"Faces" Per default gucken die neuen Face-Normals immer in die gleiche Richtung, wie die des/r Ausgangs-Faces. 'invert normal(s)' dreht sie um.
"Distance" hier gebt Ihr an wie weit, wie oft und in welche Richtung extrudiert wird, ein Haken bei 'negative' kehrt die Richtung um.

So das war's, der erste Raum ist erstellt. Unter dem Thema "Den Raum ändern" geht es weiter! Speichert den Raum unter den Namen "`raum1.orf`" !!!



003 - Den Raum ändern

Schplurg

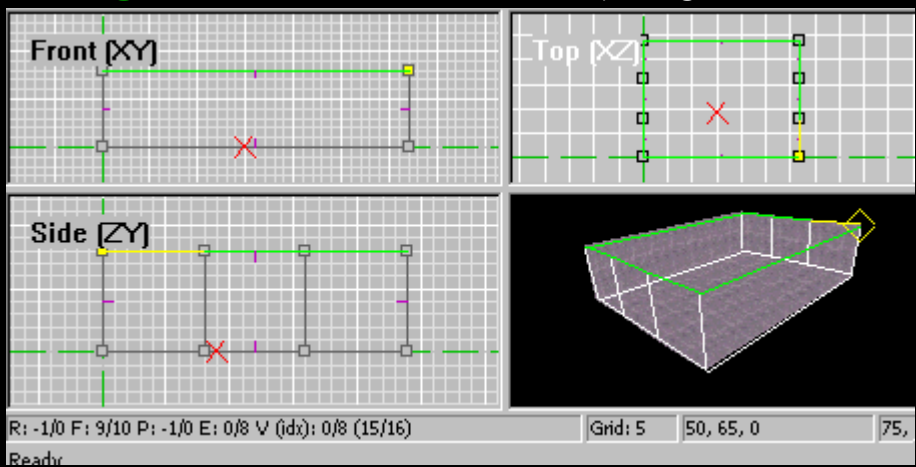


Dieser Teil schließt an das Thema "Der erste Raum" an! Öffnet D3Edit & anschließend Euren Raum (`raum1.orf`). Euer Raum sollte wie im Bild oben aussehen. Wir wollen uns den Raum einmal genauer ansehen. In "Der erste Raum" haben wir die Raumhöhe auf 40 Einheiten gestellt, nun wollen wir das einmal überprüfen. Die Mittellinie ist in der Front(XY) und Side(ZY) sichtbar. Anmerkung: ich habe den Grid auf 5 eingestellt (siehe Bild unten in der Statuszeile, "Grid: 5").

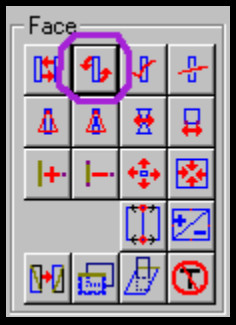
Es gibt 8 Rasterfeldquadrate zwischen der Decke und dem Fussboden, pro Rasterfeldquadrat 5 Einheiten (Grid 5) folglich ist der Raum 40 Einheiten hoch, da 8*5=40.

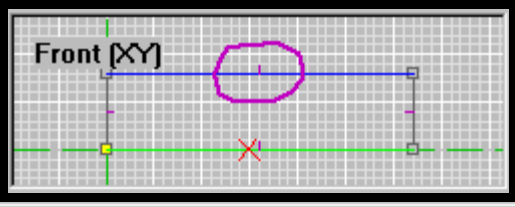
Beachtet das current face, das in dem Raum grün ist. Schaut euch die Ausgabe in der Statuszeile vom vorigen Bild an; der Bereich F:2/10 besagt: die erste Zahl ist die Nummer des current face, die zweite ist die Gesamtanzahl der Faces. Wenn Ihr jetzt die Taste **F** betätigt, wird das Face 7 zum current face. Es wird nach oben gezählt und fängt wieder bei 0 an. **Shift+F** machts genau umgekehrt 9...0.

Wir brauchen jetzt noch eine Lichtquelle in dem Raum, ansonsten würde man später nichts mehr sehen, wenn man in den Raum hineinfliegt. Wir verwenden noch einmal die Extrude - Funktion. Wählt das Face von der Decke aus (zuvor mit **Strg + F** in den face - Mode wechseln !!!), das geht wie folgt: klickt bei der Ansicht Top(XZ) bis das obere Face bei Front(XY) grün wird. Ihr könnt es auch mit der **F** - Taste erledigen ! Jetzt noch das Face mit Space, oder im Menü Mark.... Mark/unmark, markieren. Nun klickt Ihr einige Male bei Front(XY) ausserhalb des Raumes, jetzt könnt Ihr im Bild rechts sehen, daß das Face markiert ist.

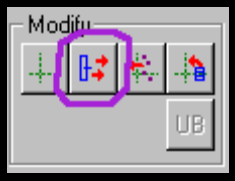


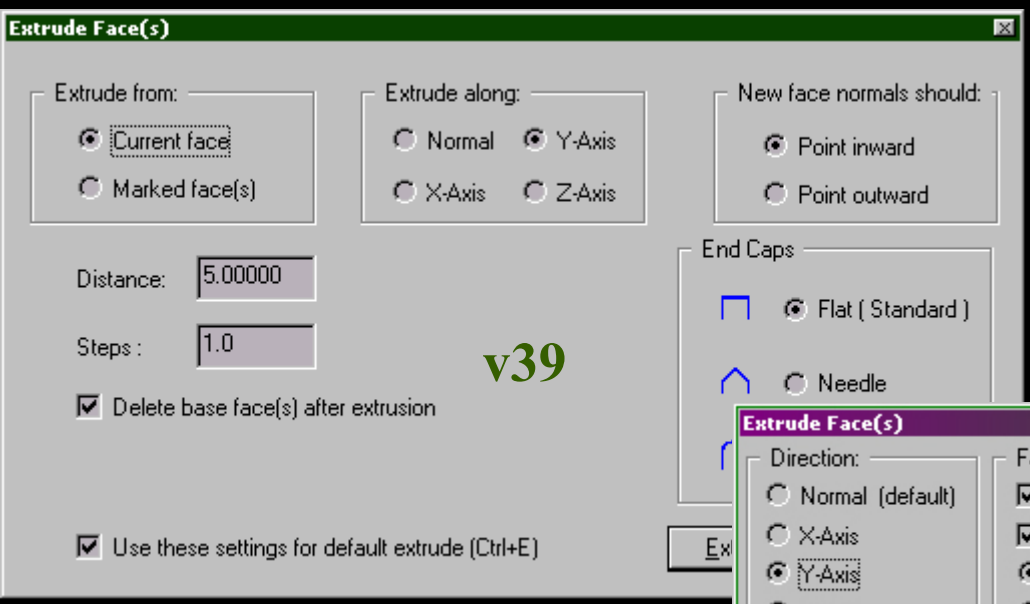
Im vorigen Bild ist das Face schon ausgewählt. Die Statuszeile unten zeigt F:9/10, das bedeutet, das current face ist 9. Beachtet das der "Normal" nach unten zeigt oder in den Raum. Als wir das erste mal die Extrude - Funktion verwendet hatten, zeigte der "Normal" nach oben. Wir müssen das Face umdrehen und das machen wir mit der **flip** - Funktion (Bild rechts, Tastenkürzel **N**). Im kleinen Bild unten seht Ihr das Ergebnis.



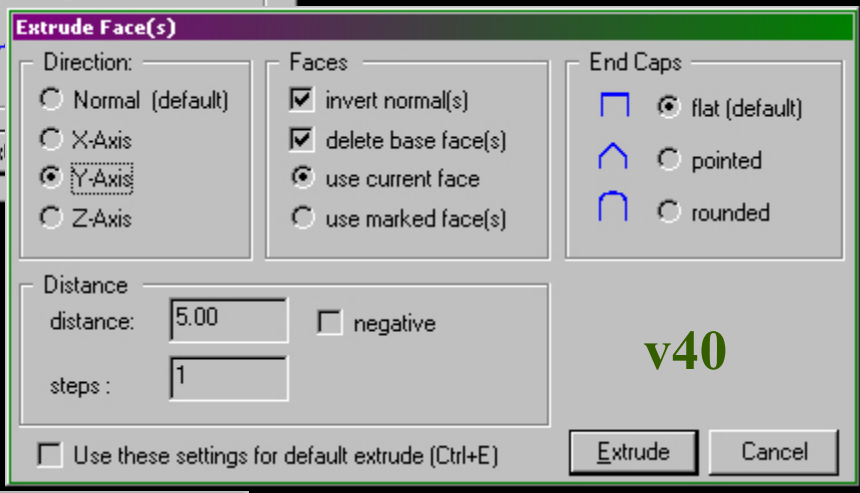


Jetzt können wir weitermachen. Macht das face 9 wieder zum current face, dann ruft die Extrude - Funktion auf:

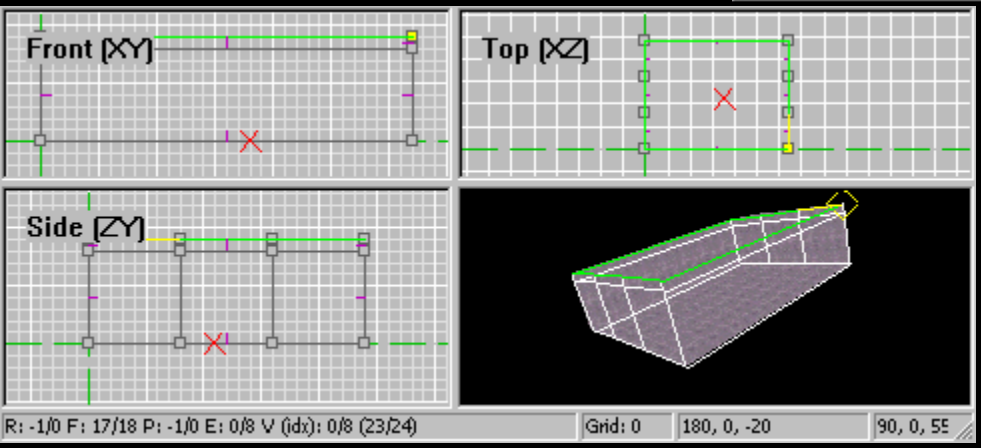




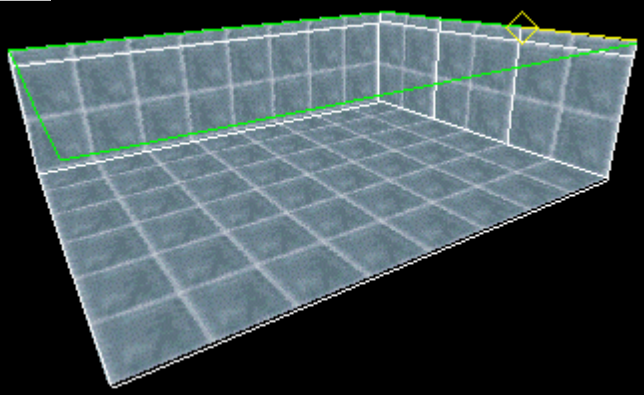
Tragt dasselbe ein was Ihr im Bild links seht. Wir werden den Raum jetzt um 5 Einheiten nach oben erweitern und wollen das das Ursprungs - Face gelöscht wird. Wenn Ihr mit der neuen Version arbeitet, nehmt das andere Bild.



Das Ergebnis seht Ihr weiter unten.
So, jetzt haben wir den Raum fast fertig. Wir müssen ihn jetzt noch texturieren, wechselt in den face Modus (**Strg+F**) und markiert alle faces indem Ihr **M** oder im Menü Mark... Mark all drückt. Schließt das Geometrie - Menü indem Ihr auf den



Button **R** klickt. Jetzt noch einen Klick auf **Tx** und das Texture Menü öffnet sich, hier ein Klick auf die Schaltfläche **Choose...**, im nachfolgenden Dialogfeld wählt **Metal Textures** und sucht dann nach der Texture **AdvMetalPanel1**. Jetzt auf die Texture klicken und per Drag und Drop in die Texture-Liste 'Custom' schieben. Schliesst anschliessend das Dialogfeld **Texture Palette** und klickt auf **AdvMetalPanel1**, in **Custom**, die Texture ist nun die aktuelle Texture. Klickt die Schaltfläche **To Marked**. Rechter Mausklick in die Raumvorschau und wählt Textured with Outline, jetzt könnt Ihr den Raum mit Texturen anschauen. Er sollte aussehen wie hier.



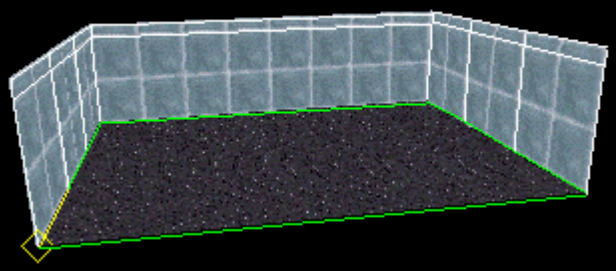
Speichert den Raum und fertig.

004 - Texturing

Schplurg

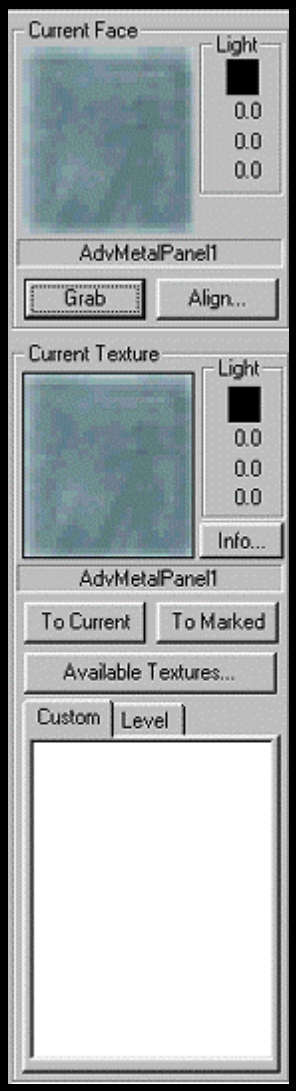
Alle Bilder die ich hier verwende sind von Gameedit

Zu diesen Teil benötigt Ihr "`raum1.orf`" aus dem Thema Den Raum ändern ! Öffnet D3Edit und anschliessend euren Raum.

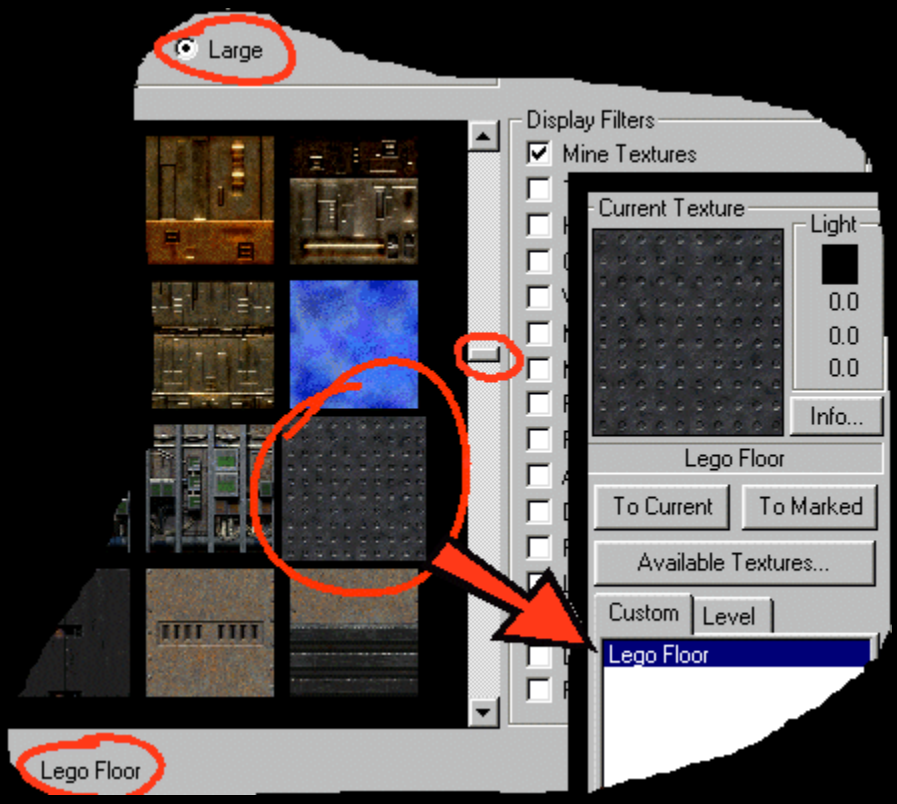


(Fussboden bereits geändert)

Zunächst wechselt mit **Strg+F** in den face - Modus, wir bleiben beim ganzen Thema in diesem Modus. Wir möchten etwas das Aussehen des Raumes verändern. Im Bild links ist bereits der Fussboden geändert, das möchte ich euch nun erklären.

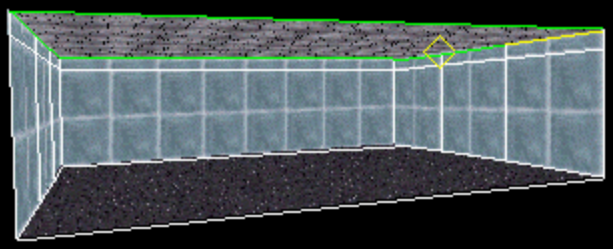


Verwendet die **f**- Taste um das face, wie im Bild zu sehen ist, zum current face zu machen. Ihr könnt das auch mit der Maus erledigen. Klickt den Button **TX** an um das Texture - Menü zu öffnen, das sollte dem rechten Bild ähneln. Jetzt klickt auf Choose (neuere Versionen sonst Available Textures). Im nachfolgenden Fenster (**Texture Palette**) wählt Large und klickt **Mine Textures** an. Dann sucht die Texture '**Lego Floor**', diese zieht Ihr in das Feld, siehe Bild unten.



Jetzt klickt Ihr die Texture in dem Feld an und dann auf die Schaltfläche To Current ,minimiert das Fenster Texture Palette und schaut euch den Raum im Room View an, er sollte so aussehen wie im ersten Bild. Wir texturieren nun noch die Decke. Sucht in dem Fenster Texture Palette nach '**LWceil.tga1**'. Jetzt müsst Ihr die Schritte, wie oben beschrieben, nur noch wiederholen.

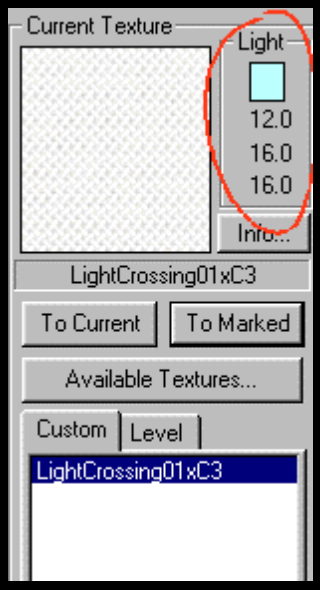
Sieht Euer Raum jetzt so aus wie in Bild unten mitte? Gut! Jetzt beleuchten wir den Raum noch etwas.



Beleuchtung - die kurze Version

Die Beleuchtung hier näher zu erklären würde das Thema sprengen. Ich werde das unter einem anderen Thema noch mal ausführlich erklären, denn man kann sein Level mit der richtigen Beleuchtung eine besondere Note verleihen. Siehe mein Level Planet X. :-)

Zunächst löscht alle Häkchen in dem Fenster **Texture Palette** (wir haben diese vorhin minimiert!) unter **Display Filters**. Jetzt klickt Ihr auf **Light Textures** und sucht die Texture '**LightCrossing01xc3**'.



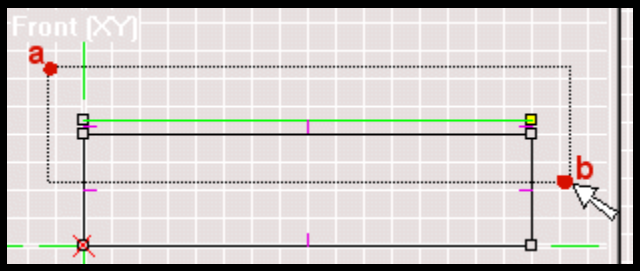
Jetzt schaut Euch das rechte Bild an, der rot eingekreiste Bereich gibt Auskunft über die Farbe und RGB Farbwerte (die Zahlen unterhalb von Light). Wenn Ihr auf den Button **Info**... klickt, erhält man noch etwas mehr Information über die Texture, aber man kann da nichts ändern. Die RGB - Farbwerte sollen dabei behilflich sein die Farbe und Helligkeit abschätzen zu können. Es ist ein Unterschied ob man einen grossen oder kleinen Raum beleuchten muss.

Jetzt wisst Ihr warum wir den Raum im vorherigen Thema geändert haben. Wenn wir eine grössere Fläche dazu genommen hätten, wäre es viel zu hell in den Raum ! Das müsste ausreichen um den Raum ausreichend zu beleuchten.

Texturierung mehrerer faces

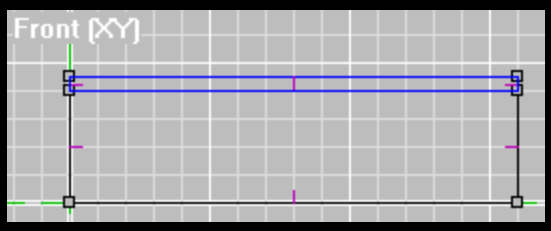
Es gibt einige Weisen, wie man ein face texturieren kann. Man kann jedes face einzeln texturieren, wie wir es mit dem Fussboden und der Decke gemacht haben, oder man texturiert mehrere faces auf einmal.

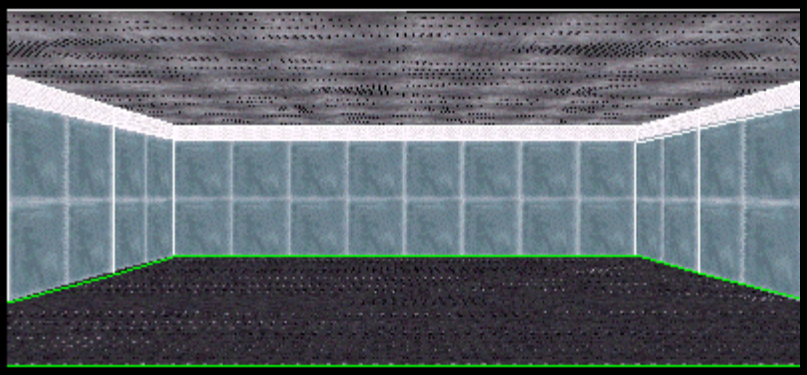
(Farben geändert für bessere Ansicht)



Faces und verts können in den jeweiligen Modus markiert werden, indem man mit der Maus ein Rechteck um die zu markierenden Faces oder Verts zieht. Macht das selbe wie im linken Bild (wir sind immer noch im face - Modus). Das a steht für Anfangspunkt und b steht für Endpunkt. Wenn Ihr bei dem Punkt b angelangt seid, lasst die Maus - Taste wieder los. Das Resultat seht Ihr im Bild unten rechts.

Aber Moment ! Wir habe ja auch die Decke mit markiert, diese müssen wir demarkieren. Klickt in der Draufsicht solange bis Ihr das Decken face als aktuelles face habt (grün umrandet), jetzt Leertaste betätigen oder im Menü Mark... Mark/Unmark. Ihr könnt die Methode wählen die Euch am einfachsten erscheint. Wenn Ihr das erledigt habt, dann klickt im Texture - Menü auf den Button **To Marked** ,fertig.





Das Resultat seht Ihr im Bild links.
Jetzt solltet Ihr den Raum Speichern, denn in D3Edit gibt es keine "Rückgängig" Funktion!

In dem nächsten Thema werden wir den Raum als Level (`mn3`) abspeichern und die Lichter berechnen. Es wird heissen "Der Erste Level".

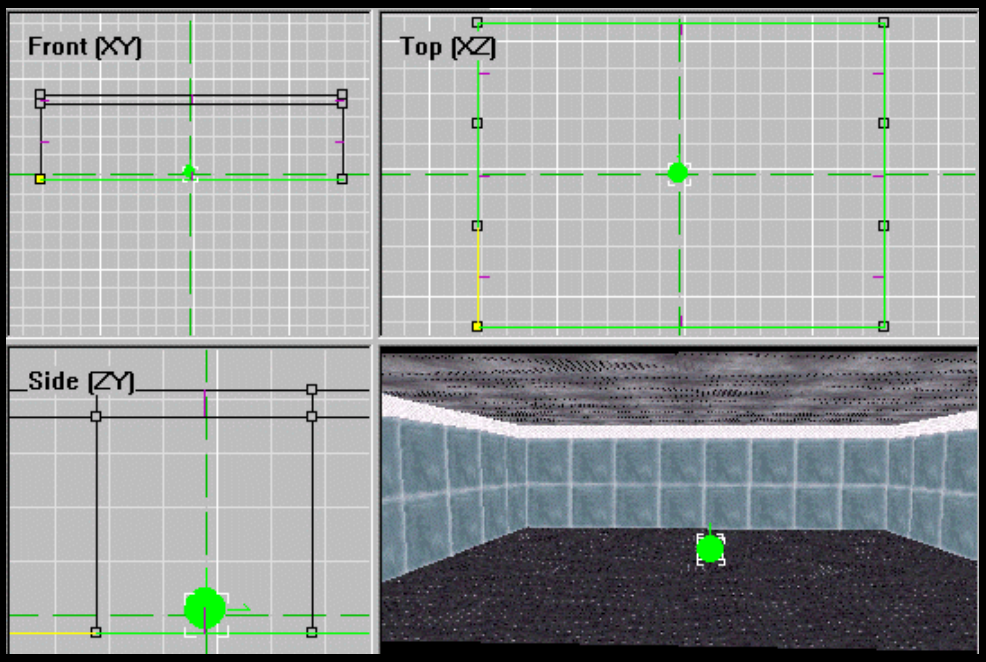
005 - Der erste Level

Schplurg

Alle Bilder die ich hier verwende sind von Gameedit

Für dieses Thema benötigt Ihr den Raum aus ‚Texturing'! Öffnet D3Edit und dann den Raum "`raum1.orf`".

Wählt im Menü File/New.../New Level - Import Room und auf **OK** klicken. Im folgenden Dialogfeld sucht Eure Datei "`raum1.orf`". Dieser Raum müsste so aussehen wie im Bild hier unten.



Hier sehen wir das ein grüner Punkt hinzugekommen ist, das ist der erste Playerstartpunkt. In jedem Level gibt es diese Punkte, es können mehrere sein aber mindestens einer.

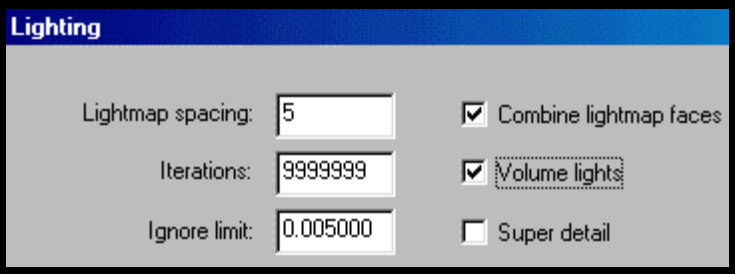
Im Menü Window, das aussieht wie hier (euer Menü kann etwas anders aussehen) beschäftigt uns vorläufig nur der untere Teil des Bildes.

Hier könnt Ihr navigieren, wenn Ihr mehrere Räume geöffnet habt. Da behält man die Übersicht welchen Raum man gerade bearbeitet. Jetzt auf Untitled.d3l - World View klicken, nun sind wir in der Weltansicht. Hier kann man seine Level komplett in 3D anschauen oder zu einem anderen Raum wechseln und diesen zum Current Room machen. Nun speichern wir erst einmal den Level, denn D3Edit neigt zu abrupten Abstürzen. Wählt im Menü File/Save as..., sucht das Verzeichnis in dem Ihr die Levels speichert und benennt den Level "`thelevel.d3l`".

Nachdem Ihr das getan habt, wieder zurück ins File Menü und wählt Edit Level Info....(neuere Editoren: File > Level Settings...) In diesem Fenster könnt Ihr unter anderem den Namen für Euer Level festlegen; den seht Ihr dann wenn Ihr den Level startet bzw. ingame (Karte, etc.). Füllt die Felder aus und klickt auf **OK**.

Suche nach Fehlern

Geht zurück ins File Menü und wählt Verify Mine. Dieser Menüteil ist extrem wichtig zur Fehlersuche! Wenn hier alles auf 0 steht dann habt Ihr perfekt gearbeitet. In dem unteren Teil seht Ihr **BOA: NOT VALID**, wenn Ihr den Level jetzt spielen würdet, bekommt Ihr eine Fehlermeldung beim Start des Levels "BOA not valid!", das werden wir gleich ändern. Geht wieder ins Menü File und dort wählt Ihr Compute BOA Vis Table, das nächste Dialogfeld bestätigt Ihr mit **OK** und schon ist die Table berechnet. Jetzt noch einmal Verify Mine, da könnt Ihr jetzt sehen **Boa: valid**. Speichert nun erneut den Level! Wenn Ihr einen Raum hinzufügt, dann wiederholt diesen Vorgang. Schliesst jetzt den Raum "`raum1.orf`". Die Frage ob Ihr den Raum speichern wollt, könnt Ihr mit nein beantworten. Wir haben ja nichts an den Raum geändert.



Jetzt machen wir Licht in dem Level

Hier werde ich Euch erklären wie man das Licht berechnet. Geht in das Menü Window und wählt dort Lighting. Ich erkläre erst einmal die einzelnen Eingabefelder.

Lightmap spacing - ist der Abstand zwischen den Licht Pixeln. Höhere Einstellung macht gröberes Licht, kleinere Werte macht realistisches Licht - und Schatteneffekte. (= die Auflösung der Lichter)

Iterations - gibt an, wie viele Lichter berechnet werden sollen. Hier sollte man immer die höchste Zahl wählen, da man dann sicher ist, wirklich alle Lichter berechnet zu haben und daß der Level besser aussieht.

Ignore limit - Licht wird nicht errechnet, sobald die Intensität/Helligkeit unter den angegebenen Wert fällt. Je niedriger die Zahl desto realistischer.

Combine lightmap faces - D3Edit versucht 2 oder mehrere Faces in ein einzelnes "Lightmap" zu kombinieren. Lasst hier auf jeden fall ein Häkchen drin.

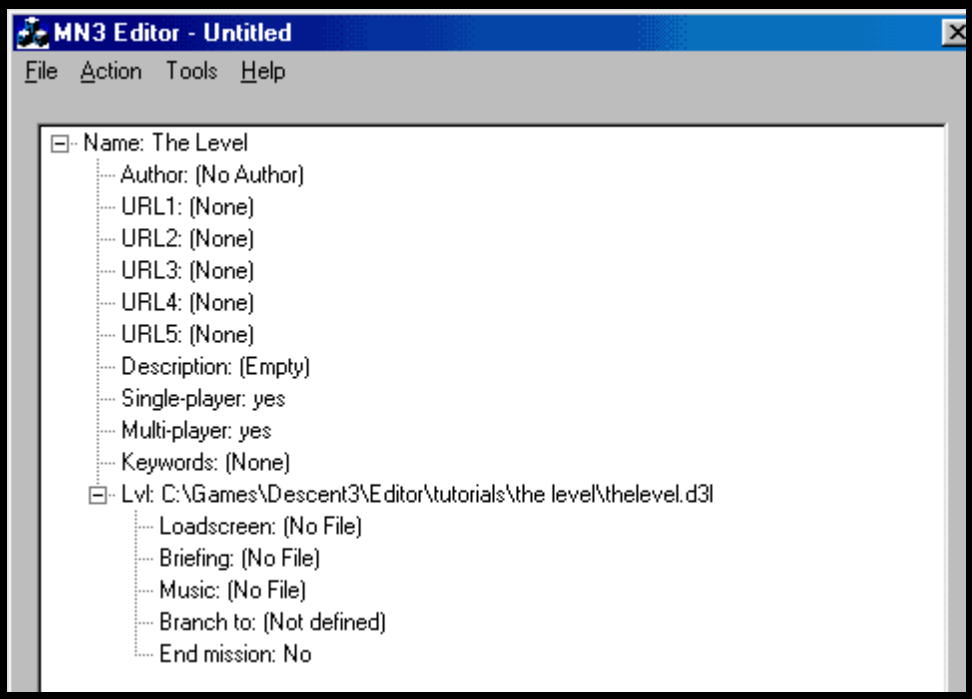
Volume lights - gibt an, das eine dunkle Ecke auch dunkel bleibt. Man kann sich dann in dieser dunklen Ecke auch mit einem Schiff verstecken. Ich habe es so bei Planet X gemacht.

Super detail - setzt die Detail-Stufe feiner. Ist nicht unbedingt zu empfehlen, da die Lichtberechnung erheblich länger dauert. Könnt es ja mal ausprobieren.

Setzt alle Werte so wie im Bild rechts oben. Anschliessend klickt Ihr in der Box **Entire Mine** auf **Count Lights**, im nächsten Dialogfeld auf **NEIN** klicken (ich möchte, das Ihr seht, wie die Lichter gezählt werden). Klickt nun auf die Schaltfläche **Light It**. In den folgenden Dialogfeld klickt auf **Yes**. Nun kommt noch ein Dialogfeld, das Euch sagt, das die Berechnung sehr lange dauern kann. Es kommt darauf an, wie gross Eure Level ist, bei Planet X, mit Aussenwelt, hat es 18 Stunden gedauert und ich habe einen P3 500 MHZ. Wenn die Berechnung anfängt, denkt man D3Edit wäre abgschmiert, lasst Euch nicht täuschen, D3Edit fängt mit der Berechnung an! (wenn Ihr ein Tool habt, daß die Prozessor- Auslastung anzeigt, könnt Ihr sehen, das noch gearbeitet wird) Nachdem das erledigt ist, speichert den Level.

Der MN3 Packager

Eine Level muss als `mn3` gespeichert werden oder man kann auch sagen gepackt werden. Das geschieht mit dem **MN3 Packager**. Man kann dem Level auch ein Startbild, Musik oder Dateien zufügen, das macht man mit dem **MN3 Packager**. Diesen findet Ihr im Menü File.../MN3 Packager. (neuere Versionen: Tools > mn3edit)
Klickt mit der rechten Maustaste auf Name, im folgendem Menü auf Edit mission information. Hier gebt Ihr der Level einen Namen, (der Name den man dann in Descent3 in der Levelliste sieht) tragt die Information über den Autor ein und klickt auf OK.



Jetzt wählt Ihr im Menü Action.../Add level. Nun auf Browse, unter Level filename, sucht `thelevel.d3l` und **OK**. Der MN3 Editor sollte wie im Bild oben aussehen. Die URL-Angaben sind erst einzutragen, wenn der Level komplett fertig ist. Das heisst, den Level zu meiner Homepage oder irgendeiner anderen uploaden, um ihn über diese URLs dann downloaden zu können (im Vortex zB). Da erfahrt Ihr auch welche URL eingetragen wird. Speichert nun den Level**,** File/Save as... geht zu Euren Missionsverzeichnis (Euer `Descent3/missions` Verzeichnis) und speichert den Level unter dem Namen "`derlevel.mn3`". Schliesst den MN3 Editor und D3Edit.

Startet nun Descent3, **New Game** und startet **The Level**.
Hier seht Ihr noch einmal ein Bild von Eurer Level. **Gratulation** das ist Euer erster Level, den Ihr erstellt habt !!!!

Update:

Heutzutage geht das mit dem Levelpacken wesentlich einfacher, Dank dem Quickedit. Es auf die alte Weise zu machen ist als Erfahrungswert allerdings nicht wettzumachen

006 - Einen zweiten Raum hinzufügen

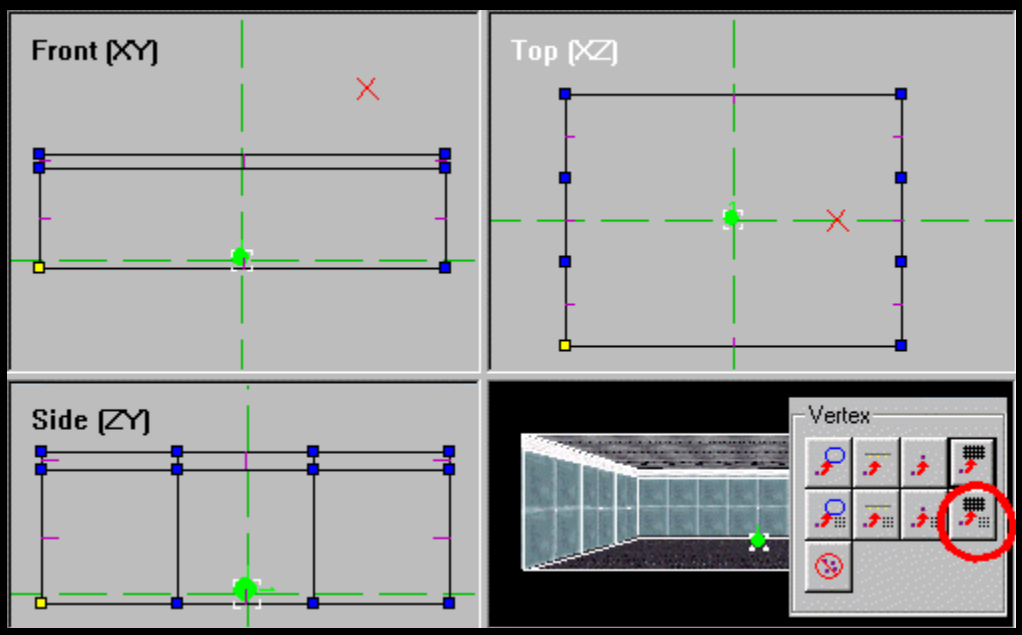
Schlplurg

Für dieses Thema benötigen wir die Datei "`thelevel.d3l`" von **Der erste Level**.
Öffnet D3Edit und anschliessend "`thelevel.d3l`"

Anmerkung: Wir müssen zunächst einen Fehler beheben, in unseren vorher erstellten Level. Jedes Mal, wenn Ihr eine Level erstellt, müsst Ihr den Raum am Grid ausrichten, sonst liegt Eure ganze Level neben dem Grid. Das sollte jeder Leveldesigner berücksichtigen !!!

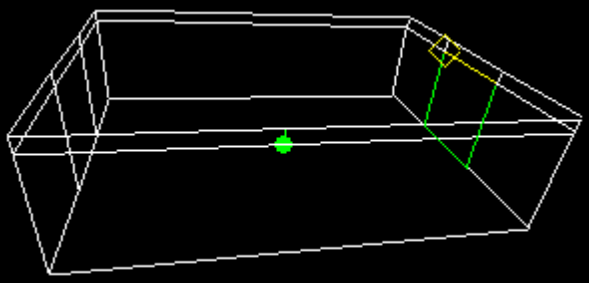
Wir hatten einen Raum mit der New Level-Import Room-Funktion in ein neues Level eingefügt und der Raum wurde aber nicht genau am Grid, sondern zwischen zwei Rasterfeld-Punkten angefügt. Angenommen, Ihr versucht andere Räume zu erstellen oder zu bearbeiten, dann kann es zu Fehlern kommen.

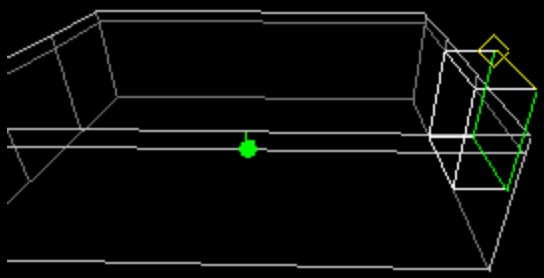
Was wir nun machen müssen um den Fehler zu beseitigen: Zunächst schaut nach, ob Ihr Euch im **vert Modus** befindet (**Strg+R**). Jetzt markiert alle verts mit der Taste **M** oder macht es mit der Maus, linke Taste gedrückt halten und von links oben nach rechts unten ziehen. Achtet darauf, das dann ein Vert gelb ist (das ist das Current vert), dieses vert wird anschliessend auf den Grid ausgerichtet und nimmt alle markierten Verts mit. Schaut Euch mal das rechte Bild an, so müsste es aussehen.



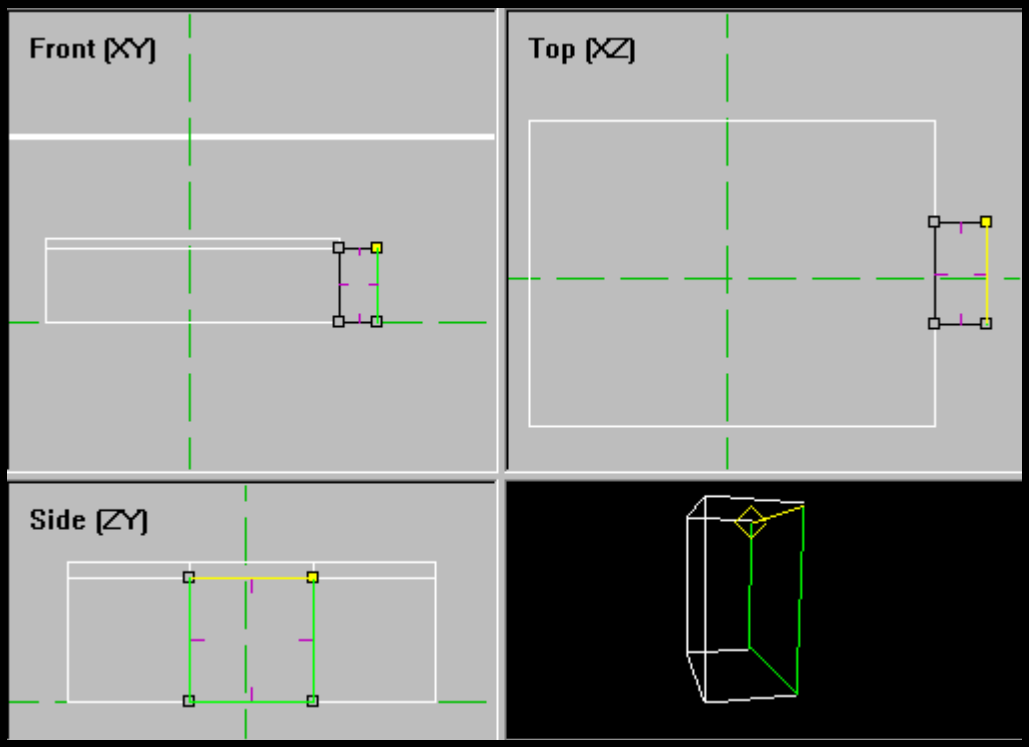
Öffnet nun die "**Geometrie Bar**" mit dem Button **R**. Wir finden unter Vertex zwei Schaltflächen , zu denen es keine Info in der Statuszeile gibt. Uns interessiert der untere von den beiden, im Bild rot umrandet. Wenn alle verts markiert sind, klickt auf diese Schaltfläche und wir haben die erste Achse ausgerichtet. Wiederholt den Schritt noch in den zwei anderen Fenstern, damit der Raum in jeder Achse am Grid liegt. Nun ist unser Raum am Grid ausgerichtet.

Dank an **Sirian** für diesen Tipp!
Zurück zu dem zweiten Raum! Achtet darauf das Ihr Euch in World View befindet. Es gibt mehrere Methoden Räume hinzuzufügen, die einfachste ist die Funktion Add Room at Current Face. Ihr findet sie im Room Menü ganz oben. Macht das Face , wie im linken Bild, zum Current Face (grün umrandet), dann wählt Add Room at Current Face, das Resultat sieht dann wie im rechten Bild aus.

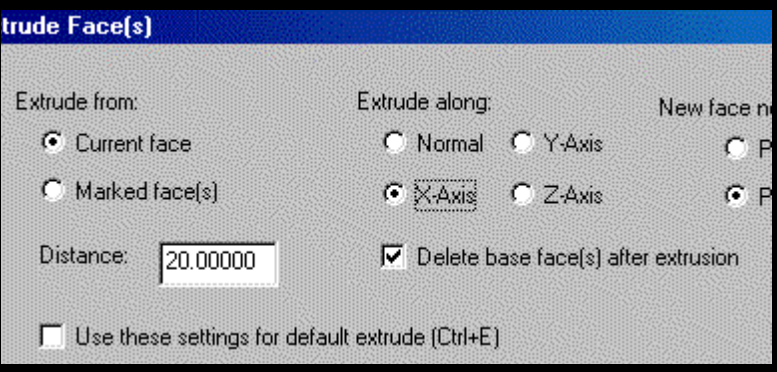


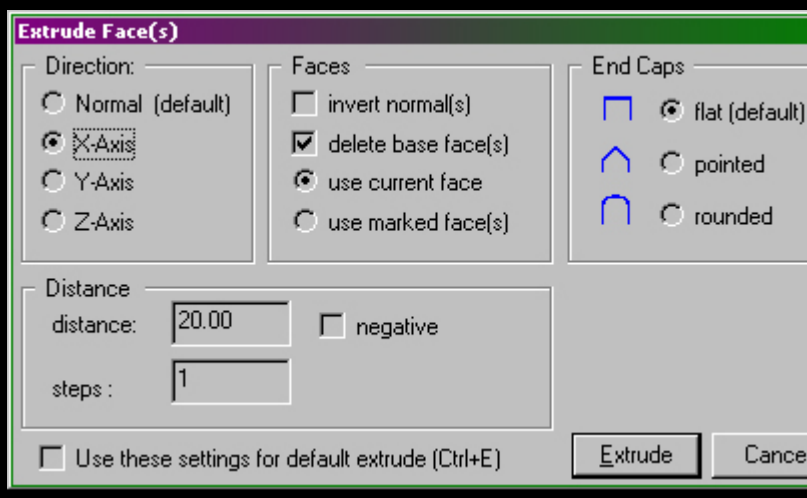


Beachtet, das der neue Raum jetzt der Current Room ist. Klickt mit der rechten Maus-Taste und wählt Display current room view oder die Tastenkombi **Strg + Tabulator**, um den Raum in der Raumansicht zu öffnen.

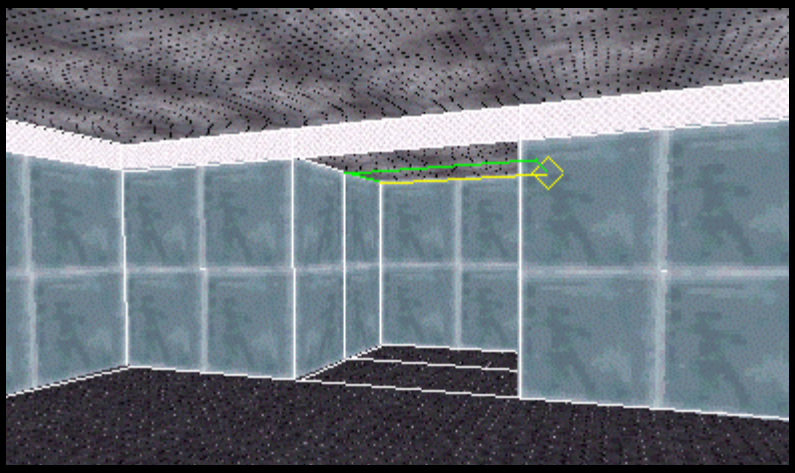


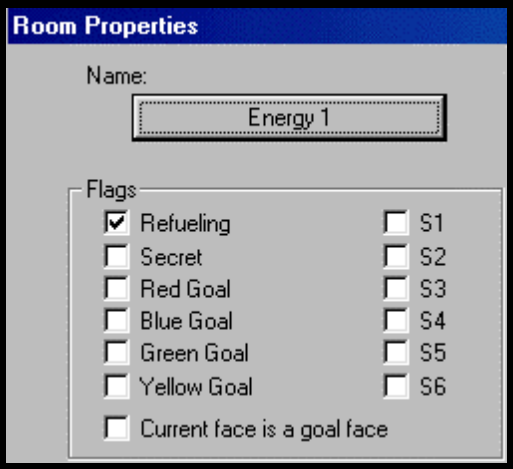
Wir haben hier einen Raum (Bild rechts) mit den Maßeinheiten 20 x 40 x 40. Wir werden den Raum jetzt noch mit der Extrude-Funktion um 20 Einheiten erweitern. Ich habe diese Funktion schon einmal unter 'Den Raum ändern' erklärt. Wir machen es hier nun etwas anders. Beachtet, das das äussere Face das Current Face ist und klickt in der Geometrie Bar auf die Extrude Schaltfläche. Füllt das Dialogfeld entsprechend Eurer Editorversion wie im Bild aus und extrudiert den Raum. Linkes Bild für v39 und älter, rechts für die v40.





Jetzt müssen wir noch die Texturen setzten. Öffnet die Texture-Bar (**Tx - Button**). Wenn Ihr nun unten schaut, wo eigentlich die Texturen in dem Fenster stehen, ist es leer. Klickt auf Level und hier seht Ihr alle Texturen, die in Eurer Level schon eingebaut sind. Gestaltet den Raum nun so, daß er aussieht, wie der erste Raum (bis auf die Lampen an der Deckenkante). Das Resultat seht Ihr im Bild links. Speichert nun den Level in der World View Ansicht!





So, der Raum ist nun fast fertig. Wir wollen dem Raum jetzt noch eine Aufgabe zuordnen, wir könnten ja den Raum als Tunnel zu einem anderen Raum nutzen, doch das kommt im nächsten Thema.

Öffnet das Menü Window.../Room Properties. Ihr müsst Euch in der **Current Room** Ansicht befinden !!! Unter **Name** (Bild unten links) klicken wir auf die Schaltfläche und nennen den Raum **Energy 1**, dann setzt das Flag wie im Bild. Damit haben wir einen Energietanker erstellt. Jetzt berechnen wir noch die Lichter in unserem Level, das können wir so machen, wie es in "**Der erste Level**" beschrieben ist.

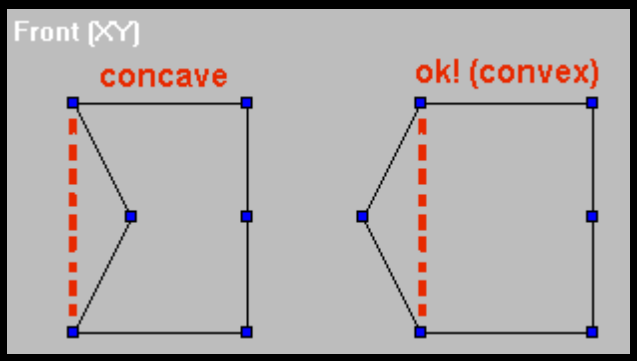
Speichert den Level ab und erstellt eine `mn3`-Datei um Euch das Resultat der Arbeit anzusehen. ☺

In dem nächsten Thema werden wir die Extrude mehrmals verwenden und einen fantastischen Raum erstellen !!!

007 - Reaktor-Raum

Schplurg

Für dieses Thema benötigen wir die Datei "**thelevel_1.zip**" von **Der zweite Raum**. öffnet D3Edit und dann "`thelevel.d3l`". Klickt im Menü auf File/New... und wählt New Room. Wir haben uns in den vorherigen Themen schon mit der Extrude-Funktion beschäftigt. Jetzt wollen wir einen komplizierteren Raum erstellen. Es gibt allerdings Grenzen in D3Edit, ich möchte jetzt mal eine beschreiben.

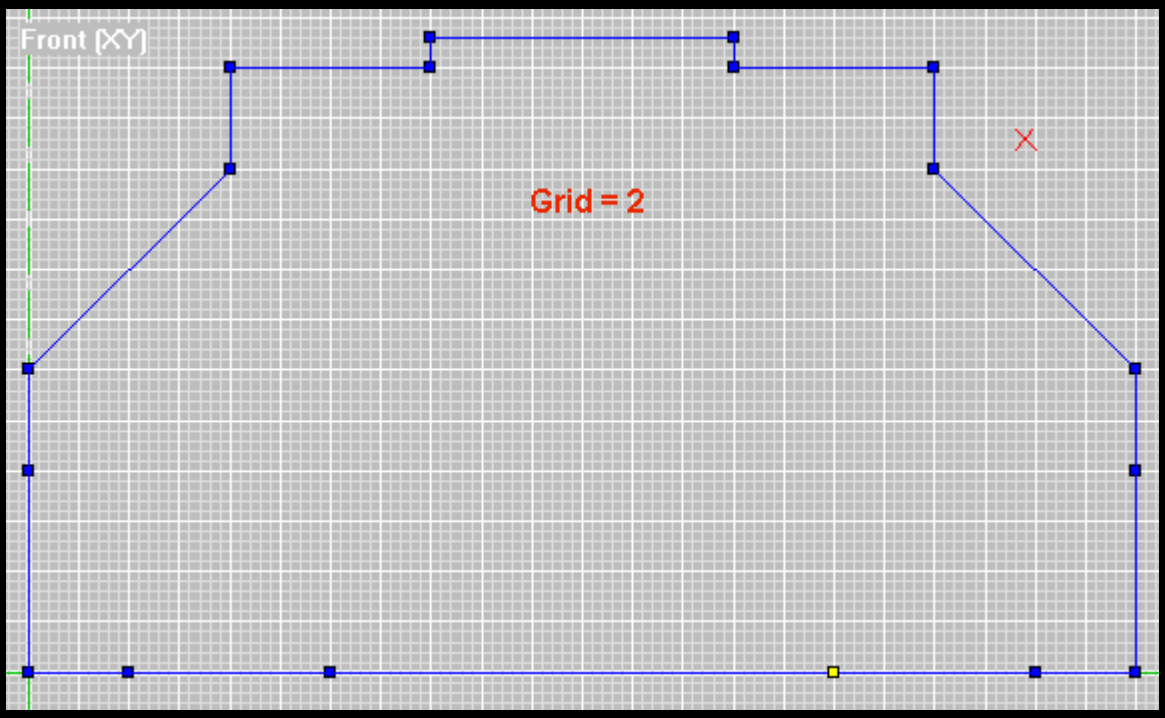


Concave Faces

Ein Face ist Concave wenn ein Vert, das Teil eines Faces ist, in ein Face zeigt. (kleineres Bild, links)

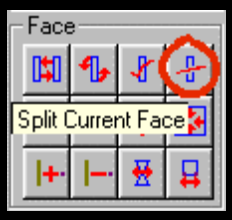
Wir werden in diesem Thema ein Convex Face erstellen und es dann berichtigen. Man muss das Convex Face in zwei Hälften teilen. Erstellt ein Face wie es im größeren Bild zu sehen ist. (siehe: Der erste Raum)

Dieses Face ist am Ende eines langen Raumes, anstatt der Fussboden. Folglich bearbeiten wir das Face in der Frontansicht. Fügt die Verts nacheinander entgegen der Uhr ein, denn so erhalten wir ein Face in der Richtung, wie wir es brauchen. Läßt man nun Verify Room drüber, wird „`Found 1 concave faces`“ angezeigt.

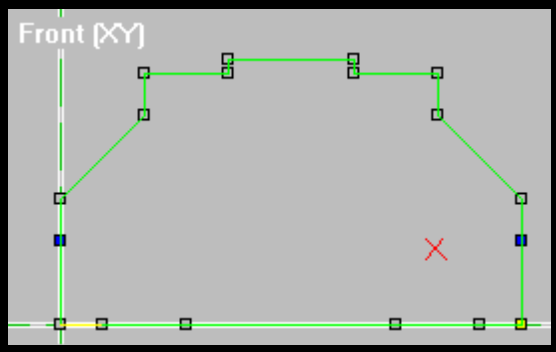


Split Faces

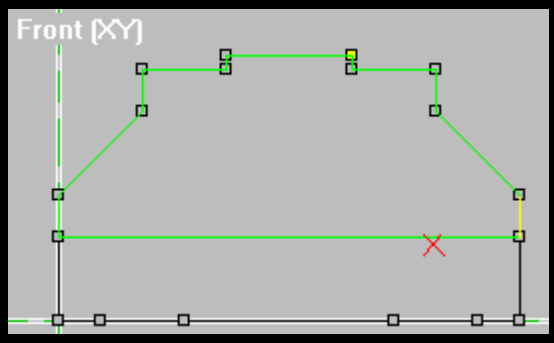
Um das Face fehlerfrei zu erstellen, müssen wir dieses in mehrere kleine Faces teilen. Das machen wir mit der



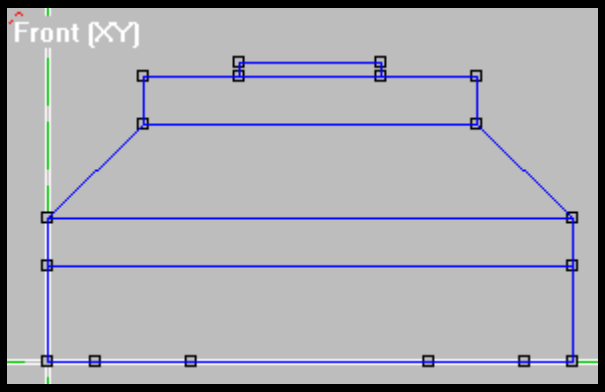
"Split" Funktion (Bild oben) im Geometriemenü (**R**). Markieren wir die beiden Verts. Wir ziehen mit der Maus einen Kasten um die beiden Verts, das Ergebnis sehen wir im kleineren Bild unten. Wie Ihr seht, sind die beiden Verts 40 Einheiten vom Fussboden entfernt, da wird der Raum später zusammengefügt.



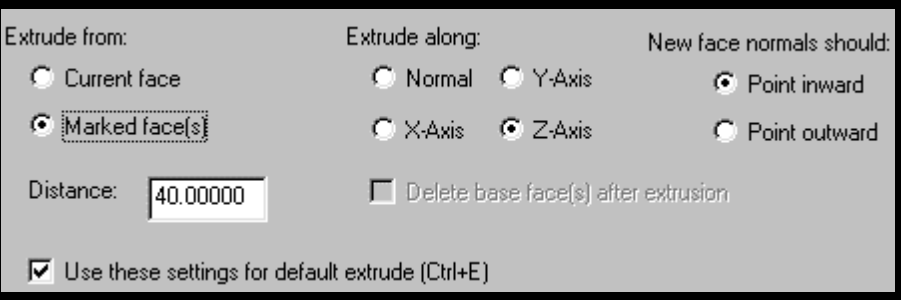
Schaut nach ob das Face aktuell (Current) ist, nun die Schaltfläche **Split Current Face** (die rot umrandete im oberen kleinen Bild). Das Ergebnis ist im nächsten Bild (links) zu sehen. Wenn Ihr jetzt in Menü File.../Verify Room anklickt, seht Ihr das immer noch ein Concave Face da ist, aber wir haben das gemacht, weil ja später der Anschluss an den zweiten Raum platziert werden muss.



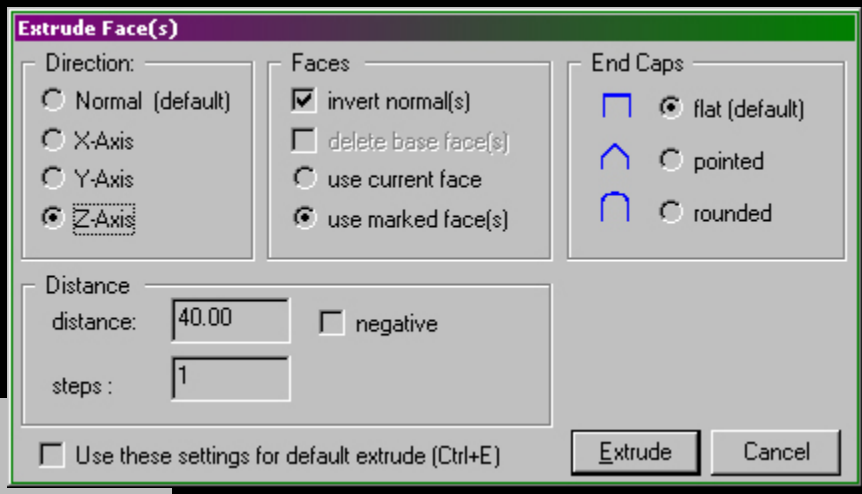
Teilt den Rest des Faces so, das es so wie im Bild rechts aussieht.



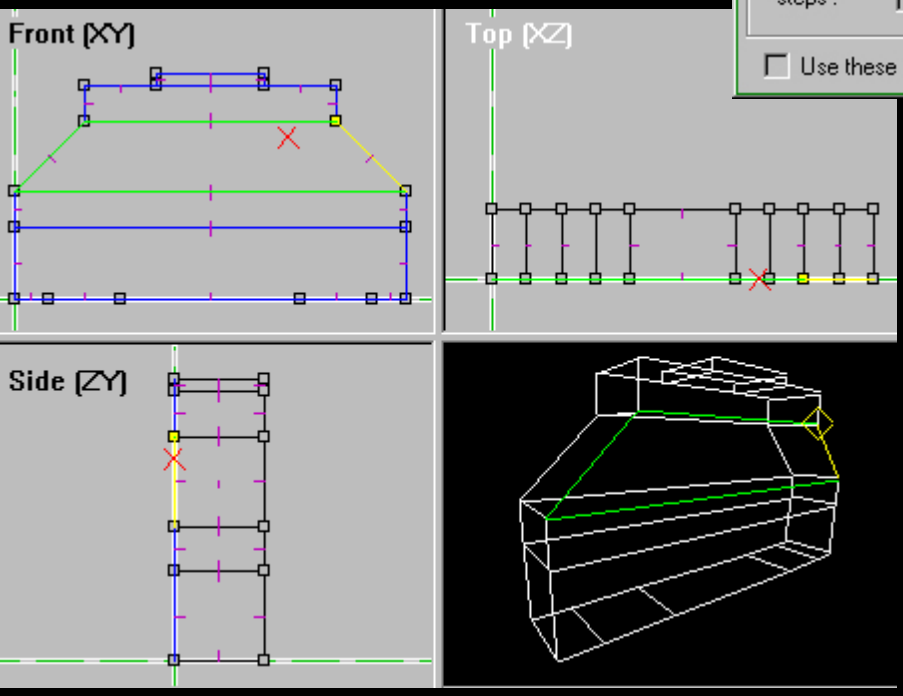
Nachdem das getan ist, schauen wir noch einmal nach, ob wir immer noch ein Concave (File.../Verify Room)haben und Ihr werdet sehen das null Fehler in unseren Raum sind. Die unteren Faces am Fussboden brauchen wir jetzt noch nicht zu teilen.



Jetzt können wir weiter machen und aus dem Face einen Raum. Das machen wir wieder mit der Extrude Funktion. Zunächst müssen wir alle Faces markieren (Taste **M**), jetzt noch ein Face anklicken (wir brauchen ein Current Face) und schaut in **Side** (**ZY**) nach, ob die Normals alle in eine Richtung zeigen. Jetzt öffnet das Extrude Dialogfeld (siehe ‚Der erste Raum') und füllt alles so aus, wie im entsprechenden Bild; oben für v39 und älter, rechts für v40.



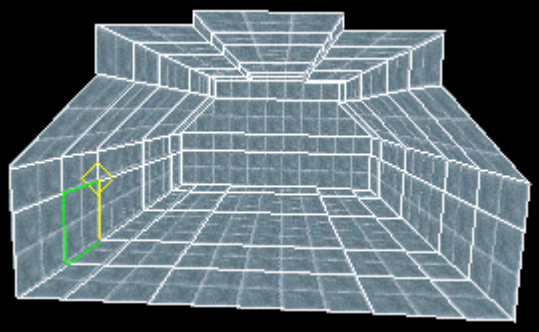
Den Extrude Button noch anklicken und das Resultat sollte wie im Bild unten aussehen.



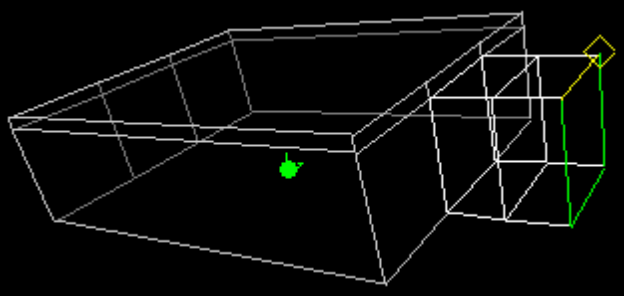
Die Ursprungs Faces sind noch markiert. Die Taste **U** drücken zum demarkieren, jetzt markieren wir das Face daß soeben erstellt worden ist, wir wenden die Extrude Funktion noch öfter an. Nachdem Ihr die Faces markiert habt, klickt auf um die Faces zu drehen weil man sonst nicht den gewünschten Effekt erzielt. Schaut Euch die Normals an, denn die sollten nach aussen zeigen. Wenn man den Raum jetzt betrachtet, fällt auf, daß Faces mitten in Raum sind, diese werden wir zu einen späteren Zeitpunkt löschen.

Wir müssen den Raum noch vier mal extrudieren, und zwar:

1.40 Einheiten in der Z Achse.
2.80 Einheiten in der Z Achse.
3.80 Einheiten in der Z Achse.
4.40 Einheiten in der Z Achse.

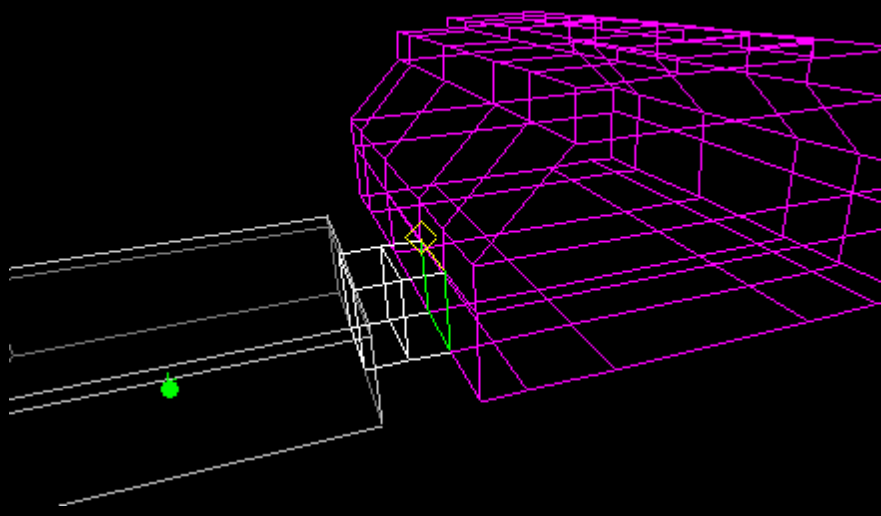


Wir machen das auf dieselbe Weise wie oben beschrieben ist. Nachdem Ihr das gemacht habt, wählt Ihr im Raum Vorschau die überflüssigen Faces aus, markiert diese und löscht sie. Ihr könnt es auch in der 2D Ansicht machen, aber Ihr müsst schauen welche Ihr löscht!! Jetzt müsste Eurer Raum so wie im Bild rechts aussehen. Jetzt müssen wir noch die duplicate verts löschen, dies tun wir mit .
Speichert nun den Raum!



Jetzt müssen wir den Raum noch in unseren Level einfügen. Markiert alle Faces mit der Taste **M** und macht jenes Face, daß im vorigen Bild zu sehen ist, zum Current Face. Wechselt jetzt in die Weltansicht (World View) und macht das Face, wie in Bild links zu sehen ist, zum Current Face. Das ist der Punkt, an dem die Räume zusammengefügt werden.

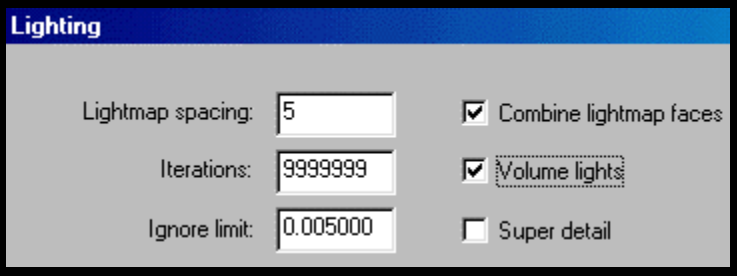
Jetzt gehen wir in das Room Menü und klicken auf Place Room at Current Room. Im Bild rechts seht Ihr das Resultat.



Jetzt noch einmal in das Room Menü und auf Attach Room klicken.

Wir haben die beiden Portals genauso gross gestaltet, damit die Räume Exakt zusammen passen. Man muss die Faces nicht genauso gross machen, aber das Risiko eines Fehlers ist kleiner und wir können den Eingang zuvor schon anders gestalten (Säulen, Balken ect..).

Macht den neuen Raum zum aktuellen Raum und wechselt in Window/Room Properties. Gebt den Raum jetzt einen Namen **Der Barren** und schliesst **Room Properties**. Wenn Ihr jetzt Verify Mine laufen lässt, werdet Ihr sehen, das Ihr keinen Fehler in dem Level habt, bis auf die Boa. Berechnet nun das Licht mit den Einstellungen wie im Bild links.



Speichert nun den Level ab und testet ihn.

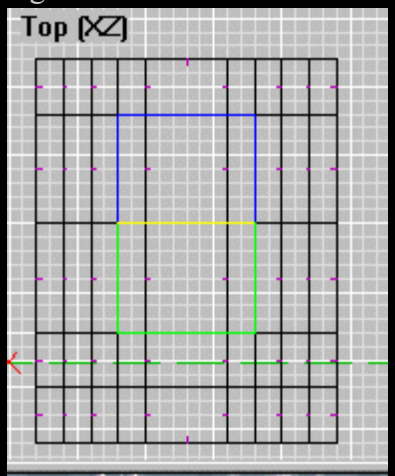
File: 008-thelevel_2.zip

008 - Reaktor

Schplurg

Für dieses Thema benötigen wir die Datei "thelevel_2.zip" von ‚Reaktor-Raum'.

Im vorherigen Thema haben wir unserem Level einen Raum zugefügt. In diesem Thema werden wir einen Reaktor in den Raum stellen, das werden wir mit dem "**Lathe**" Tool erledigen. Desweiteren werden wir einen Player-Startpunkt einfügen.

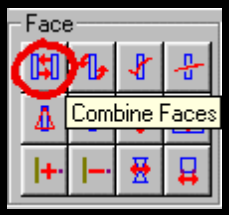


Zunächst benutzen wir den Zoom um den Raum im Ganzen zusehen, Taste S. Wir werden jetzt eine angehobene Plattform bauen, worauf später der Reaktor zu stehen kommen wird.

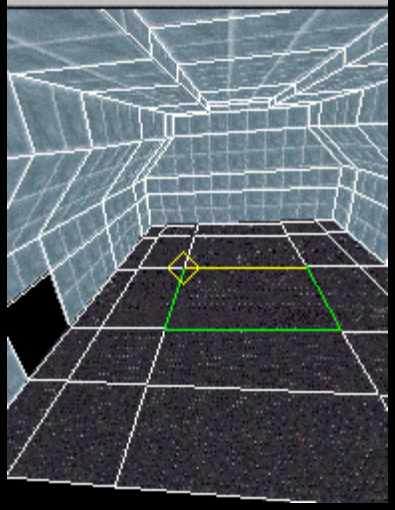
Kombinieren von Faces

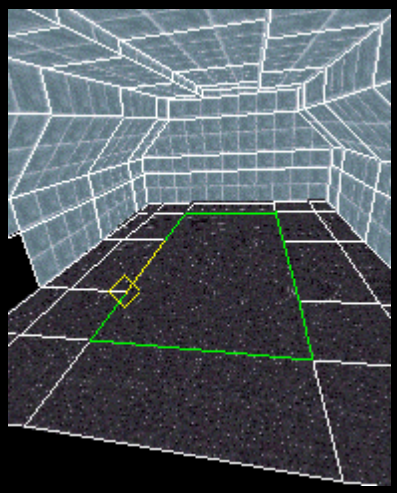
Um zwei Faces zu kombinieren, müssen diese dieselbe Grösse haben, da man keine Faces kombinieren kann, die nach dem kombinieren ein Concave Face ergeben würden. Man bekommt auch einen Hinweis von D3Edit, das dies nicht möglich ist.

Klickt das erste Face an und markiert dieses (Taste Space). Nun klickt das anliegende Face an, so das es das Current Face ist. **Nicht Markieren**! Es sollte aussehen wie im Bild rechts.



Klickt nun auf die **Combine Face**-Schaltfläche (links, rot markiert). Macht nun das nächste Face zum Current Face und klickt noch einmal auf diese Schaltfläche. Das Ergebnis seht Ihr im mittleren Bild links.



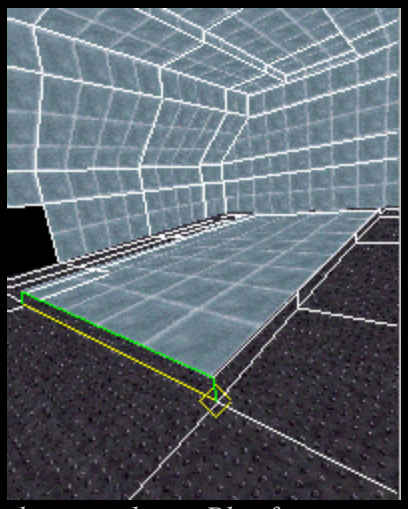


Jetzt werden wir die Extrude Funktion verwenden um die Plattform zu erstellen. Wir extrudieren das Face um 5 Einheiten (Extrude From = Current Face, Extrude along = Normal, New face normals should = Point outward). Setzt noch ein Häkchen in das Feld "Delete base face(s) after extrusion", damit D3 nicht die verdeckten faces mit berechnen muss.
Speichert nun den Level !

Lathe Tool

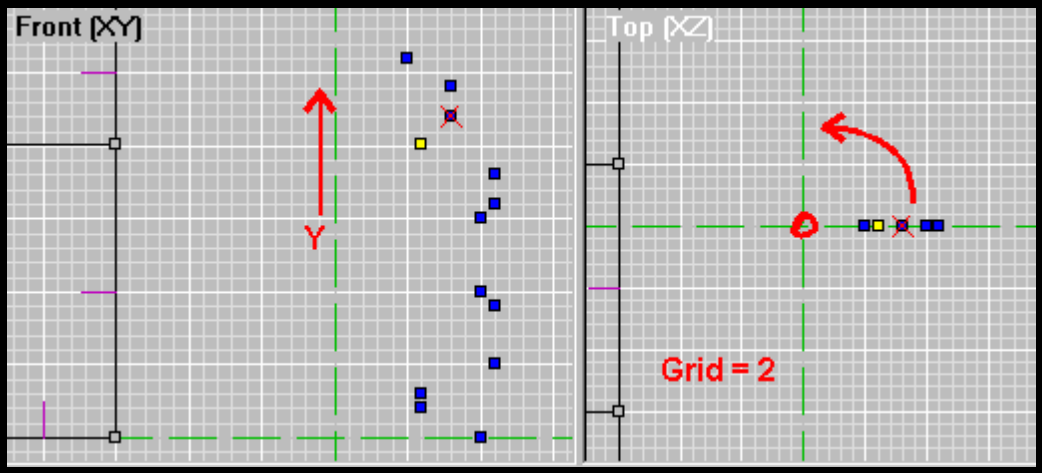
Dieses Tool ist eins der leistungsfähigsten in D3Edit. Das Erstellen von runden oder eckigen Teilen ist damit kein Problem mehr. Man kann das, was man mit diesem Tool lernt, auch in einem ‚reellen' 3D Programm anwenden.

Das Tool macht folgendes, es nimmt einige platzierte Verts und führt diese um eine Mittellinie und erstellt Faces.



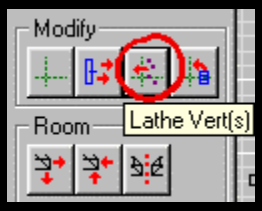
die extrudierte Plattform

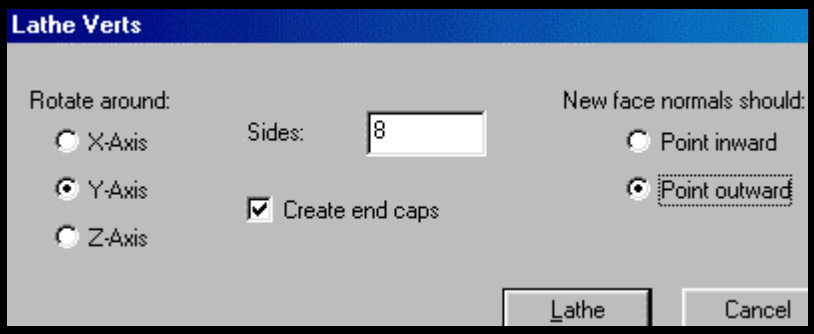
Wechselt nun in den Vert Modus und überprüft das alle Verts unmarkiert sind. Wir sollten unser Bezugsfeld verschieben. Im folgenden Bild ist es außerhalb und auf der rechten Seite unseres Raumes. Es ist besser außerhalb von Räumen Dinge aufzubauen, da es übersichtlicher ist. Wenn wir fertig sind, werden wir es an die richtige Position bringen.



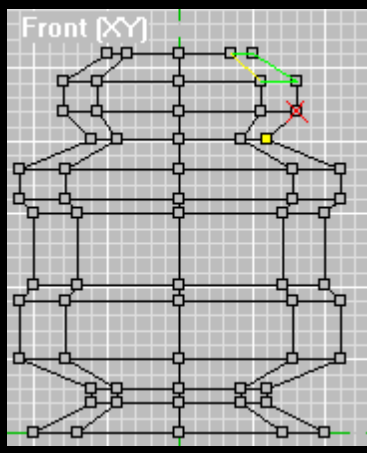
Platziert die Verts wie im Bild links, von oben nach unten.

Öffnet das Lathe Tool, indem Ihr auf den Button vom rechten Bild klickt.





Die meisten Optionen sind selbsterklärend. Create end caps, da wir die Verts nicht direkt auf der y-Mittellinie platzierten. Dieser Platz wird dann durch ein Face umfasst, das end cap genannt wird. Wenn wir die nicht erstellen, haben wir oben und unten ein Loch (Bohrung). Sides bedeutet, daß wenn wir auf den Reaktor schauen, 8 Seiten zu sehen sind.



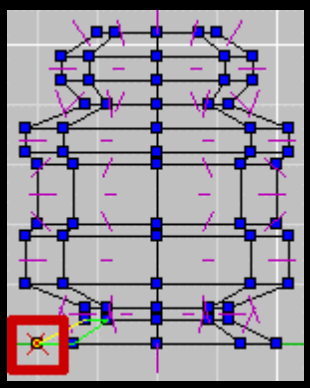
Drückt nun die Lathe Schaltfläche und seht was passiert. (Bild rechts)

Jetzt kommen wir zu den UV's (Texturen Ausrichtung). Das ist eine sehr wichtige Funktion, um die Texturen richtig zu positionieren. Stellt Euch vor, Ihr habt eine Ziegel Texture und mehrere Faces, die diese Textur enthalten und aneinander liegen. Wenn wir diese Texturen nicht ausrichten, würden die Fugen von der Ziegel-Textur nicht übereinstimmen.

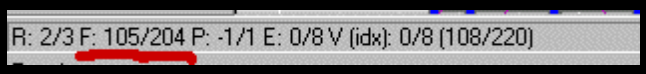
Nun wollen wir beginnen unsere Texturen auszurichten (ist einfach). Texturiert zunächst erst mal den Reaktor. Ihr könnt ihn so texturieren wie es Euch gefällt. Wenn Ihr damit fertig seid, drückt die Taste **M** um alle Texturen in den Raum zu markieren. Nun klickt Ihr in der Texturbar auf **Align...**, ein Dialogfeld wie im Bild links öffnet sich.

Wählt oben die Option **Marked face(s)** und klickt auf die Schaltfläche **Default Uvs**; jetzt sind die Texturen in Eurem Raum "Optimiert".

Verschieben des Reaktors



Zunächst müssen wir alle Faces demarkieren, mit der Taste **U** (Unmark). Jetzt wechseln wir in den Vertex Modus, wieder die Taste **U** drücken. Wir arbeiten weiter in der **Front(XY)** Ansicht. Markiert alle Verts vom Reaktor und macht eines der unteren Verts zu Current (Bild rechts). Einfach das Face anklicken und mit der Taste **V** auf Eurer Tastatur eines der unteren auswählen.

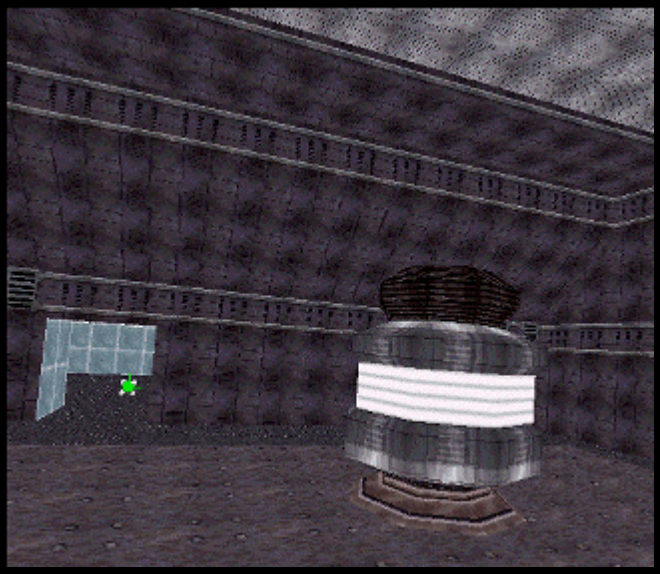


Nun müssen wir noch die Face Nummer unserer Plattform herausfinden, auf der der Reaktor stehen soll. Dazu klickt auf die Plattform und schaut in der Statuszeile nach der Face Nummer. (Bild oben)

Schreibt Euch die Nummer auf oder merkt sie Euch**. Anmerkung**: Ihr werdet noch die Edge Nummer benötigen, doch in diesem Beispiel ist sie 0. Wenn Ihr das nachprüfen wollt, klickt ein Face wie im Bild an und drückt die Taste **E**. In der Statuszeile seht Ihr dann welches Edge aktuell ist. (E: 0/4 bedeutet dann Edge 0 von 4 ist aktuell)

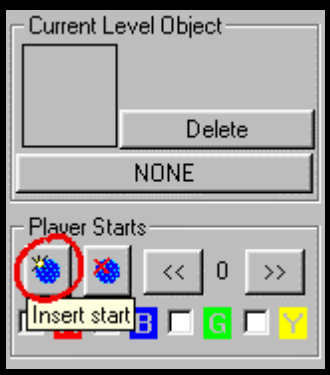
Wir klicken jetzt auf die Schaltfläche wie im Bild links und werden anschliessend nach der Face Nummer gefragt (105 eintragen). Nun werden wir nach der Edge Nummer gefragt und tragen 0 ein. Unser Reaktor ist nun auf der selben Ebene wie unsere Plattform. Stellt nun den Grid auf eins ein, um den Reaktor anschliessend in die Mitte unserer Plattform zu platzieren. Bewegt den Reaktor nun mit den Tasten **4** und **6** auf Eurem **NumBlock**, mit Hilfe des Normals unten, könnt Ihr ihn in die Mitte stellen.

Einsetzen eines Player Startpunktes



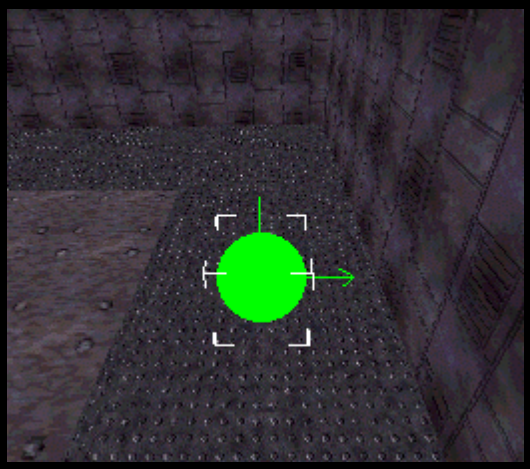
Seht Ihr den kleinen grünen Punkt im Bild rechts, das ist ein Player Startpunkt. Wir werden nun einen zweiten in unserem Reaktorraum platzieren. Öffnet die Objekt Bar mit der Schaltfläche in der Menüleiste.

Positioniert das **X** mit einem Klick Eurer Maus oder macht es mit **Strg + Cursortasten**. Ihr müsst in allen drei Ansichten nach schauen, ob das **X** auch im Raum liegt. Drückt nun die Schaltfläche die im Bild links rot umrandet ist. Nun habt Ihr zwei Player Startpunkte im Level.



An dem grünen Pfeil könnt Ihr sehen, in welcher Richtung der Player steht. Ich muss meinen noch drehen, dazu muss der Player Startpunkt aktuell sein. Das sieht man an den weissen Linien, die das Objekt umgeben. Verschieben kann man diesen Punkt mit den Tasten **8**, **4**, **6** und **2**. Drehen kann man das Objekt mit den Tasten **7** und **9**, auf dem Numern-Block.

Jetzt muß noch das Licht berechnet werden. Stellt folgendes ein:
Lightmap spacing = 4,
Iterations = 99999999 ,
Ignore limit = 0.000500
und setzt das Häkchen bei
Volume lighting. Wenn Ihr das erledigt habt, speichert den Level und testet ihn in Descent.

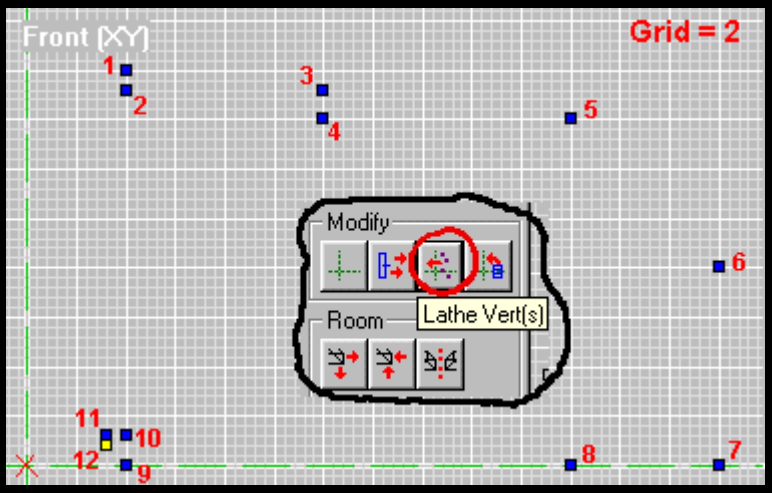


009 - Lathe einen Raum

Schplurg

Für dieses Thema benötigen wir die Datei "`thelevel_3.zip`" von ‚Reaktor'.
Öffnet D3Edit und die Datei "`thelevel.d3l`".

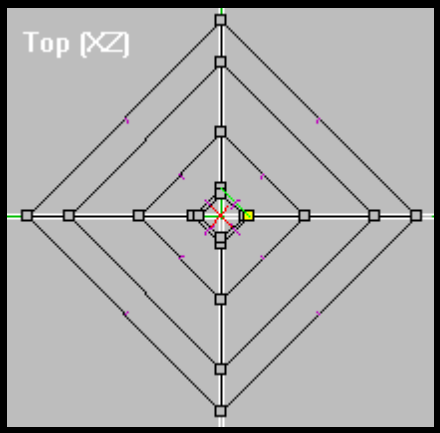
Wir hatten schon im vorherigen Thema mit dem Lathe Tool gearbeitet. Wir haben einen Reaktor in einem Raum erstellt. Stellt Euch vor, der Reaktor wäre ein riesiger Raum und die Faces würden nach innen zeigen, dann könnte man in ihn fliegen. Wir werden nun einen Raum mit dem Lathe Tool erstellen.



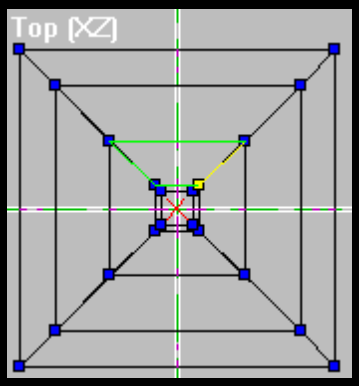
Nun erstellen wir mit dem Tool einen Raum mit 4 Seiten. Öffnet einen neuen Raum und setzt die Verts wie im Bild links.

Beachtet den Abstand zwischen der vertikalen Bezugslinie und den nächsten Verts. Dieser Platz wird mit den **end caps** gefüllt.

Speichert den Raum.



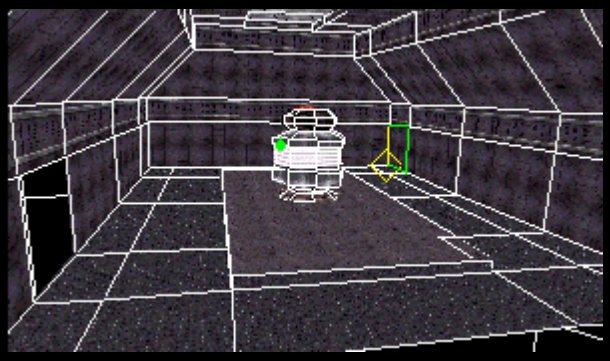
Öffnet das Lathe Dialogfeld. Nehmt folgende Einstellungen vor: **Rotate around** = Y, **Sides** = 4, **Create end caps** ein Häkchen setzten, **Normals** = Point inward. Klickt auf die **Lathe** Schaltfläche.
Das Resultat seht Ihr im Bild rechts. Wechselt nun in die Ansicht Top (XZ) und markiert alle Verts mit der Taste **M**. Wir müssen den Raum jetzt noch drehen. Drückt die Tasten **1** und **3** auf Eurem Num-Block und dreht ihn soweit, bis es so wie im Bild links aussieht.



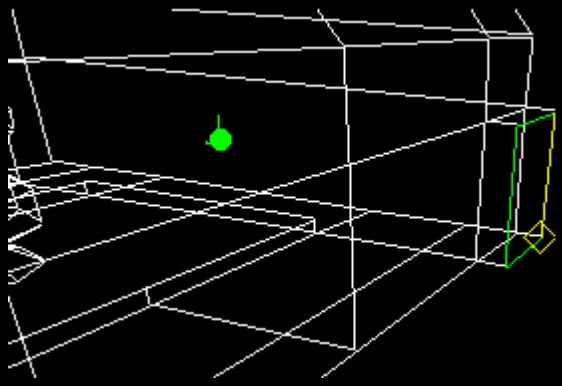
Als nächstes müssen wir den Raum noch am Grid ausrichten. Wenn man genau hinschaut, sieht man, das die Verts nicht auf den Grid liegen.
Anmerkung: Als ich das gemacht habe, musste ich den Grid auf 2 einstellen, ansonsten hatte ich dann immer non-planar-faces!!!

Stellt also den Grid auf 2 und macht ein Vert zum aktuellen (gelb). Nun klickt die Schaltfläche im Bild rechts, die rot umrandet ist. Jetzt wird das aktuelle Vert zum Raster gezogen und alle Markierten auch. Wenn Ihr jetzt im Menü File.../Verify Room laufen lasst, könnt Ihr sehen, das unser Raum keinen Fehler aufweisst. Speichert den Raum. Jetzt können wir den Raum texturieren.

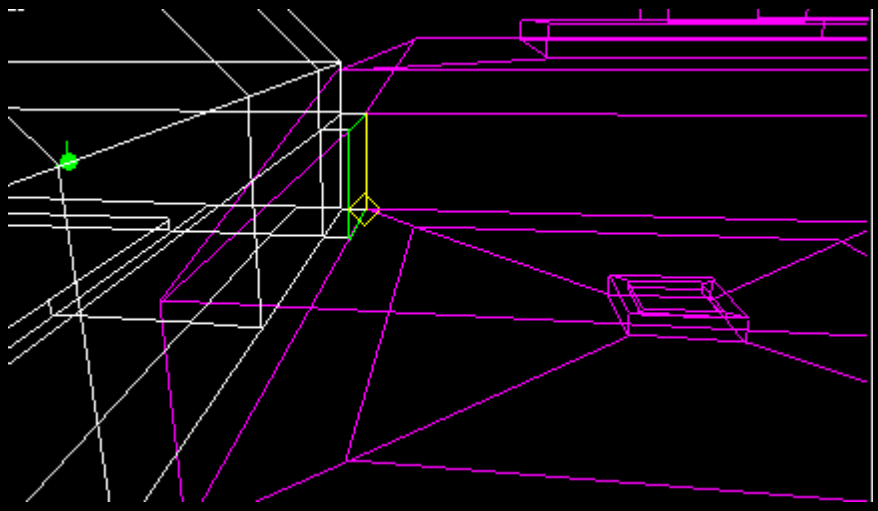
Hinzufügen des Raumes zu unserem Level



Wechselt in die World View Ansicht und klickt auf den Reaktor-Raum. Wechselt jetzt in die Room View Ansicht. Wir müssen das Portal zu den neuen Raum vorbereiten. Klickt nun das Face an, das im Bild rechts zusehen ist und extrudiert dieses entlang der X-Achse, um 10 Einheiten, Normals Point Outward (Damit der Raum nicht direkt an unseren Reaktor Raum angebunden wird). Nachdem das erledigt ist, texturiert die neuen Faces.



Wir machen nun das Face, das im Bild rechts zu sehen ist, zum aktuellen Face. Nun wechseln wir wieder zu unseren Raum, den wir anbinden wollen und markieren in Face Modus alle Faces (Taste **M**). Jetzt müssen wir noch eine der vier Seiten des Raumes zum aktuellen Face machen. Wir wechseln wieder in die World View Ansicht und klicken in den Menü Room auf Place Room at Current Room. Jetzt ist unser Raum, den wir anbinden wollen, lila markiert und ist in der Mitte von unseren neuen Raum. Geht jetzt wieder in das Room Menü und wählt Snap Placed Room. Jetzt wird der Raum anders ausgerichtet, wir können das beeinflussen, indem wir die Taste **V** auf unsere Tastatur drücken, damit verändert man das aktuelle Vert, das im Bild mit einen gelben Viereck gekennzeichnet ist.

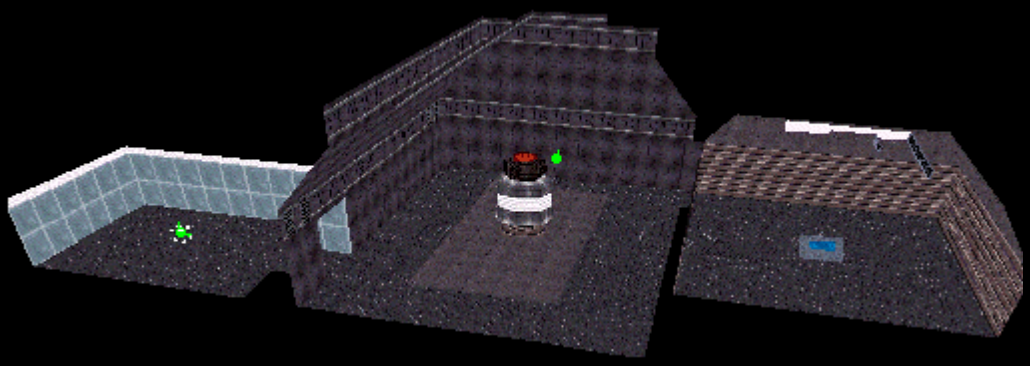


Probiert es mal ein wenig, um zu sehen wie sich der Raum anbinden lässt und wie man ihn ausrichten kann. Snapt ihn solange, bis es aussieht wie im Bild links. Wenn das der Fall ist, dann wieder in das Room Menü und auf Attach Room klicken, jetzt ist der Raum an unser Level angefügt.

Speichert nun den Level.

Berechnet jetzt noch die BOA (Lichter), wie das geht müsstet Ihr doch schon wissen. Nun könnt Ihr das Level testen.

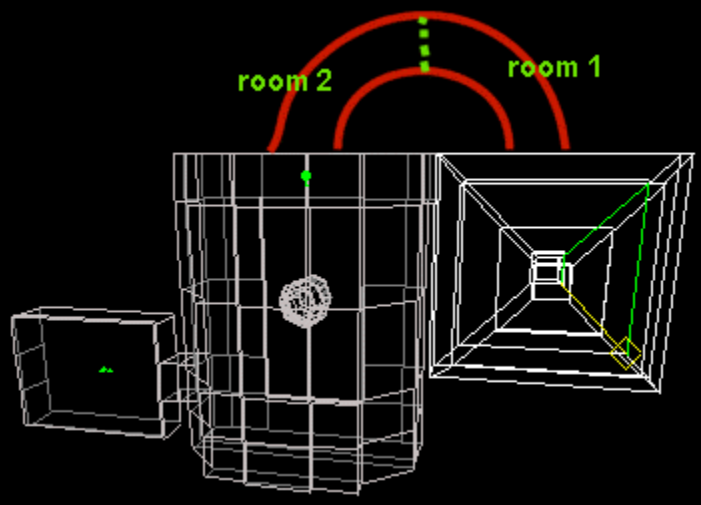
Hier ein Bild, das zeigt wie es bis jetzt in etwa aussehen müßte.



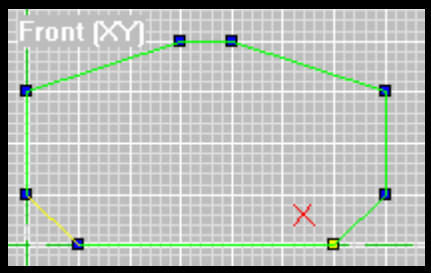
010 - Das Bend Tool

Schplurg

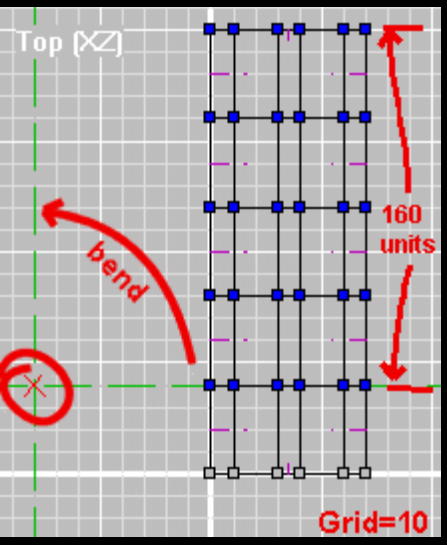
Für dieses Thema benötigen wir die Datei "`thelevel_4.zip`" von ‚Lathe einen Raum'. Öffnet D3Edit und die Datei "`thelevel_4.d3l`".



Wir werden in diesem Thema einen Tunnel erstellen, der in etwa so wie im Bild links aussieht. Genauer gesagt sind es zwei Tunnels, je 90°, einer rechts gebogen, einer links. Es gibt auch andere Möglichkeiten Tunnels zu erstellen, doch dazu wird es noch etwas in der "Tipps und Tricks" Sektion geben.



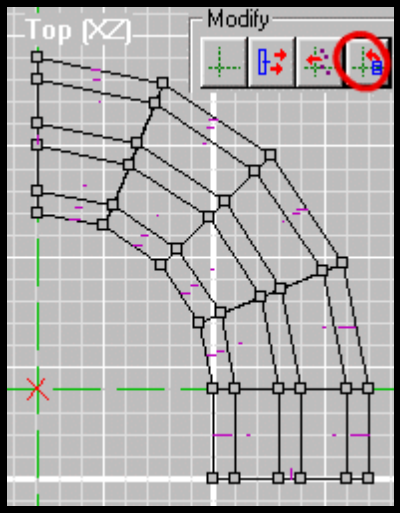
Erstellt einen neuen Raum und in diesem, in der Front (XY) Ansicht, ein Face. Ihr könnt es gestalten wie es Euch gefällt, achtet nur darauf, das es nicht konkave ist und die Höhe des Faces nicht 40 Einheiten überschreitet (wegen des Anschlusses an die bestehenden Räume). Mein Face seht Ihr im Bild rechts.

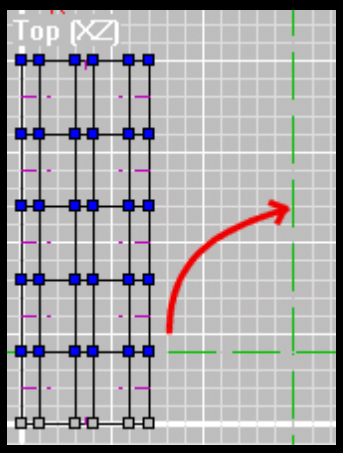


Erstellt nun 5 Segmente zu je 40 Einheiten mit der Extrude Funktion. (Z-Achse) Wenn Ihr das erledigt habt, müsste Euer Tunnel wie im Bild links aussehen.

In diesem Bild seht Ihr noch weitere Informationen. Das rote x, das eingekreist ist, ist der Punkt, um den der Tunnel "gebogen" wird. Es werden nur die Verts "gebogen" die auch markiert sind. Die horizontale Linie (grün) muss genau auf der unteren markierten Reihe Verts liegen. Das ist der Punkt, an dem die Drehung anfängt. Um die grüne Linie zu bewegen, drückt die **Strg** - Taste und klickt mit der linken Maustaste. Stellt es so ein, wie im Bild links; 80 Einheiten Abstand. Wenn Ihr das erledigt habt, speichert den Raum unter `base_tunnel.orf`, wir werden diesen noch einmal benötigen.
Klickt nun auf die **Bend Verts** Schaltfläche. Im folgenden Dialogfeld klickt Ihr **Y - Achse** an. In die Eingabebox **Distance** gebt Ihr 160 ein, die 160 errechnet man wie folgt: Wir haben 4 Segmente, ein Segment ist 40 Einheiten lang, 4*40 = 160.

In die Eingabebox **Angle** gebt Ihr 90 ein, weil wir ja eine 90° Drehung wünschen. Wenn das erledigt ist, drückt den **Bend** - Button. Euer Tunnel sollte nun in etwa wie im Bild rechts aussehen. Speichert den Raum wiederum, diesmal unter den Namen `tunnel_rechts.orf` .

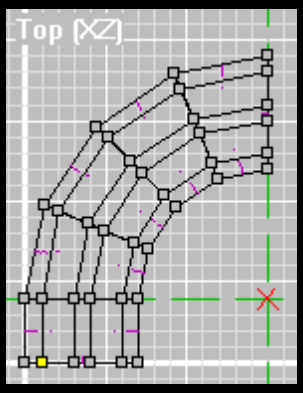




updaten

Öffnet nun `base_tunnel.orf`, auf diese Weise brauchen wir nicht einen neuen Raum zu erstellen. Wir werden nun die linke Hälfte von unserem 180° Tunnel erstellen.

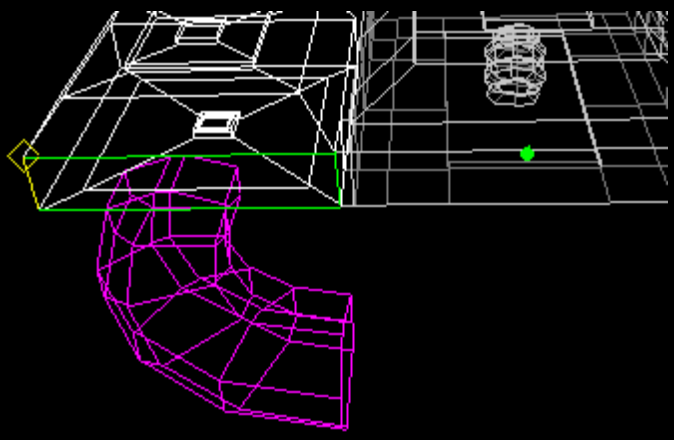
Setzt die grünen Hilfslinien wie im Bild links (80 Einheiten nach links). Klickt nun wider auf die **Bend Verts** Schaltfläche und klickt dann auf **Y- Achse**. Im Feld **Distance** gebt Ihr **-160 (Negativ 160 !!!)** ein, Ihr könnt es ja auch einmal anders versuchen, aber Ihr werdet dann alles

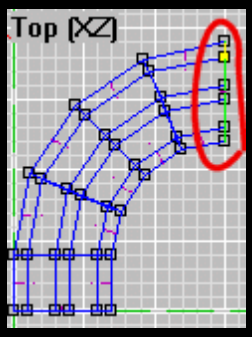


mögliche erstellen, nur nicht das was Ihr wollt. Ich kann Euch dieses Phänomen auch nicht erklären. Ok, bei **Angle** gebt Ihr wieder 90 ein. Drückt die **Bend**-Taste wieder, das Resultat seht Ihr im Bild rechts.

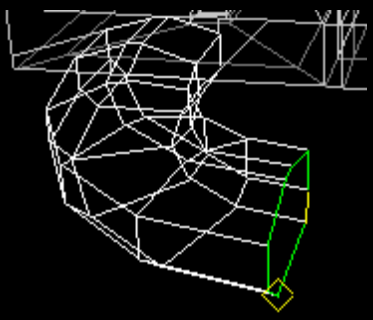
Speichert den Raum unter `tunnel_links.orf`.

Jetzt müssen wir noch unsere Tunnel in den Level einfügen, dazu öffnet den Raum `tunnel_rechts.orf`. Wechselt in den Face Modus, markiert alle Faces und macht das allererste Face zum aktuellen. Geht in die World View Ansicht und macht das Face, wo Ihr den Tunnel anfügen wollt, zum aktuellen. (Bild links)

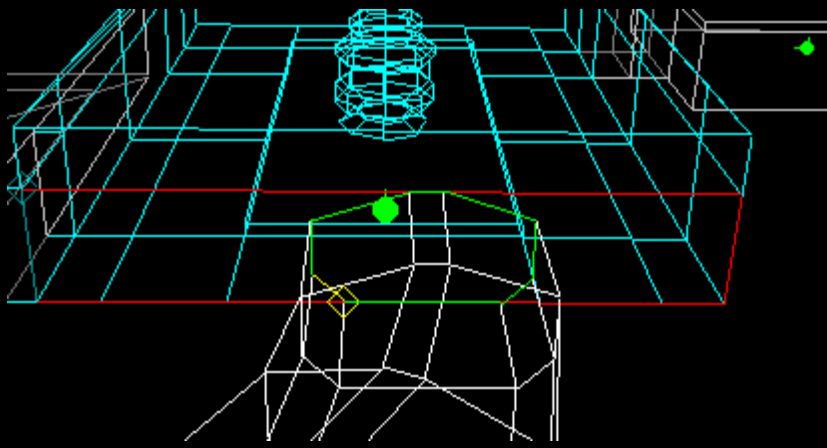




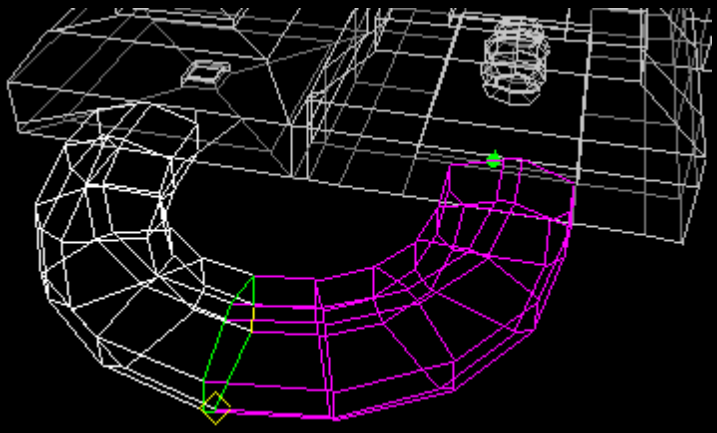
Wechselt nun in das Menü Room/Place Room at Current Room. Ok, unser Tunnel ist richtig platziert, wählt jetzt Room/Attach Room. Nun ist der erste Teil des Tunnels in dem Level. Öffnet jetzt den Raum Tunnel_links.orf, markiert alle Faces und macht das Face, das im Bild rechts rot markiert ist, zum aktuellen. Wechselt jetzt wieder in die World View Ansicht und macht dort das Face, das im Bild unten zu sehen ist, zum aktuellen. Jetzt wählt im Room Menü Place Room at Current Room und dann Room/Attach Room.



Jetzt müssen wir noch ein Portal erstellen, das den Tunnel mit dem Reaktor-Raum verbindet. Hierzu markiert in der World View Ansicht, das Face vom Reaktor-Raum. Macht das entsprechende Face zunächst zum aktuellen, dann markiert es mit der Taste **M**. Macht nun das Face des Tunnels zum aktuellen Face. Das Face vom Reaktor-Raum, das markiert ist wird rot umrandet dargestellt (Bild rechts).



Geht jetzt ins Room-Menü und wählt Join Rooms, Ihr werdet nun gefragt ob der Raum in die Mitte des Faces soll, klickt auf **NO** oder **Nein**, da unsere Faces bündig miteinander sind.



Perfekt!!! Der Tunnel ist jetzt komplett. Texturiert ihn noch mit einigen Lichtern, berechnet diese und testet den Level.

Der Walkthrough von Robo behandelt grundsätzlich in etwa dasselbe wie Schplurg in seinen ersten fünf Tuts, dafür aber etwas eingehender.

Wem es bei Schplurg's The Level anfangs zu schnell ging, ist hier richtig.

Das Original ist in Englisch und unter folgender Adresse erreichbar:

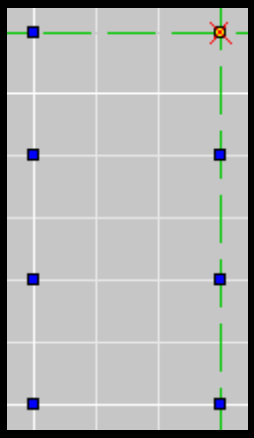
http://www.planetdescent.com/d3edit/

011 - Basics

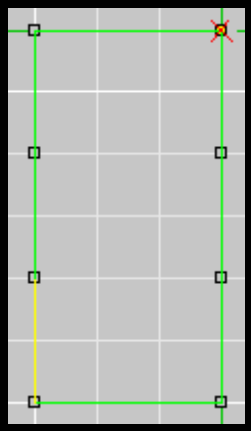
Robo

Bevor Du beginnst in etwas gut zu werden, mußt Du die Grundlagen kennen. Zum Beispiel, bevor Du flüssig Englisch sprichst, beginnst Du mit einfachen Worten. Dasselbe gilt fürs Levelbauen. Du mußt die Grundlagen kennen um erfolgreich zu sein. Definitionen sind ein guter Startpunkt, da Du anfangen kannst Dir Begriffe anzulernen.

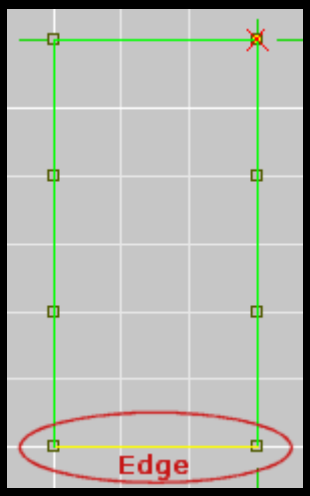
Vertex:
Ein unsichtbarer, null-dimensionaler Punkt der Kanten verbindet um daraus ein Face zu formen. In D3Edit werden sie als kleine gefärbte Quadrate angezeigt, die sind grau wenn unmarkiert, blau wenn markiert und gelb wenn current (diese Begriffe werden noch erklärt). Auch kurz **Vert** genannt.



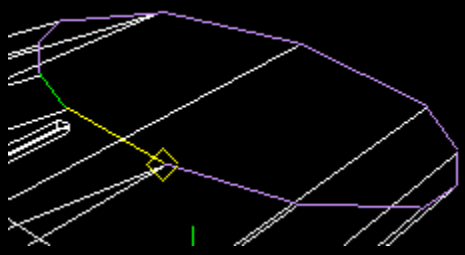
Face:
Ein Polygon mit einem Minimum von drei Vertices. Sie sind durch schwarze Linien dargestellt wenn unmarkiert, blau wenn markiert und grün wenn current, ähnlich wie die Vertices. Levels sind einfache Polygone, arrangiert um Wände, Boden, Decken, Röhren, Reaktoren - alles was Du siehst - zu formen.



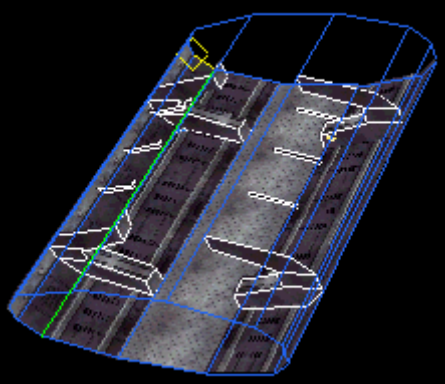
Edge (Kante):
Ein Edge ist eine Linie, die zwei Verts in einem Face verbindet. Sie sind immer gelb, aber Du siehst sie nur, wenn Du ein current Face hast. Einfach zu verstehen, wenn Du's mal ausprobierst.



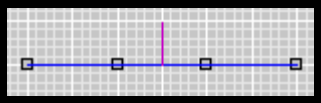
Portal:
Nicht das was Du denkst. Ein Portal ist ein Paar von Faces, welche zwei Räume verbinden. Sie sind ingame unsichtbar, außer Du machst sie zu Kraftfeldern oder Ähnlichem. Sie können auch verwendet werden um Ereignisse auszulösen. In D3Edit sind sie purpurn, wenn sie ausgewählt sind.



Shell:
Diese sind die äußeren Faces in jedem Raum, welche das innere Deines Levels von dem dunklen Abgrund separieren, welchen wir gern das ,' nennen ☺ Im Bild rechts habe ich die Shell-Faces markiert, sodaß Du besser sehen kannst daß es die äußersten Faces sind.



Normal:
Ah, das ist ganz schwer zu verstehen (und zu erklären). Eine Normal ist ein kleiner purpurner Strich bzw. Vektor, welcher in jener Richtung aus dem Face guckt in welche das Face zeigt. Das ist aus mehreren Gründen wichtig zu wissen; das das Face nur aus einer Richtung sichtbar ist. Eine Seite ist mit was auch immer Du willst texturiert, Metall, Wasser, Holz oder mit dem Gesicht deines nächsten Opfers, die andere Seite ist unsichtbar. Du kannst tatsächlich durchfliegen! ☺



Denk Dir eine Uhr: gehen die Zeiger im Uhrzeigersinn, dann schaut das Ziffernblatt = das Face zu Dir, die Uhr ist ‚sichtbar'. Gehen die Zeiger gegen den Uhrzeigersinn siehst Du Die Uhr halt von hinten und siehst nix.

Gut, das sind die Hauptbegriffe, nun kommen wir zum Interface. DieserTeil könnte etwas verwirend werden ☺

Panels

In D3Edit gibt es gegenwärtig elf Paneele, jedes mit seinem Verwendungszweck. Sie werden durch klicken auf die entsprechenden Buttons aufgerufen. Hier ist eine Liste aller Buttons, ihren assoziierten Paneels und einer kurzen Beschreibung:

C	Kamerakontrolle	Damit manipuliert man die Kamera-Ansichten.
R	Geometrie	Zur Behandlung von Polygonen, Verts und Edges.
Tr	Terrain	Damit wird der Außenbereich bearbeitet.
Tx	Texturen	Managt die Texturen
P	Pfade	Fügt Pfade für Robots und Kameras ein.
O	Objekte	Organisiert die Objekte.
D	Türen	Fügt Türen zwischen Räume ein.
S	Sound	Zur Behandlung der Soundquellen in Deinen Levels.

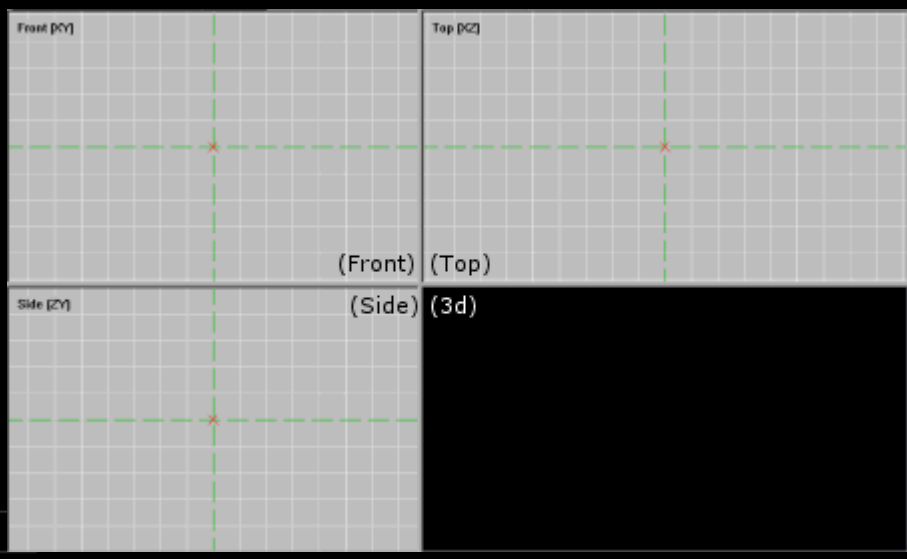
Seit D3Edit-Version Dev8 gibt es zusätzlich noch:

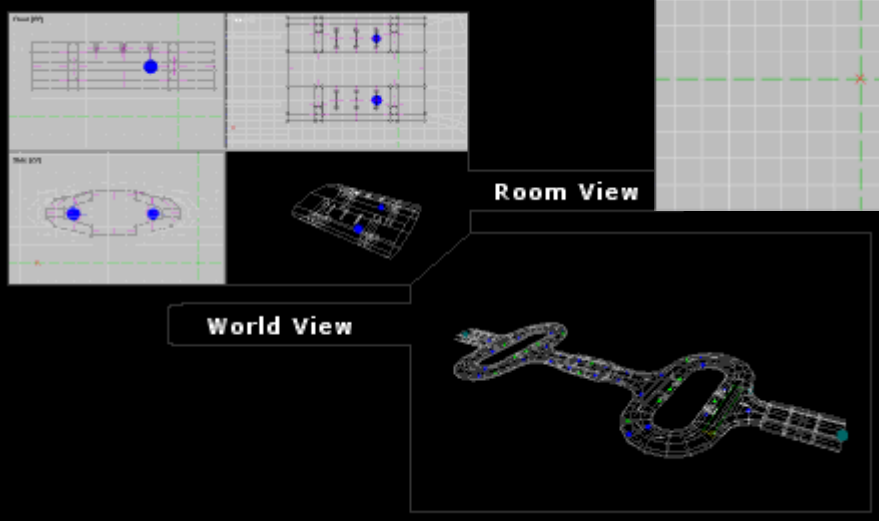
M	Nachrichten	Hier kannst Du Messages von D3Edit einsehen, ohne Notepad öffnen zu müssen.
Q	Quicktest	Zum schnellen Testen Deines Levels, ein prima Tool.
K	Markierung	Erlaubt es Faces nach bestimmten Kriterien auszuwählen.

Es könnte hilfreich sein sich das auszudrucken und irgenwohin zu pinnen, um bequem nachspicken zu können. Nach einer Weile kann man's dann auswendig ☺

Ansichten (Viewports)

Es gibt vier Ansichten: Top (Oben, X-Z), Side (Seite, Z-Y), Front (Vorne, X-Y) und die 3d-Ansicht. Sie zeigen Deine Räume aus unterschiedlichen Perspektiven. (Bild rechts)

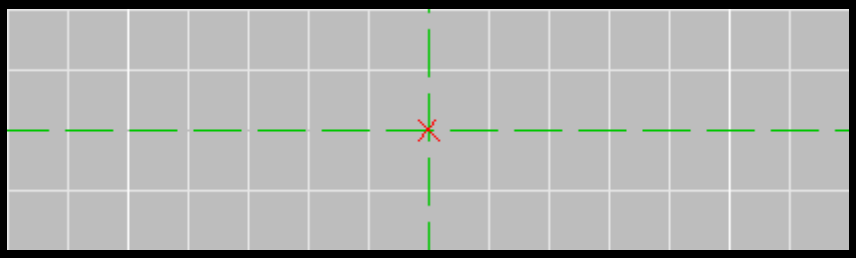




Current Room View / World View:

Hier links siehst Du die zwei Ansichten in D3Edit, jeweils betitelt. Die obere Ansicht ist ist die **Current Room View**, die Untere die **World View**. Wenn Du das ganze Level betrachten willst, gehst Du in die World View, aber wenn du einen Raum betrachten willst der Teil Deines Levels ist gehst Du in die **Current** (selektierter) **Room View**. Das ist ein bischen konfus, ich weiß. Man kann nur ein Level geöffnet haben, einen Raum im Level ausgewählt und 50 Räume, welche separat von Deinem Level sind. Die separaten Räume haben ihre eigenen Room Views.

Das Grid und die Achsen:
Um Dich dabei zu unterstützen alles exakt zu positionieren, hat D3Edit das Grid (Gitter). Es ist in den 2d-Ansichten (Front, Top, Side) zu finden, seine Größe kann eingestellt werden und es kann an- oder abgeschaltet werden. Nicht schwer zu verstehen, ganz anders als andere D3Edit-Features.



Weil Du ein 3d-Spiel zockst, mußt du auch alles **in 3d machen**. Um besser mit dem Grid umzugehen, mußt Du die Achsen verstehen:

- **X** – Links/Rechts
- **Y** – Rauf/Runter
- **Z** – Vor/Zurück

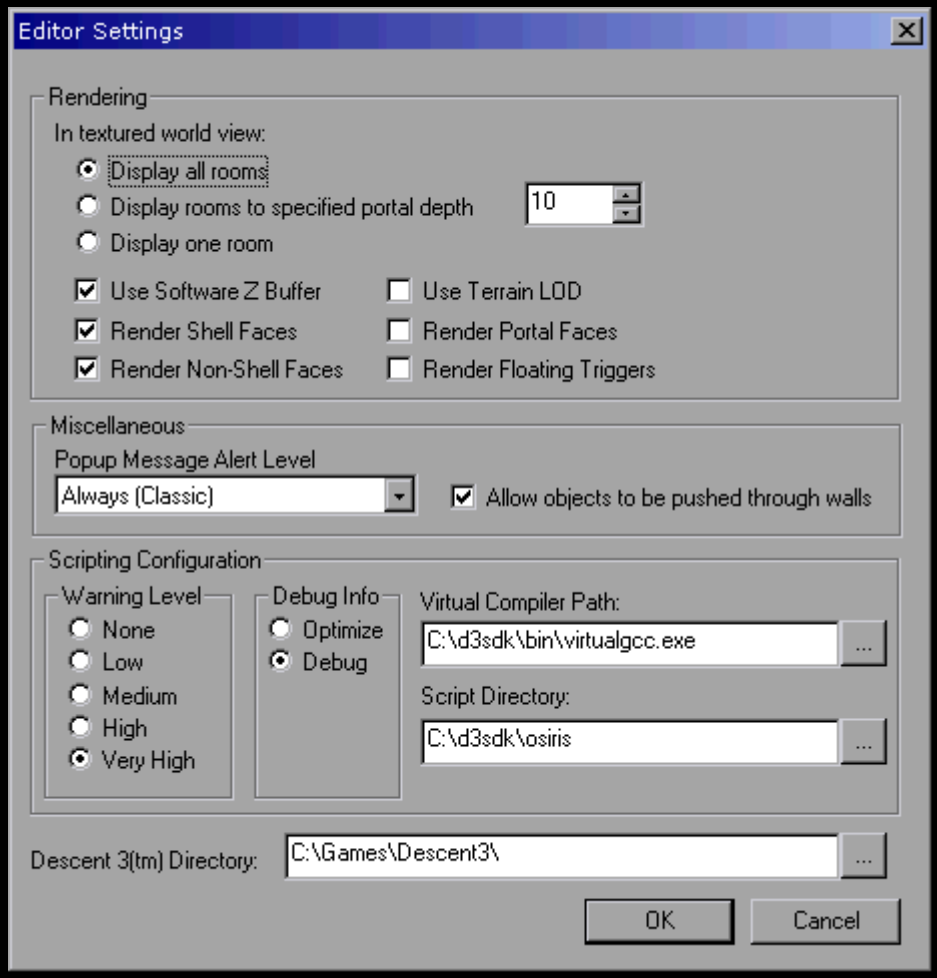
Das heißt also, die Richtungen in X-Y werden in der Front-Ansicht gezeigt, die Richtungen in X-Z in der Top-Ansicht und die in Z-Y in der Side-Ansicht. Du wirst das schnallen wenn Du ein wenig herumexperimentierst.

Das ist mal das Grundsätzliche Know-How.
Und weiter geht's...

012 - Ein einfacher Raum

Robo

Wenn Du D3Edit downgeloadet und entzippt hast, siehst Du beim ersten Programmstart einen Einstellungsdialog wie diesen:



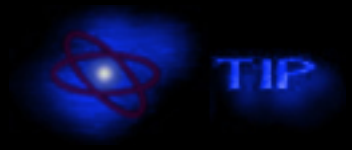
Das ist **mein** Einstellungs-Fenster. Ich schlage vor Du kopierst die Einstellungen, aber du mußt auf das Descent 3-Verzeichnis zeigen, wenns in einem anderen als hier ist. Nachdem das erledigt ist, startet das D3Edit normal.

Schalte alle Paneele aus (unter Verwendung der Buttons in der Toolbar), und gehe nach File...New und wähle New Room. Jetzt kommen wir zum Arbeiten. Faces zu erzeugen ist einfach wie Punkte verbinden, wobei die Punkte Verts sind. Vesuchen wir's mal . Stelle sicher daß Du im Vertex-Modus bist indem Du in die Statuszeile guckst:

Sie zeigt Dir die Position Deines Cursors, Deine eingestellte Gridgröße, in welchem Modus Du Dich befindest und wie viele Faces oder Verts markiert sind.

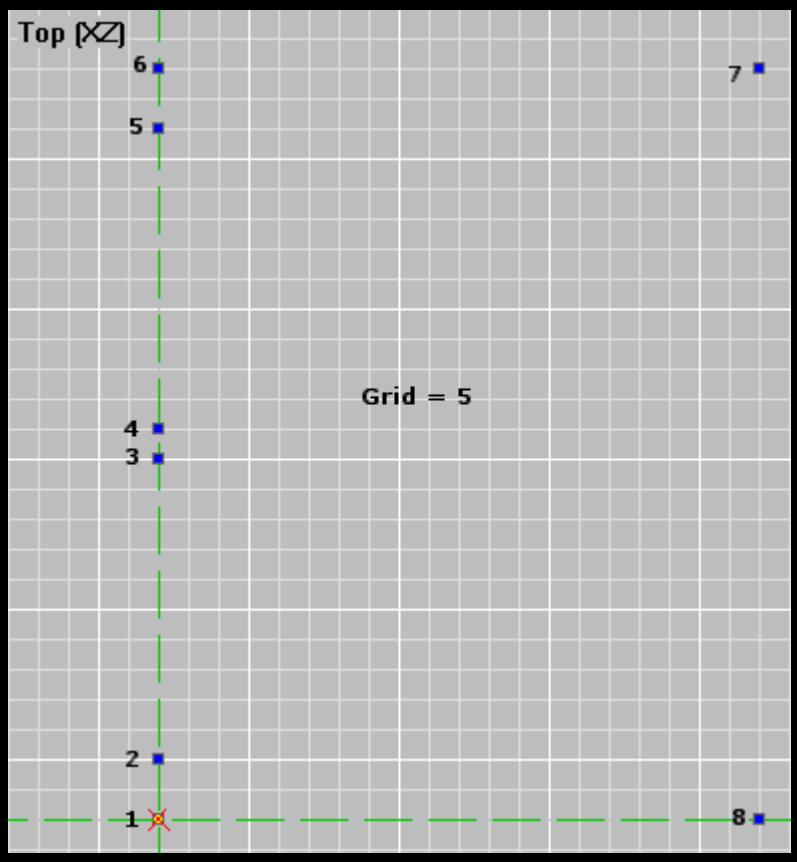
Dann klicke zweimal in die Top-Ansicht um sie aktiv zu machen und das rote X angezeigt zu kriegen, und drücke **Ins** bzw. **Einf**. Was Du jetzt sehen solltest, ist ein kleines blaues Quadrat, das ist ein Vertex. Um einen oder eine Gruppe von Vertices zu löschen drücke **U** um erstmal alles abzumarkieren und klickziehe eine Box über die Verts die Du loswerden möchtest – die werden dann blau, was anzeigt daß sie markiert sind. Dann einfach **Del** drücken ☺

Lösche alles was Du bis jetzt gemacht hast und setze die Verts in ihrer korrekten Position und Anordnung entschprechend dem Grid, was bedeutet daß man sie im Uhrzeigersinn einfügt. Stelle sicher, daß Du sie in die Top-Ansicht gesetzt hast, oder Du steckst in Schwierigkeiten. Falls Du einen Fehler machst, verwende die bisherigen Instruktionen. **Achte auf die Gittergröße!**

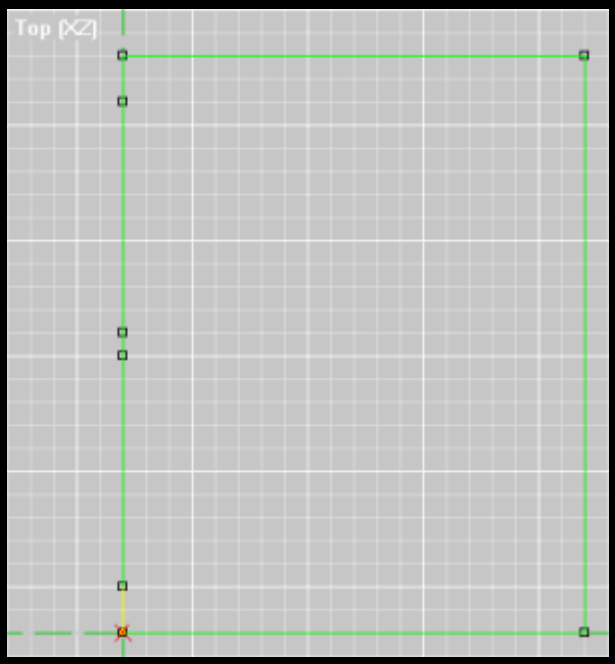


Um die Gridgröße zu ändern, drücke **BildAuf** um sie zu kleiner, und **BildAb** um sie größer zu machen. Wenn sich die Größe ändert, wirst Du rein- oder rauszoomen müssen um die Gitterlinien zu sehen, dazu drücke **Ctrl-Shift-Linke Maustaste** und bewege die Maus vor oder zurück, oder benutze die Tasten **S** und **W**. Alternativ geht auch **Ctrl-Mausrad**.

Wie Du rechts sehen kannst, erzeugen die Verts einen rechteckigen Umriß, welcher die Raumform darstellt. Nun gehe durch drücken von **Ctrl-F** in den Face-Modus, checke das durch einen Blick auf die Statuszeile und stelle sicher daß die Verts noch immer markiert (blau) sind, und drücke **Ins**.



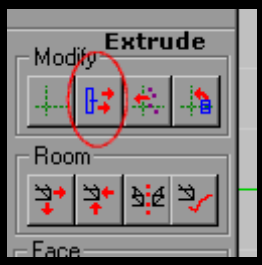
Du solltest nun ein Face haben wie dieses:

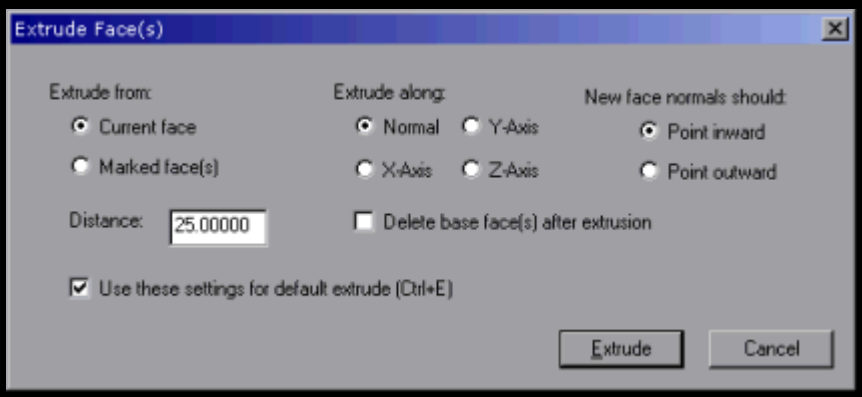


Dein erstes Gesicht ☺
Dies ist das System welches verwendet wird um Faces von Grund auf zu erstellen. Alles was Du machen mußt ist die Verts in der Anordnung zu setzen, wie Du die Edges haben willst. Hier sind es 100(x) mal 125(z) Descent3-Einheiten, verwende das Grid um nachzumessen.

Nun werde wir daraus einen Raum machen, mit einem Boden, Wänden und einer Decke - unter Verwendung der Extrude-Funktion, welche verwendet wird um einzelne Faces zu manipulieren. Klicke auf den Button **R** und was du zu sehen bekommst ist das Geometrie-Paneel, mit seinen vielen Buttons ☺

Klicke auf Dein neues Face um es current (grün) zu machen, dann klicke auf den Extrude-Button (rechts).





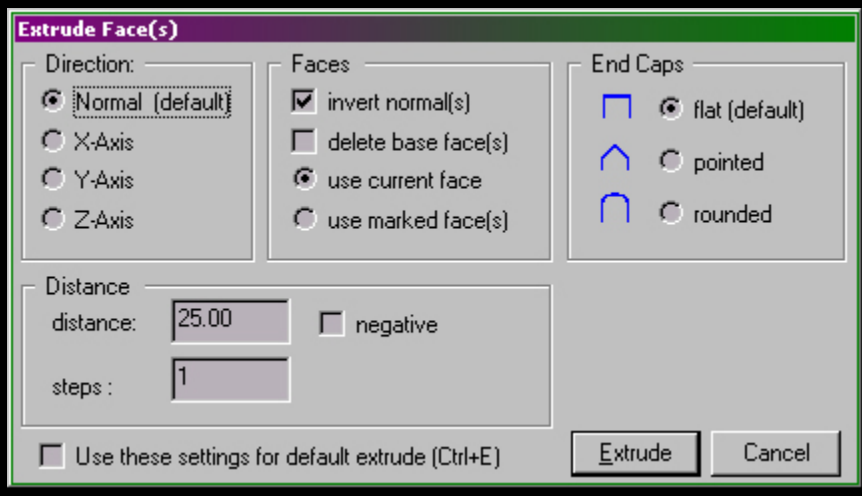
Wenn Du noch mit v39 oder älter arbeitest, dann kommt eine Dialogbox ähnlich wie diese hier, setze die Optionen entsprechend dem Bild links.

Extrude From: Current Face/Marked Face(s) – Extrudiert von einem einzelnen current Face, oder mehreren markierten Faces. Um mehrere Faces zu extrudieren, mußt du alle diese Faces markiert haben und eins davon current.
Distance: xx.xxxxx – Wie weit die Faces Extrudiert werden sollen. Als Daumenmaß gilt: 10 Units entsprechen 1 Schiffslänge/-breite/-höhe.
Extrude Along: Normals/X-Axis/Y-Axis/Z-Axis – Die Richtung in welche die Face(s) extrudiert werden. Bei Auswahl von ‚Normal' wird das Face in die Richtung extrudiert, in welche die Normal zeigt.

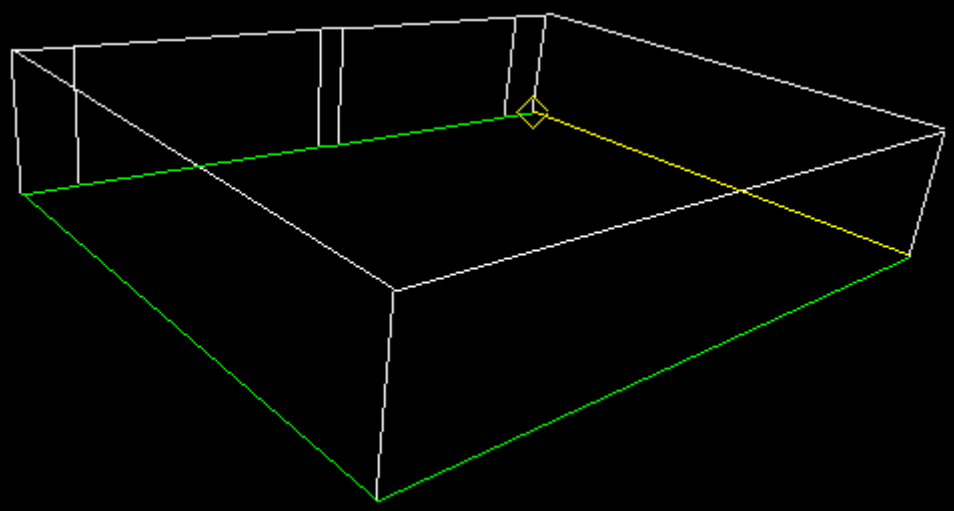
New Faces Normals Shou ld: Point Inward/Point Outward – Bestimmt die Richtung, in welche die Normals der neuen Faces zeigen sollen; bei Inward zeigen sie nach innen (Faces von innen sichtbar) bzw. nach außen bei Outward (Faces von außen sichtbar).

Wenn Du die neue v40 hast, sieht Dein Dialogfeld so aus:

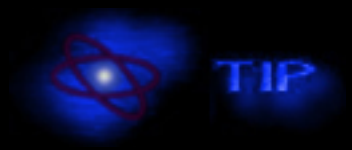
Faces - Hier gibst Du Dein Basisface an, ob vom current oder vom/n markiert/en face/s. Hier sagst Du auch in welche Richtung die Normals der neuen Faces schauen sollen; standardmäßig schauen sie in die selbe Richtung wie das Basisface.
Distance - Hier gibst Du an wie weit. Ein Haken bei negative kehrt die Richtung um.
Direction - Die Richtung, in die's gehen soll. Wenn bei Direction 'Normal' ausgewählt ist, geht es vom Face aus in Richtung der Normal.
Übernimm die Einstellungen aus dem Bild das für Deine Version zutrifft,



Dann drücke **Enter**... und dann hast Du's:



Woohoo! Ein Raum! Wie du sehen kannst hat er einen Boden, eine Decke und Wände - wobei eine davon in Segmente gesplittet ist. Wenn Du den Raum mit dem Grid in allen 2d-Ansichten ausmißt, sollter er 100 x 25 x 125 Einheiten messen. Wenn das so ist, gehe nach File... Save Room As und speichere den Raum als `box1.orf`.



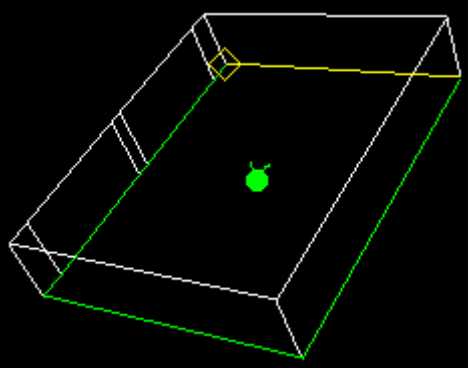
Wenn du mehr als zwei Vertices in einem Face hast, welche in einer Linie liegen dann erzeugen die Verts in der Linie - nicht die an ihren Enden - separate Faces, wenn sie extrudiert werden. In unserem Beispielraum haben wir sechs Vertices in einer Linie, und nach der Extrusion haben wir fünf separate Faces. Mit der Zeit wirst Du aus dieser Vorgehensweise den vollen Nutzen ziehen.

Das wars fürs erste, als nächstes machen wir aus dem Raum einen Level ☺

013 - Ein Level anfangen

Robo

Wenn Du bis hierher gekommen bist, ist es denke ich sicher zu sagen daß Du interessiert bist. Das ist gut, weils nun noch interessanter wird ☺



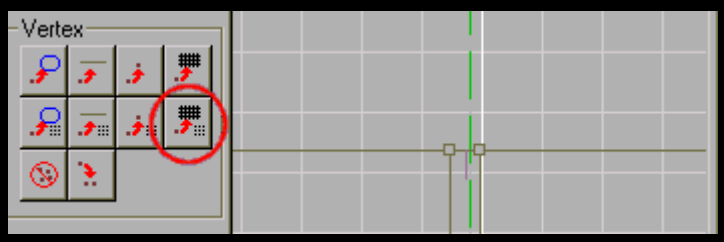
Öffne D3Edit und gehe nach File... New, wähle New Level - Import Room. Finde `box1.orf` und klicke OK um ihn zu öffnen, dann gehe in die Room View. Ich wette, Du denkst Dir dasselbe, was sich jeder First-Timer denkt: *Was zum #!%£&*&% macht dieser grüne Punkt da? Da paßt was nicht...*

Defakto ist alles im Lot. Dieser grüne Punkt ist der Spielerstartpunkt (Player Start) oder Spawnpoint. Er ist deshalb hier, weil Du es nicht länger mit einem simplen Raum zu tun hast, sondern mit einem Level.

Noch sind wir nicht fertig, wir haben noch immer ein paar Sachen zu editieren und ein Problem zu lösen. *keuch* Was? Ein Problem? O_o

Schau im Zweifelsfall zwischen die Linien:

Wenn Du einen näheren Blick auf Deinen Raum wirfst, wirst Du vielleicht feststellen daß sämtliche Verts und Faces nicht am Grid auserichtet sind. Das bedeutet, wenn Du etwas an einer Wand ausrichten willst, wie etwa eine Lichtbefestigung oder etwas ähnliches, würde das komplett unmöglich zu machen sein. Alles was wir machen müssen, ist den Raum wieder auf das Gitter zu bekommen bevor wir irgendwas anderes hinzufügen. Wenn wir ihn hinterher ausrichten würden, würde das viele Fehler erzeugen – abhängig davon, wie komplex Dein Level ist. Das kommt daher, weil das Ausrichten Deinen Raum bewegt, und wenn Du bereits Räume angebaut hast werden diese separiert.



Gehe in den Vertex-Modus (**Ctrl-R**), prüfe das durch einen Blick in die Statuszeile und drücke **M** um alle Verts zu markieren. Dann klicke auf irgendein Face um es auszuwählen, das macht ein face grün und ein Vert gelb (beide sind dann current) und drücke auf diesen Button in der Geometrie-Toolbar (Button **R**):

Klicke auf die Top-Ansicht, dann auf den Button, dann in die Side-Ansicht und dann in die Front-Ansicht, jedesmal diesen Button drücken. Dies holt die Verts wieder in allen Achsen auf das Grid zurück, da nur zwei in jeder Ansicht dargestellt werden. Dies sollte das Problem gelöst haben, wenn nicht, nochmal probieren.

Jetzt gehe nach Window...Untitled.d3l - World View, was Du jetzt sehen solltest ist Dein Level auf einem Schwarzen Hintergrund. Dann gehe nach File... Save Level... und speichere den Level als `level1.d3l`. Wenn Du eine Dev oder Beta-Version von D3Edit verwendest, rechtsklicke auf den schwarzen Hintergrund und schalte Show Terrain aus. Wenn Du eine andere Version verwendest, drücke **Ctrl-T**, beide Methoden lassen diese nervigen weißen Punkte verschwinden. Speichere nochmals.

Nun checken wir ein paar Sachen aus, die folgenden Prozeduren werden Dein Level einrichten, so daß Du später nichtmehr herumpfuschen mußt.

Gib mir einen Namen:

Gehe nach File... Level Settings..., und vergib bei Level name einen Namen. Den siehst Du in der Mitte des Bildschirms, wenn das Level lädt. Ich weiß, daß das Layout dieser Dialogbox je nach Editor-Version unterschiedlich ist, hier D3Edit Dev8th. Trag unter Designer Deinen Namen ein und unter Copyright (mit dem Jahr), und gib Deinem Level eine Beschreibung wenn Du willst. Schließlich trage noch den Wert für die Gravity ein, was bewirkt daß Napalm-Blobs und Spew nach unten fallen. Ich mag sie bei 16.1

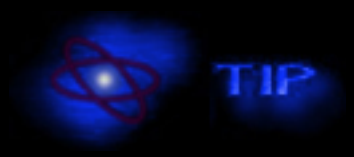
Kannst Du soweit zählen?

Als nächstes gehe nach File... Show Level Stats. Die aufpoppende Box enthält die gesammelte Statistik zu Deinem Level. Gegenwärtig hat das Level zehn Faces und 16 Verts, mit einem Objekt, den Spielerstart. Manche Leute finden allerdings, daß ihre fertiggestellten Levels sehr hohe Stats haben. Eins meiner Levels hat 21 Räume, 1460 Faces, 2128 Vertices, 22 Portale und 42 Objekte. Und das ist nur ein kleiner Level! ☺ Du kannst diese Stats in der Textdatei zu deinem Level angeben.

Problemzeit:

Nun werden wir den Level auf Fehler prüfen. Gehe nach File... Verify Mine. Wie Du sehen kannst sind alle Zahlen null, was gut ist, wir haben keine Fehler. Falls welche da wären, würde bei der entsprechenden Zeile eine von null verschiedene Zahl stehen, zB. `1 Bad Shell`.

Schließlich gehe nach File... Compute Boa Vis Table und wähle OK. (Boa steht für **B**ig **O**le **A**rray) Dies berechnet, welche Räume von Punkten in anderen Räumen sichtbar sind, sodaß D3 nicht den gesamten Level zeichnen muß, wenn Du drin herumfliegst - nur was Du siehst. Das ist absolut essentiell beim Bauen, Testen und schließlich beim Releasen – es muß immer berechnet werden. Mehr über die Boa Vis Tabelle und ihre Beziehung zu Räumen gibt's dann unter Neue Räume anfügen.



Wenn Du eine finale Version releast, oder ein Update, ist es eine gute Idee sich eine Checkliste zu machen, mit den Sachen die im Level zu ändern, hinzuzufügen oder zu entfernen sind.

Als nächstes werden wir ein bischen texturieren, es wird Zeit diese Default-Textur loszuwerden und ein wenig Metall einzusetzen. (Oder Hirn!)

File: 014-tut4.zip

014 - Texturierung

Robo

So, Zeit um ein wenig zu texturieren! Das sollte kurz und bündig sein.

Öffne `level1.d3l` und gehe in die World View. Rechtsklicke in den Hintergrund und wähle Textured with Outline. Aktiviere das Textur-Paneel mit dem Button **Tx**. Dieses Paneel verwendet man um den Level zu texturieren, bemalen sozusagen, Texturen auszurichten, aufzugreifen und auszuwählen

Das macht es alles:

Grab: Nimmt die Textur vom current face auf und legt sie in die Current Texture-Box.

Align: Ermöglicht es Texturen zu drehen, verschieben, strecken und quetschen.

To Current: Fügt die current Texture in Dein current Face ein.

To Marked: Fügt die Current Texture in alle markierten Faces ein.

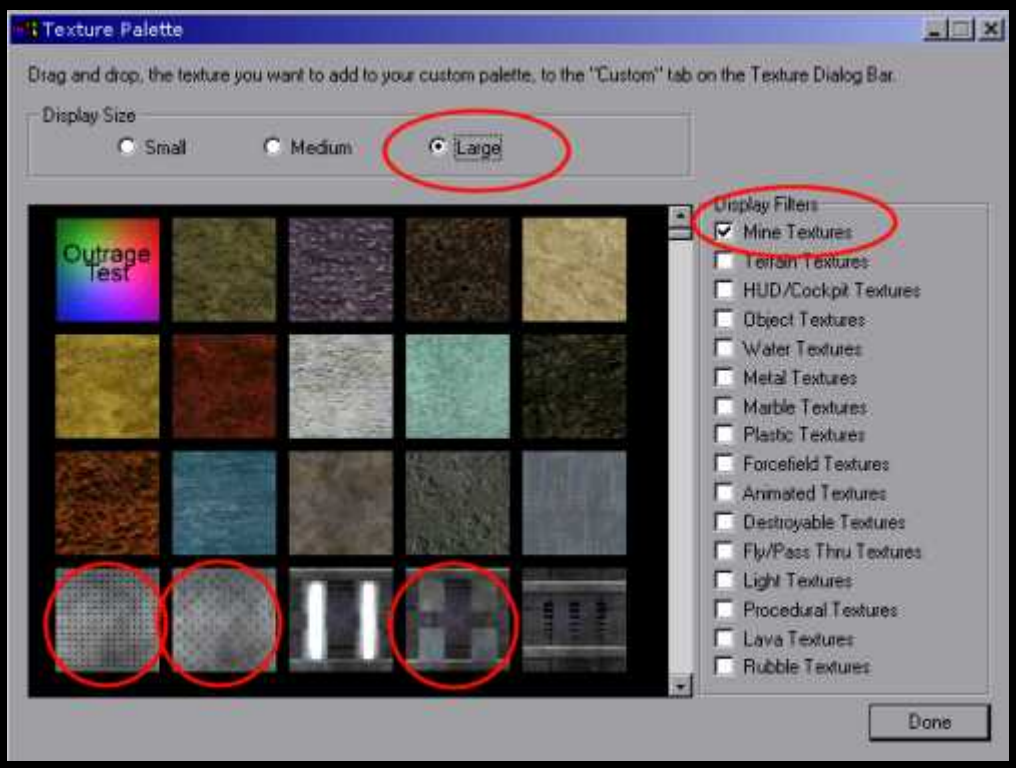
Propagate: Bewirkt, daß sich Texturen und/oder ihre Ausrichtung auf andere Faces ‚ausbreiten'.

Choose: Läßt eine Liste aller Texturen aufpoppen.

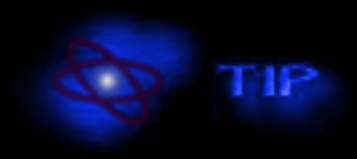
Custom Box: Enthält alle Texturen, die Du aus der Texturen-Liste gewählt hast.

Level Box: Enthält alle Texturen, die im Level enthalten sind.

Keine Sorge, die Verwendung dieses Panels wird Dir ganz natürlich erscheinen, wenn Du erst mal angefangen hast.

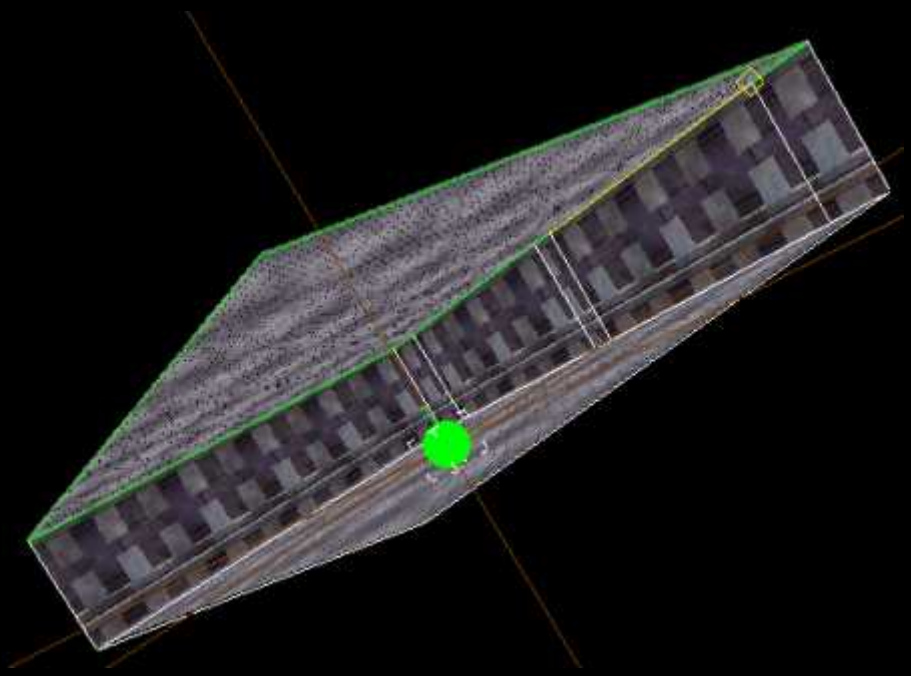


Klicke auf **Choose...** um die Texturen-Liste zu öffnen. Dann wähle Large und dann Mine Textures. Wie Du sehen kannst sind 949 Texturen zu laden, also ein wenig Geduld. Um eine Textur zu verwenden, muß sie in Deiner Custom Texture-Box sein – dazu klicke auf eine Textur und ziehe sie über die Box, dann laß sie los. Die Texturen die wir brauchen sind: **LWceil.tga1, LWflor.tga1** und **LWout2.tga1**. Sie sind alle fast am Anfang der Liste, so daß sie einfach zu finden sind.



Plane immer bevor du was machst. Das beinhaltet auch Texturen-Schemas. Letzte-Sekunde-Änderungen an der Geometrie können Fehler erzeugen, wenn sie nicht ordentlich ausgeführt weden.

Schließe die Liste und gehe in die Current Room View. Geh in den Face-Mode (**Ctrl-F**) und drücke **M** um alle Faces zu markieren, dann wähle Lwout2.tga1 als Current Textur aus und drücke dann To Marked, dies appliziert diese Textur auf alle markierten Faces. Danach klicke auf den Boden des Raumes um ihn current zu machen, wähle Lwflor.tga1 als current und drücke To Current. Mache dasselbe mit der Decke, verwende aber LWceil.tga1. Du solltest in der 3d-Ansicht (Rechtsklick) Textured with outline zu diesem Resultat kommen:



Speichern...

Schließlich und endlich fängt es an gut auszusehen. Als nächstes werden wir Unseren Level beleuchten und dann einen ersten Testflug unternehmen. Fortfahren!

File: 015-tut5.zip

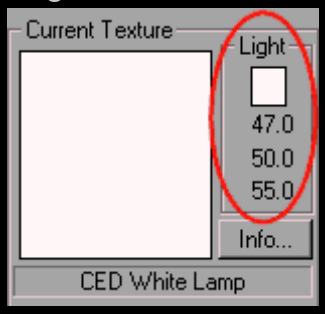
015 - Beleuchtung

Robo

Wir haben den Level nun texturiert, um ihn optisch ansprechender zu machen. Nun werden wir das noch verfeinern.

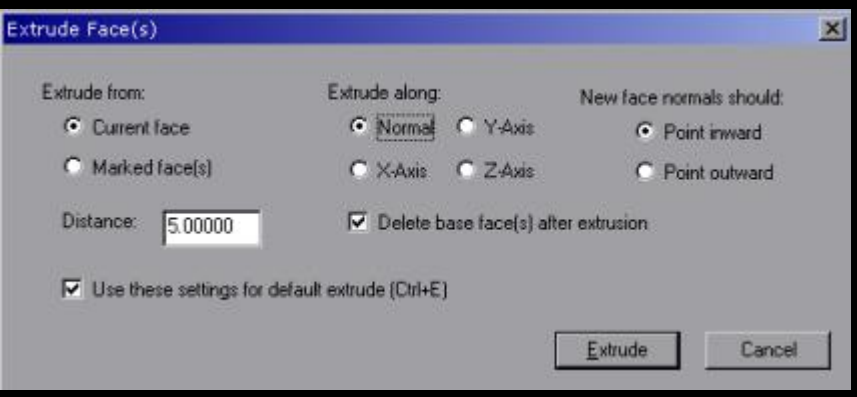
In D3 wirst Du bemerkt haben, daß manche Räume dunkler als andere sind, und manche vielleicht gefärbte Flecken an der Wand haben. Das ist wegen D3's Fusion Engine. Diese enthält ein kleines nettes Feature mit dem Du Deine Räume wie Du willst beleuchen kannst, manchmal hübsche Level erzeugen. D3Edit verwendet das um Dir zu helfen, Deine Level einfach und simpel zu beleuchten, dank dem Radiosity-Programm.

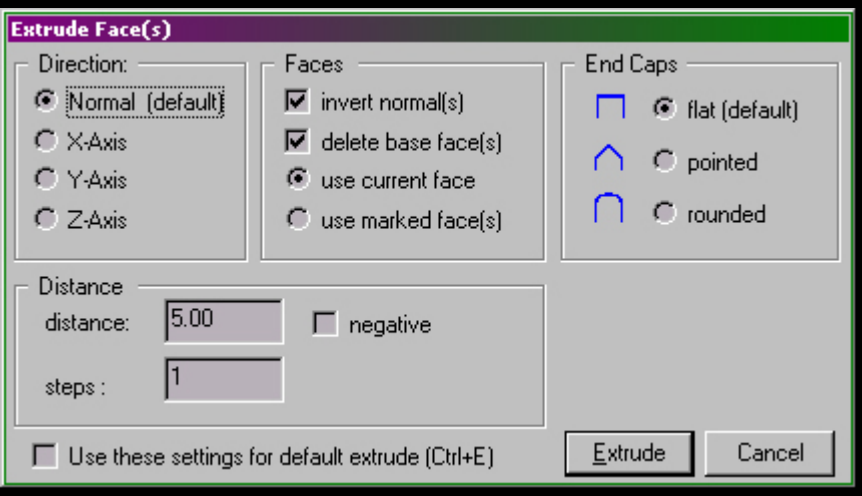
Licht wird von Light Textures in bestimmten Mengen und Farben emittiert. Es ist ganz ähnlich dem RGB-Farbmodell, drei Zahlen welche bestimmte Farbtöne und Helligkeiten erzeugen.



Rechts haben wir eine Light Texture genannt CED White Lamp. Ich verwende diese Textur manchmal in großen, offenen Bereichen um viel Licht zu erzeugen. Eingekreist sind die verschiedenen Werte von rotem, grünem und blauem Licht zusammengemixt zu sehen. Zum Beispiel: 17,17,17 ist eine moderate Helligkeit, 4,4,4 ist ein dämmriges Licht und 255,170,170 ist ein helles pinkfarbenes Pfirsich. Das Quadrat über den Zahlen zeigt die Lichtfarbe, paß auf daß du keinen blendest, wie in meinen ersten Levels! Ray Charles müßte in meinem zweiten Level die Augen zuzwinkern :P

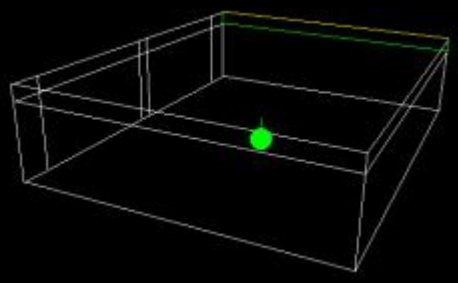
Es werde Licht:





Öffne `level1.d3l` und gehe in die Current Room View. Geh in den Face-Mode (**Ctrl-F**) und wähle die Decke (stelle sicher daß es auch wirklich die Decke ist), drücke die **Leertaste** einmal um es zu markieren. Drücke **N** um das Face umzudrehen, was die Normal dazu bringt nach oben zu zeigen (kann in einer der 2d-Ansichten überprüft werden), dann drücke **U** um alles abzumarkieren. Selektiere erneut die Decke und klicke auf den Extrude-Button, übernimm die Einstellungen aus dem Deiner Editorversion entsprechenden Bild links (oberes für v39 und älter, unteres für v40) und drücke Enter:

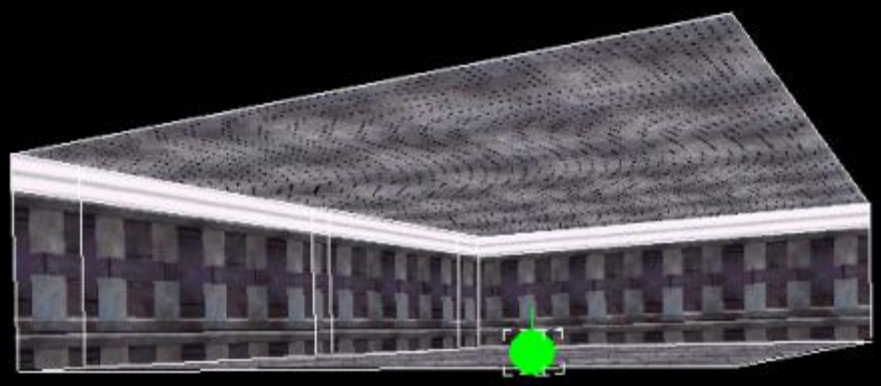
Das Ergebnis sollte etwa so wie rechts aussehen. Durch die Extrusion der Decke haben wir das ‚Dach' in unserem Raum angehoben, was uns ein Set von kleineren Faces im oberen Bereich liefert welche wir zu Lichtern machen werden.

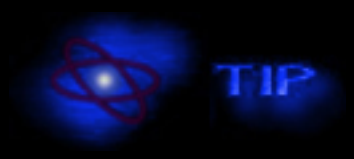


Du wirst bemerken daß dies ähnlich dem ist, was Schplurg in seinen Tutorials gemacht hat. Das ist, weil es einfacher für Dich zu machen ist und es ist beinahe streßfrei.

Licht texturieren:

Öffne die Texturauswahl-Liste mit Choose..., klicke auf Light Textures, und dann finde und übertrage LightCrossing03x. Dies wird unsere Lichttextur sein. Ziehe eine Box über das obere Set von Faces welches wir erzeugt haben um es zu markieren (blau), und stelle sicher daß du diese Textur ausgewählt hast – drücke **To Marked**. Das hat die Textur schnell und einfach auf alle markierten Faces gebracht, aber dies hat auch die Decke texturiert. Also wähle das Decken-Face aus, in der Current Texture-Box wähle LWceil.tga1 aus und drücke **To Current**.
Du solltest jetzt folgendes kriegen:





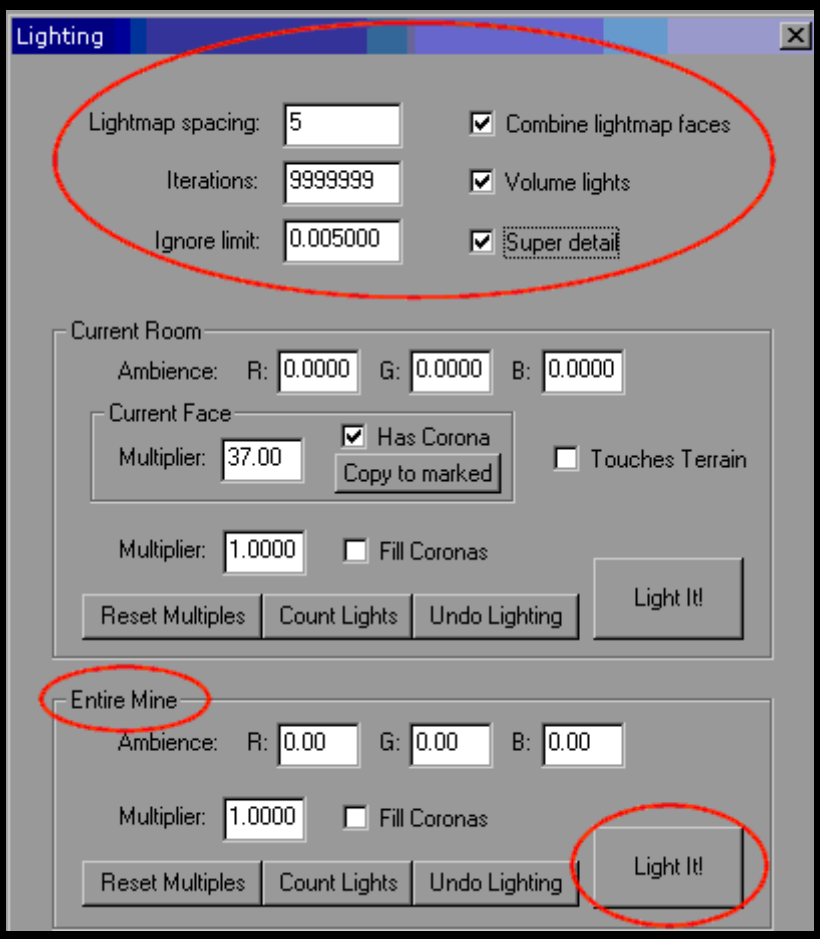
Versuche nicht zuviele helle oder große Lichter in Deinen Levels zu haben. Das kann nach einiger Zeit hart für die Augen der Spieler werden, was sie müde macht (und Kopfweh auslöst!). Außerdem peppt das Deine Beleuchtung auf und läßt Dein Level besser aussehen, wenn Du es mehr atmosphärisch haben willst.

Eine letzte Sache ist noch zu erledigen, nämlich das Lichtberechnungs-Programm. Gehe in die World View und speichere Dein Level, dann gehe nach Window...Lighting um den Beleuchtungsdialog aufzurufen. Übernimm die Einstellungen aus dem Bild auf der nächsten Seite und klicke **Light it!**

Lightmap Spacing: 5 – Ignorieren, am besten so lassen wie's ist.
Iterations: 9999999 – Gibt an wie oft das Licht von den Faces des Levels reflektiert werden soll.
Ignore Limit: 0.005000 – Der Punkt, an dem die Beleuchtung so schwach ist, daß sie nicht verwendet wird.
Combine Lightmap Faces: Yes/No glättet die Beleuchtung über die Faces in deinem Level. Am besten auch das so lassen.
Volume Lights: Yes/No – Wenn an, werden Schiffe und Objekte vom Licht betroffen. Wenn aus, erscheinen sie in voller Helligkeit.

Super Detail: Yes/No – Wenn an, wird das Licht von der gesamten Fläche eines Faces emittiert, welches eine Light Texture hat. Wenn aus, kommt das Licht nur aus dem Zentrum der Licht-Texturen.

Abhängig von der Leistung Deines Computers sollten diese Berechnungen in einigen Sekunden erledigt sein. In großen Levels mit vielen Lichtern kann das aber auch Stunden dauern ☺
(es kann auch Tage und Wochen dauern...)



Volume Lights und Super Detail beeinflussen die Berechnungszeit für die Beleuchtung, aber auch das Aussehen des Levels. Volume Lights ist ein Muß, aber wenn Dein Level so aussieht wie es soll könnte es eine gute Idee sein Super Detail abzuticken.

Speichern...

Als nächstes werden wir den Level testfliegen.

Zusatzinfo: Lighting Guide

An dieser Stelle nun eine Erläuterung der Beleuchtungs-Parameter, verfaßt von Chris "Gwar" Ledwith und Jason Leighton, basierend auf der Lighting-FAQ von Outrage; beide haben bei Outrage gearbeitet. Dieses Tut konnte ich noch über www.archive.org finden.

Ragil Ral

Texture Light

Die Zahlen neben der Texture sind in der Einheit „Watt pro Qradratfuß“. Dies bedeutet, daß die Größe des Faces das die Lichttextur trägt beeinflußt, wie hell das Licht scheint. Ganz wie eine kleine Glühbirne nicht soviel Licht abgibt wie eine große, leuchtet ein kleineres Face nicht so stark wie ein großes (vorausgesetzt, das beide Faces dieselbe Textur haben).

Texture Reflectivity

ist im Bereich von 0 bis 1.0. Reflectivity bestimmt, wieviel Licht zurückgeworfen wird, wenn eintreffendes Licht von der Textur zurückgeworfen wird. Dieser Wert ist sehr wichtig, wenn Du einen bestimmten Look für Dein Level kreieren willst (sei es rostig, steinig, verchromt, etc.). Je höher der Wert, desto mehr Licht wird reflektiert und desto mehr verhält sich die Textur wie ein Spiegel.

Hier nun eine Erklärung der verschiedenen Beleuchtungs-Features des Editors. Jede Option hat zwei empfohlene Einstellungen oder Bereiche - eine fürs Designen, und eine fürs Releasen.

Eingabefelder:

Lightmap Spacing	Iterations	Ignore Limit
Spezifiziert, wie weit die Lightmap-Pixel auseinander sind. Je kleiner der Wert ist, desto besser werden Deine Schatten, aber das braucht auch mehr Speicher und erhöht die Berechnungsdauer für die Beleuchtung.	Spezifiziert, wie viele Faces Licht werfen. Je höher dieser Wert, desto besser Deine Beleuchtung, aber es braucht auch länger bis der Beleuchtungsprozeß abgeschlossen ist. Wenn Du einen Production-Level (also einen den Du releasen würdest) machst setze diesen Wert wirklich hoch an. Zur Veranschaulichung, manche Retail-Level in D3 verwenden 1 Million Iterationen.	Dies ist ein wichtiges Feld. Dies sagt der Radiosity-Engine „Wenn du Licht auf ein Face wirfst und es trotzdem nicht viel Unterschied macht, spar dir die Zeit“. Je kleiner dieser Wert, desto besser für die Beleuchtung, doch wieder steigt die zur Beleuchtung benötigte Zeit an. Wenn Du nur die Beleuchtung austestest und ein Gefühl bekommen willst wie die Sache aussieht, laß den Wert auf 0.005.
Design: 20 (Default-Wert) Release: 5 bis 8	Design: dreimal (Anzahl an vorhandenen Lichtern) Release: ein großer Wert, groß genug um volle Konvergenz zu erreichen	Design: 0.005 (Default-Wert) Release: 0.0005 oder 0.00005 ist noch besser

Checkboxen:

Combine Lightmap Faces	Volume Lights	Super detail
Die Radiosity-Engine wird versuchen, koplanare Faces in eine Lightmap zusammenzufassen. Dies tut zwei Dinge: a) es verringert die Menge an Speicher, der für die Lightmaps verwendet wird, und b) es läßt die Nähte zwischen den Lightmaps verschwinden, ein ungewollter rasterisierender Effekt. Combine Lightmap braucht etwas Zeit, daher	Bestimmt die Beleuchtung in 'toter Luft'. Das ist der freie Raum im Level; dies wird Dein Schiff entsprechend einfärben wenn Du durch die Tunnels und Hallen fliegst. Ohne diese Option wird das Schiff immer volle Helligkeit haben, auch wenn es im Schatten sein sollte. Dies verlängert die Zeit für den Beleuchtungsprozeß um etwa 33%.	Nocheine wichtige Option. wenn sie aus ist, wird Licht nur vom Zentrum des Faces ausgesendet, wenn sie an ist von der gesamten Facefläche. Das bringt eine viel, viel akkuratere Beleuchtung aber verlängert die Rechenzeit für die Beleuchtung um einen großen Betrag. Verwende es nur für Release.
Design: Aus Release: An (Default-Wert)	Design: Aus (Default-Wert) Release: An	Design: Aus (Default-Wert) Release: An

Buttons:

Count lights in Mine - zählt einfach die Lichtquellen in der Mine und bietet an, das Feld Iterations für Dich auf den Wert Lichter*3 zu setzen.

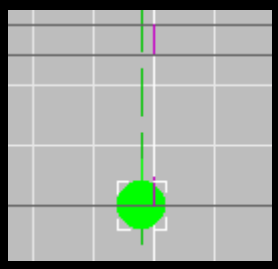
Light It! - Führt die Berechnung der Beleuchtung aus. Kann eine Weile dauern (Minuten bis Stunden), abhängig von den Optionen die Du gesetzt hast. Wenn das erledigt ist, poppt ein Dialog auf der sagt daß die Beleuchtung komplett ist. Sorg Dich nicht darum, wenn die Anzahl an ausgeführten Iterationen nicht den von Dir eingegebenen Wert erreicht; 100% Convergence ist alles was wichtig ist. SPEICHERE Deinen Level danach!

Da heutzutage, 2008, etwa acht Jahre nach Niederfassung dieses Textes, die Rechenleistung von PC's doch recht angestiegen ist empfiehlt es sich die Werte entsprechend anzupassen. Ich persönlich etwa setze bei Lightmap Spacing zumeist 1 als Wert.

File: 016-tut6.zip

016 - Den Level testen

Robo



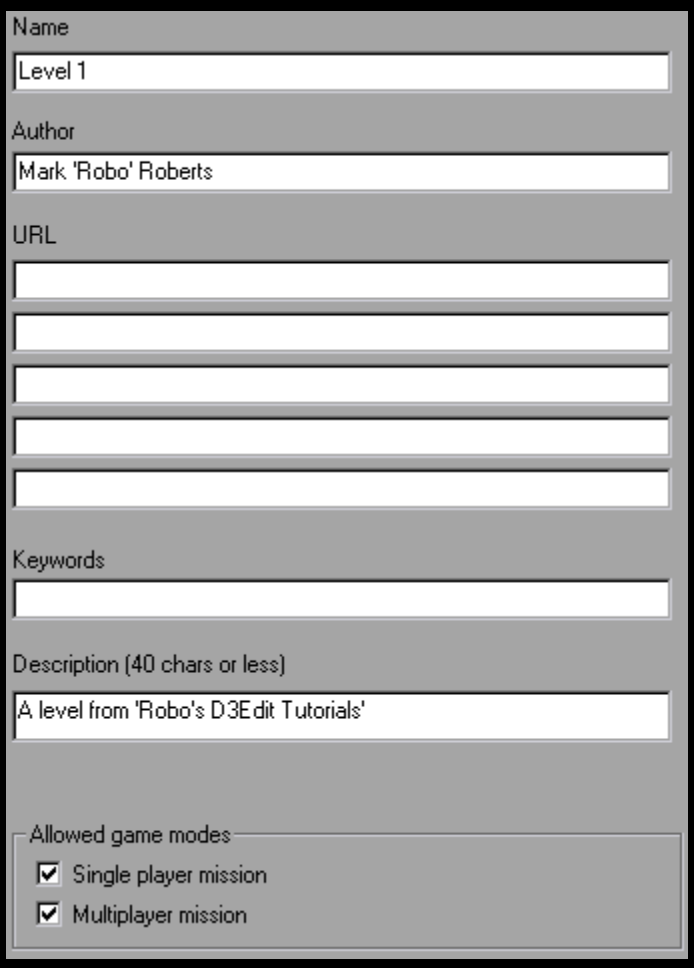
In diesem Tutorial werden wir nun den Level für einen Testflug in D3 vorbereiten, aber zuerst – wir müssen etwas richten, was nervig werden könnte.

Im Bild rechts kannst Du sehen, daß der Spielerstart recht nahe am Boden ist – genaugenommen im Boden! Wenn Du den Level startest, wird Deine Sicht durch den Boden geteilt sein, und du wirst zwischen dem Raum und dem Haus der Spiegel sein.

Was wir machen müssen, ist den Player Start auf eine Höhe von 10 Einheiten über dem Boden zu heben.

Öffne `level1.d3l` und gehe in den Objekt-Modus (**Ctrl-G**), dann klicke auf den Button **O** um das Objekte-Paneel aufzurufen. Im Objekt-Modus kannst Du Objekte selektieren, herumbewegen, drehen und löschen. Klicke ein paar mal auf den Player Start um ihn auszuwählen, dann klicke in die Front- oder Seiten-Ansicht und drücke einmal **8** auf dem Nummernblock um ihn zu bewegen. Dies hat den Spielerstartpunkt um 10 Einheiten nach oben verschoben. Problem gelöst.

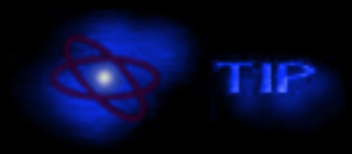
Speichere ab und öffne MN3 Edit (*Ich verwende dies in diesen Tutorials, da Quicktest in manchen Versionen oft unzuverlässig ist.*) Eine `.mn3`-Datei ist jene Datei, welche Deine Levels, Loadscreens, Skripten, Musik, Briefings und andere Dateien enthält und organisiert – alles in einem netten Paket, und dann komprimiert. Darin kannst Du die `.d3l`-Datei/Level Set-Namen, die Beschreibung, bis zu fünf Auto-Download-URLs, Schlüsselwörter angeben um die Spielmodi einzustellen und angeben ob oder ob nicht das Level Single oder Multiplayer zu spielen ist.



Klicke File...New. Unter Name trage jenen Namen ein, dem Du den Level vorher gegeben hast, Deinen Namen, und – wenn Du willst – eine Beschreibung. Die Auto-Download-URLs laß leer, da das Level noch nicht für Multiplayer geeignet ist. Es sollte etwa so aussehen:

Als nächstes gehe zu Action...Add Level. Klicke den Browse-Button neben der Level-Dateiname-Box, finde Dein Level und klicke OK. Klicke wieder OK um dieses Dialogfeld zu schließen und gehe nach File... Save As. Speichere Deinen Level in dein Missionsverzeichnis von D3, und Du hast fertig.

Starte Descent 3, beginne ein neues Spiel und finde Deinen Level. Viel Spaß!



Manche Leute (mich eingeschlossen) haben einen Bug in MN3Edit gefunden, welcher aber glaube ich nicht alle betrifft. Es scheint, daß man nicht über eine Datei zweimal drüberspeichern kann, stelle sicher daß die vorige Version gelöscht ist oder speichere unter einem anderen Namen, falls Du Probleme hast.

(016-tut7.zip)

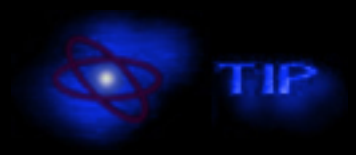
017 - Neue Räume anfügen

Robo

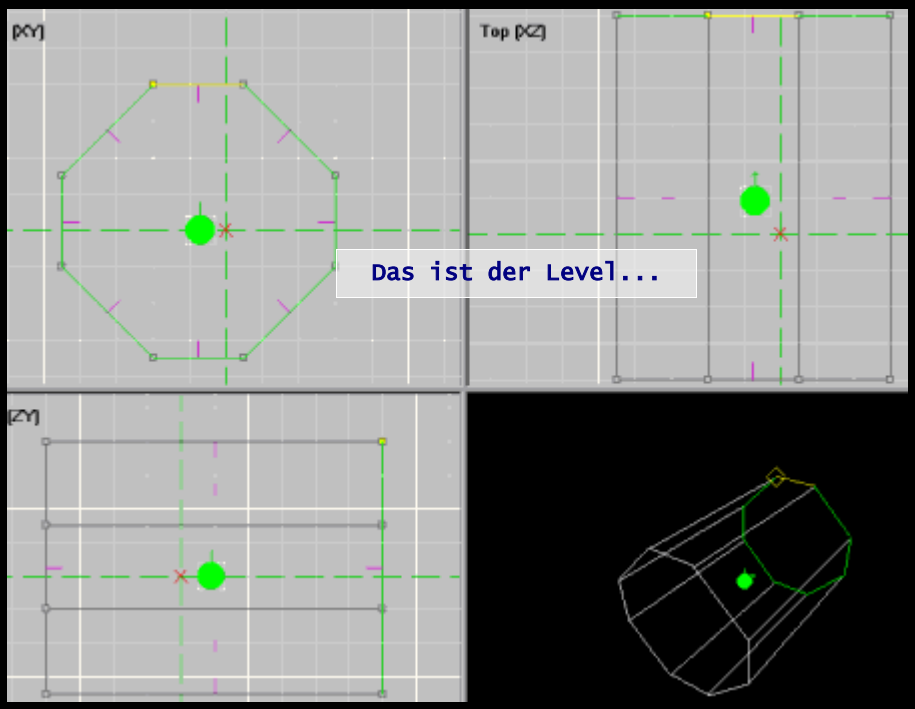
Wenn Du im Descent 3 herumfliegst zeichnet Deine Grafikkarte alle Faces, Texturen und Lichter so schnell sie das kann und schickt das an Deinen Monitor. Viel Details können Deinem System beim Verarbeiten einiges abverlangen. Um diesem Problem abzuhelfen, verwendet D3 die BOA Vis Table (**B**ig **O**l' **A**rray); ich erwähnte das bereits in ‚Ein Level anfangen'. Wenn du sie im D3Edit berechnest, schreibt es in den level welche Räume Du von jedem Punkt des Levels aus sehen kannst, sodaß wenn Du spielst D3 nur die Räume zeichnet die Du sehen kannst. Es prüft sehr effektiv.

Wieauchimmer, da sind zwei Sachen die Du machen mußt damit das funktioniert. Als Erstes mußt Du die BOA Vis Table in D3Edit berechnen und speichern bevor Du testest (offensichtlich). Aber zweitens, Du mußt Deinen Level in Räume aufgeteilt haben, falls nicht – ist's nutzlos!

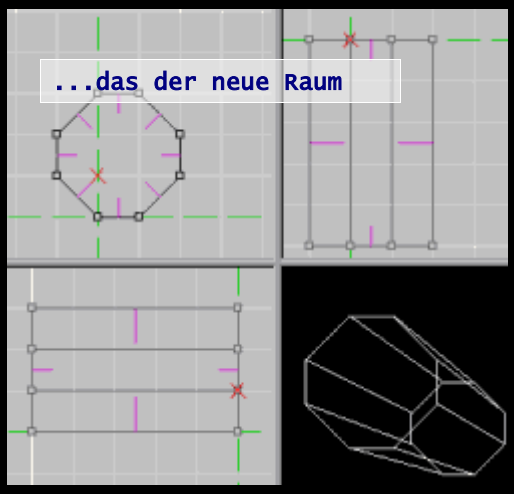
Diese Räume müssen nicht winzig sein und Du brauchst nicht viele, aber was Du brauchst sind genug Räume um Deinen Polygon-Count nieder und die Framerate hoch zu halten. Ich habe herausgefunden, daß ein viel an Räumen ein wenig unaufgeräumt wirkt, und dazu tendieren mehr Fehler zu erzeugen welche schwerer zu fixen sind, also wie viele Räume braucht es? Gut, ich neige dazu sie vernünftig groß zu machen und individuell. Ich trenne Bends und gerade Tunnels, Dogfighting-Räume und Powerup-Räume – alles was einen bestimmten Zweck hat. Du wirst Energie-Zentren, Wind-Tunnels und Räume mit Nebel oder Sound ohnehin separat machen müssen, weil diese Optionen den gesamten Raum betreffen. Aber bevor wir Räume separieren, müssen wir sie zuerst mal bauen ☺



Um durch alle Räume Deines Levels zu cyclen, verwende die **R**-Taste (In der World View).



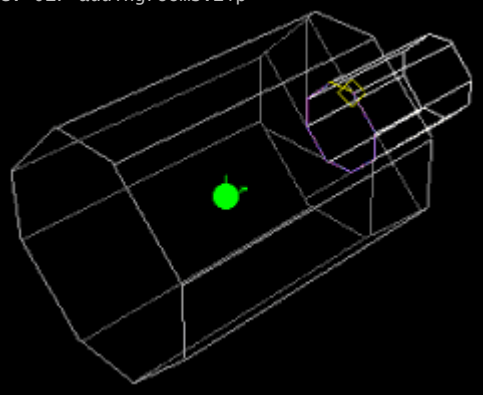
Rechts habe ich einen großen achteckigen Raum gebaut und in ein Level konvertiert... der Spielerstart ist bereits positioniert. Unten habe ich dann einen ähnlichen achteckigen Raum, wesentlich kleiner als der im Level, wir werden ihn dort dranhängen. Öffne beide Dateien aus der `.zip` und gehe in die Room View des Raumes. Gehe in den Face-Mode (**Ctrl-F**) und markiere alle Faces in diesem Raum. Dann mache eins der End-Faces durch



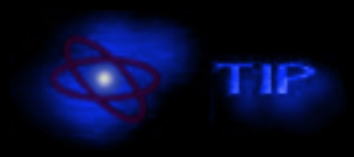
anklicken oder unter Verwendung der **F**-Taste current (grün). Nun sind wir bereit den Raum anzufügen. Stelle sicher daß Du alle Faces im Raum markiert hast und und ein End-Face current, dann gehe in die World View des Levels. Dort gehe nach Room... Place Room At

Current Room. Siehst Du den kleineren, purpurnen Raum? Das sagt Dir wo Du den Raum ins Level einfügen wirst. Wenn Du den Raum nicht einfügen willst, gehe nach Room... Unplace Room, ansonsten wähle Room... Attach Room.
Und das solltest Du jetzt bekommen:

File: 017-addingrooms.zip



Wir haben nun erfolgreich einen Raum zum Level hinzugefügt. Wenn Du **R** drückst, wirst Du bemerken daß die Linien einer der Räume mit einem helleren Weiß gezeichnet werden, was aussagt daß er current ist. D3Edit verbindet das current Face des Levels mit dem Current Face im Raum.
Im Bild ist unser neuer Raum current, und ein wenig heller. Aber was macht das purpurne Face da? Das ist ein Portal. Portale sind die Grenz-Faces zwischen den Räumen und sind unsichtbar, außer Du skriptest sie, sodaß sie sich anders verhalten. Du kannst sie in Kraftfelder verwandeln um Räume zu blockieren, und Du kannst sie in Trigger (Auslöser) umwandeln, um irgendwas an- oder abzuschalten – oder ein Skript-Ereignis auszulösen.



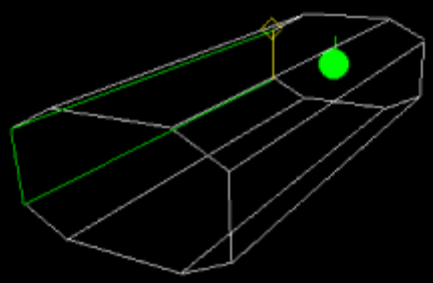
In komplexeren Situationen, wenn Du mehrere Räume hast die an verschiedenen Stellen angebaut werden sollen, mußt Du wissen wie Du den Raum anhängen sollst.
D3Edit attacht den letzten Raum, welchen Du angesehen hast und markierte Faces sowie ein current Face hat, also wenn Du in die falsche Room View geklickt hast mußt Du zurück und den Raum den Du anhängen willst erneut betrachten, bevor Du in anhängst. Konfuses Konzept... für manche ☺

Du erinnerst Dich an die BOA Vis Table? Zeit, sie zu verwenden. Gehe nach File... Compute BOA Vis Table und klicke **Yes**.
Du brauchst diese Fertigkeit, wenn Du ein voll spielbares Level mit vielen Räumen bauen willst, also verwende sie auch oft und ordentlich ☺

File: 018-objects_required.zip

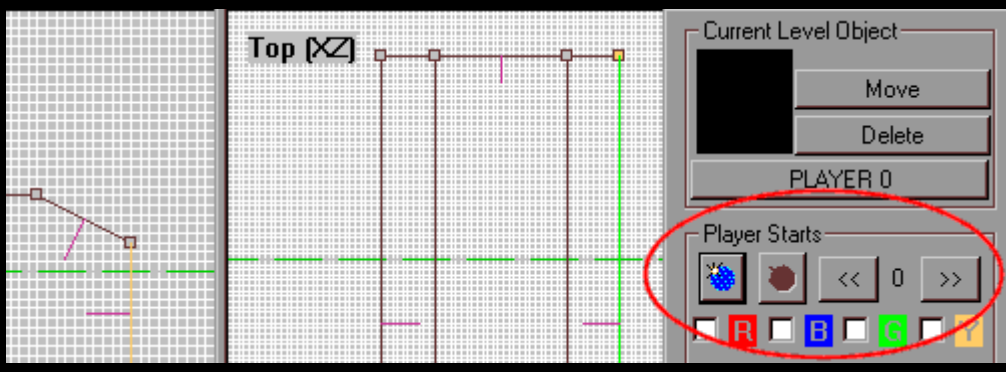
018 - Objekte &Player Starts

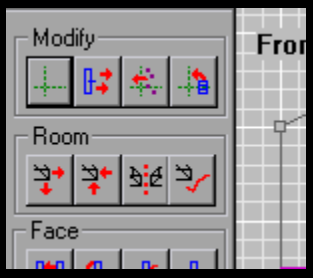
Robo



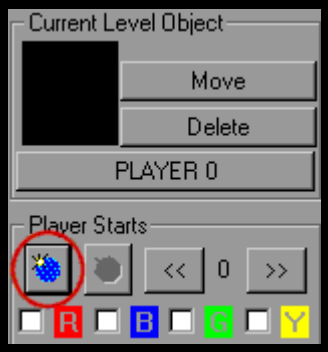
Hier werden wir eigenhändig einen zweiten Spielerstart in den kleinen Raum hier links hinzufügen, ebenso wie ein einzelnes Objekt.

Der Raum ist klein, aber adäquat für das was wir machen wollen. Verwende den Raum aus der `.zip`, da ich ein paar Sachen voreingestellt habe.
Also, legen wir los.
In diesem Raum habe ich einen Spielerstart am Ende, da dies ein Level ist. Dein Raum muß ein Level sein, bevor Du Spielerstartpunkte einfügen kannst; D3Edit würde Dich sonst ohnehin nicht lassen.
Gehe zur Current Room View und öffne das Objekte-Paneel mit dem **O**-Button. Der Teil des Paneels der uns im Moment interessiert ist dieser rot eingekreiste hier:

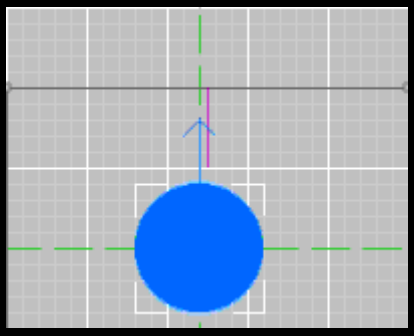




Öffne das Geometrie-Panel und klicke auf den Button ganz links oben: Set Reference Frame. Trage in die Felder von links nach rechts ein: **2042**, **-93** und **2090**. Das bringt ein paar Linien dazu, sich nahe an einem Ende des Tunnels zu schneiden. Diese Linien sind der Refernce-Frame (Referenzpunkt) und er wird verwendet, um Player Starts, Objekte, Wegpunkte und Nodes zu positionieren. Im oberen Bild siehst Du gerade ein anderes Beispiel eines Reference Frames.
Der Ort wo sich die Linien schneiden ist jener Ort, wo das Objekt platziert wird, wenn Du in die Ansichten klickst könntest Du ihn versetzen, also Obacht.
Nachdem der Referenzpunkt gesetzt ist, klicke auf den im rechten Bild eingekreisten Button im Objekte-Paneel.

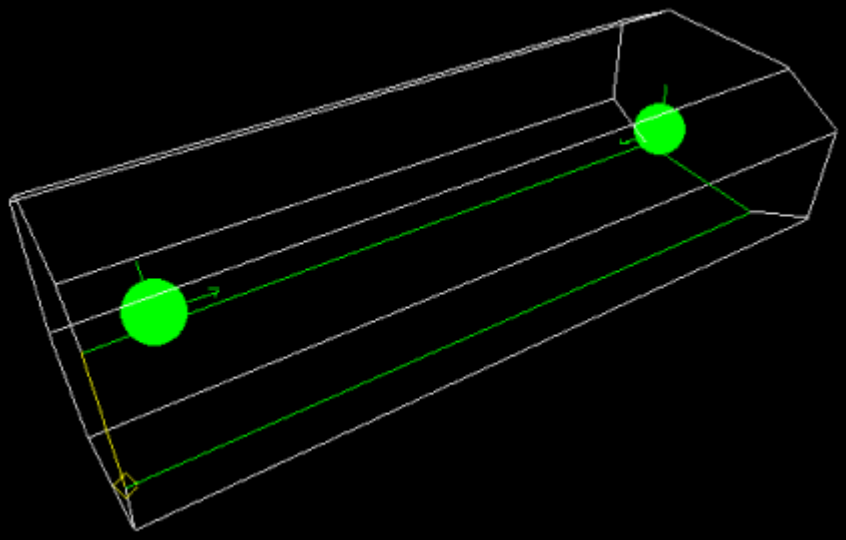


Nun solltest Du einen zweiten Player-Startpunkt im Raum haben, welchen Du in Deinen Ansichten siehst. Wenn Du den Button ‚>>' drückst, rechts vom anderen, hast Du den Spielerstartpunkt ausgewählt (was im Button darüber dann drinnensteht).
Beachte, daß keine der vier Checkboxen angetickt ist. Das bedeutet, der current Spielerstartpunkt kann von allen aus allen Teams benutzt werden. Wenn Du rot antickst, kann er nur vom Roten Team verwendet werden, dasselbe mit blau. Wenn Du **beide** antickst, können ihn beide Teams verwenden, aber weder das Grüne oder Gelbe Team. Klar?



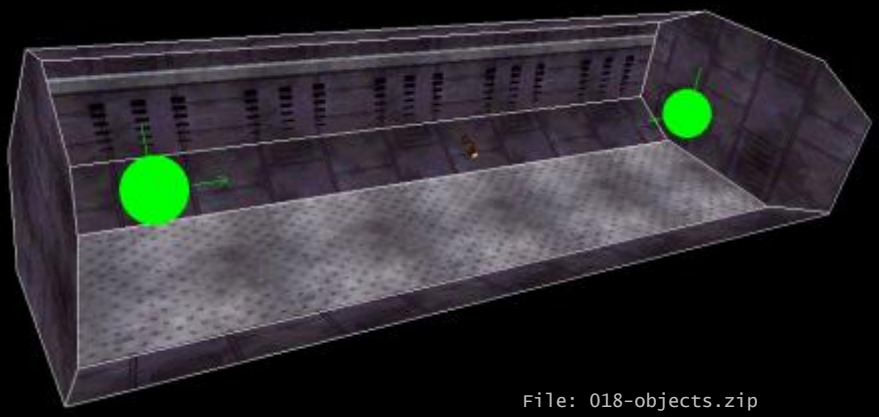
Siehst Du den Pfeil, der aus dem Objekt heraussteht (links)? Das ist die Richtung, in die Dein Pyro zeigt, wenn Du von diesem Punkt spawnst, was bedeutet daß du Dich dann einer Wand gegenübersehen wirst. Wollen wir das? Nee. ☺
Was wir jetzt machen werden ist das Objekt zurück in die korrekte Richtung zu drehen. Gehe in den Object Mode (**Ctrl-G**) und setze die Gridsize auf 50. Ich hoffe dein Referenzpunkt ist noch gesetzt, denn um den wird das Objekt gedreht. Stelle sicher, daß das Objekt selektiert ist und drücke viermal **7** oder **9** im NumBlock. Und da sind wir – in der richtigen Richtung ☺

Du solltest jetzt sowas haben, und schön brav **speichern**:



Nun wird's Zeit zu Powerups und Ähnlichem zu kommen. Setze Deinen Referenzpunkt auf **2042**, **-93** und **2050**, wo wir unser Objekt platzieren werden. Klicke auf Available Objects im Objekte-Paneel, was einen ähnlichen Dialog wie bei den Texturen aufruft. Ticke Powerups an und warte bis sie geladen sind.
Es funktioniert grundsätzlich gleich wie das Texturen-Paneel; ziehe eine einzelne Frag-Missile in die Custom Object-Box und schließe die Auswahlliste. Wähle die Frag aus, klicke auf Mine und dann drücke Insert (Tastatur oder den Button).

Speichern...



File: 018-objects.zip

Abschnitt B – Vertiefung in D3Edit

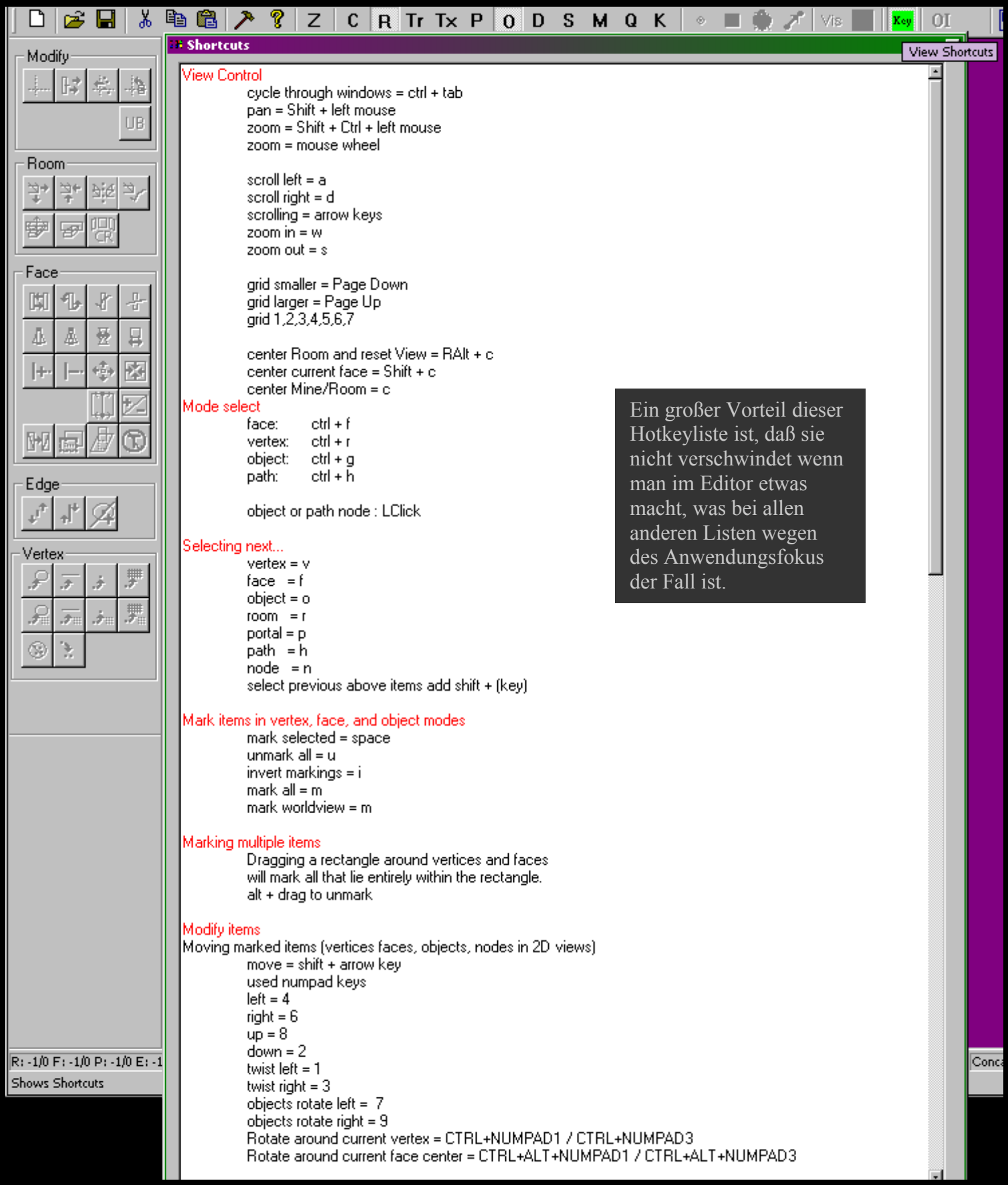
019	**Shortcuts einsehen**	Dieses Feature ist wohl unverzichtbar	Ragil Ral	69
020	**Das Geometrie-Paneel**	Alle Buttons Der Geometry-Bar einzeln, mit Beispielen! erklärt.	Papacat	70
021	**Quicktest mit D3Edit einrichten**	Einrichten des Quicktest-Toos.	WillyP	84
022	**Editoreinstellungen und -profile sichern**	Einrichten des Quicktest-Toos.	(LL)Dark	87
023	**Standard - Mod's**	Erklärt, wie man andere Spielmodi ermöglicht..	[DCG]Roadrunner)	90
024	**Tools Menü anwenden**	Einrichtung und Konfiguration des Tool-Menüs.	(LL)Atan	93
025	**Objekt Info Dialog**	Erklärung des OI-Dialogs.	(LL)Atan	94

Toc

019 - Shortcuts einsehen

Ragil Ral

Ohne sie wär's kaum möglich, einigermaßen schnell im D3Edit zu arbeiten. Es gibt da glücklicherweise diesen grünen Button, der die Hotkey-Liste anzeigt. Es gibt diese Liste also nicht nur (2x !) hier, sondern auch ‚vor Ort'. Also Schluß mit Ausreden, auswendig lernen aber zack!



Ein großer Vorteil dieser Hotkeyliste ist, daß sie nicht verschwindet wenn man im Editor etwas macht, was bei allen anderen Listen wegen des Anwendungsfokus der Fall ist.

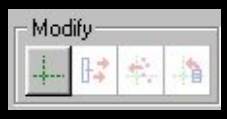
020 - Das Geometrie-Paneel

Papacat, auf v40 geupdatet

Hier werden alle Buttons der Geometry-Bar mit Beispielen erklärt.

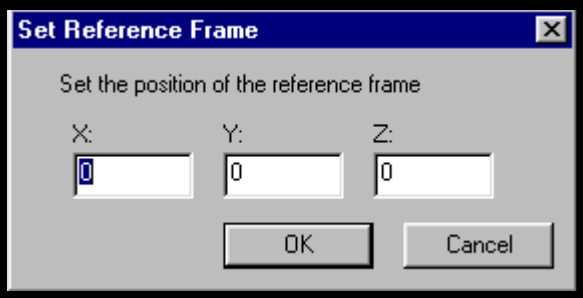
Modify:

Set Reference Frame



Öffnet ein Dialogfeld, wo die neue Position des Referenzpunktes hin soll.

Ich bevorzuge **Ctrl-Klick** um den Referenzpunkt zu verschieben. Wenn du auch die 3. Dimension gesetzt brauchst mußt Du in eine andere 2d-Ansicht **Ctrl-Klick**en. In der Regel bist Du nur mit einer Ebene befaßt um Verts darin einzufügen oder einer Achse zum Rotieren oder Benden.

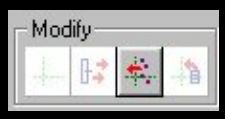


Extrude

benötigt:ein current oder ein oder mehrere markierte faces

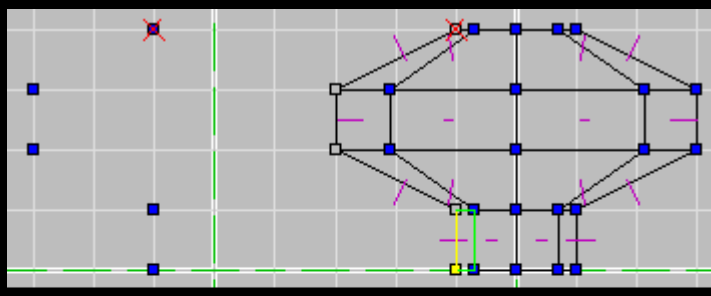
Extrudiert ein Face zu einem 3d-Objekt. Großartig für Säulen, Röhren, Platformen... etc. siehe auch Hydras Tutorials zum Thema Extrudieren (Abschnitt C, Fortgeschrittenes Bauen)

Lathe (‚Drehbank')

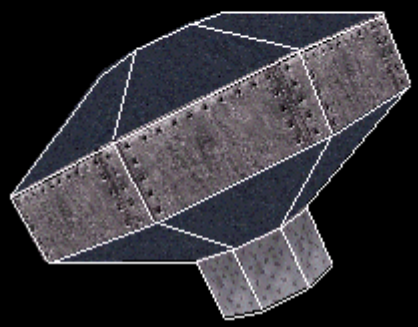


benötigt:mehrere markierte Vertices

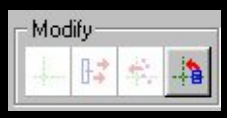
Erzeugt ein 3d-Objekt indem es die markierten Verts in einem zirkulären Muster kopiert.
Siehe auch hierzu Hydras Tuts (das Beste was ich zu dem Thema gesehen habe [Abschnitt C, Fortgeschrittenes Bauen])



Ich bevorzuge Objekte separat zu lathen und dann einzupassen. Und es ist nicht nur für Objekte; verwende Segmente für gerundete Hallen oder um Ecken abzurunden.

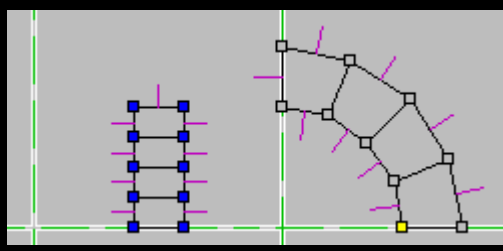


Bend



benötigt:mehrere markierte Vertices

Biegt die markierten Verts über eine Achse. Siehe hierzu Schplurgs und die anderen Tuts zum Thema.
Es ist gut um Röhren oder Säulen zu biegen, hat aber einige Nebeneffekte. Nachdem Du was gebogen hast, mußt du die UV's der Texturen zurücksetzen und sie neu ausrichten. Sieh dazu ‚Biegen ohne Bend' und ‚Einfach nein zu Bend', eine Alternative zum Bend-Tool. Eine andere wäre das Lathe-Tool.



Unbend

Sollte beim Benden mal was daneben gehen, hilft dieser Button weiter. Es ist das einzige Undo, das D3Edit kennt

---

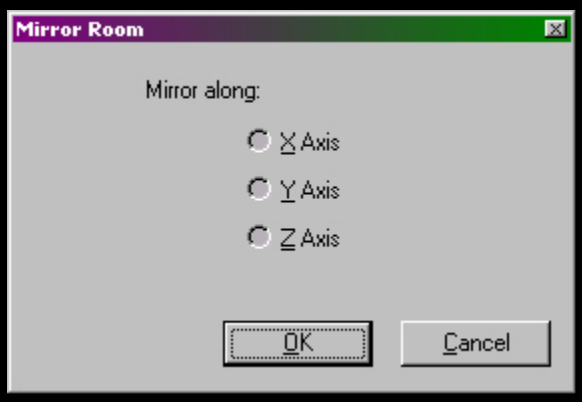
Room:

Expand / Contract Room

Damit könnt ihr den gesamten Raum aufblasen oder verkleinern. Bei den Portalen aufpassen!

Mirror Room

Damit könnt ihr den gesamten Raum entlang einer Achse spiegeln. Im Dialog müßt Ihr Euch für eine Achse entscheiden.



Verify Room

Ruft Verify auf, das Prüfprogramm.

Split Room

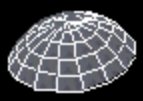
Damit zerteilt Ihr Räume, wird noch in Nr029 - Raum teilen behandelt.

Create Room from Marked Faces

benötigt: markierte Faces

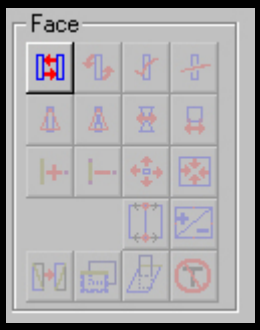
Damit wird aus einer Gruppe markierter Faces eine neue `.orf` erstellt. Es öfftnet sich eine neue Raumansicht, mit einem Raum der alle Faces enthält welche zum Zeitpunkt des Aufrufes markiert waren.

Links: Markierte Faces ->Button klicken->ergibt rechts.



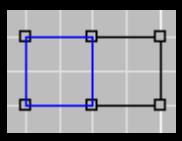
Face:

Combine Faces



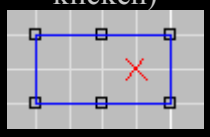
Verwende das, um deinen Face-Count zu reduzieren. Achtung: Du kannst/darfst nur ein Face markiert haben, wenn Du den Button anwendest. Alternativ: auf ein Face neben dem Current **Strg-Shift-Klicken**.

1. Schritt
ein Face markieren



2. Schritt
ein anliegendes Face current machen

3. Schritt
Tool ausführen (=Button klicken)

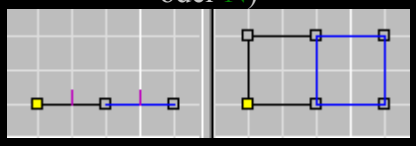


Flip Faces

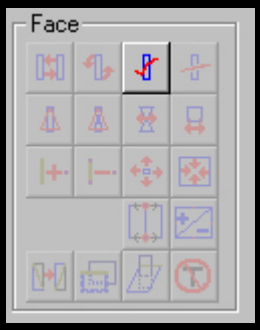
Dies dreht alle markierten Faces (=deren Normals) um. Tastenkürzel ist **N**.

1. Schritt
alle Faces die gedreht werden müssen markieren

2. Schritt
Tool ausführen (=Button klicken oder **N**)



Face Planar Check

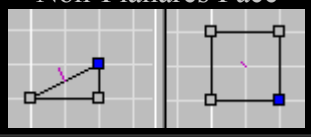


Prüft markierte Faces auf Ebenheit und bietet dann an sie für Dich in Dreiecke zu splitten. Die neuen Faces bekommen die Default-Textur, also mußt du die dann alle überarbeiten.

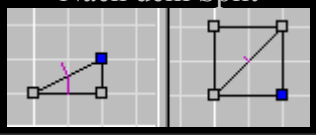
Gut um kleinere Faces zu fixen, aber Du hast keine Kontrolle darüber wie die Faces gesplittet werden.
Mit vier Verts und einer leichten Biegung im Face wird das nichts machen.

Bei fünf oder mehr könnte es unordentlich werden (Beispiel 3). Nur ein Teil des Faces müßte gesplittet werden. Die nächsten zwei Tools werden Dir die Kontrolle über das Splitten geben.

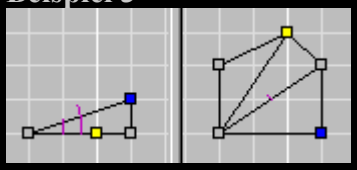
Beispiel 1
Non-Planares Face



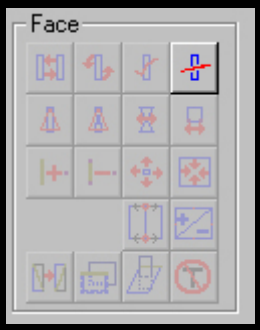
Beispiel 2
Nach dem Split



Beispiel 3



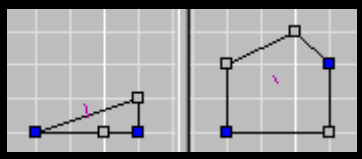
Split Current Face



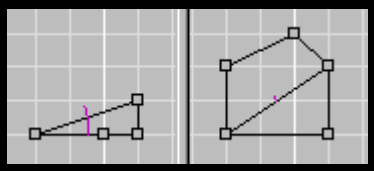
benötigt: genau zwei markierte Verts im selben Face

Erzeugt eine Kante (=Edge) welche zwei markierte Verts verbindet.Du kannst das auf planare als auch auf non-planare Faces anwenden.

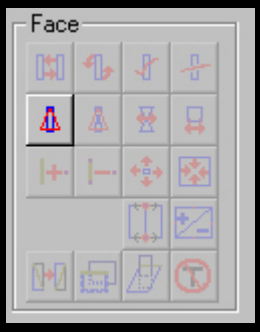
1. Schritt
Markiere die Verts die durch die Neue Kante verbunden werden sollen



2. Schritt
klicke auf das Face um es current zu machen. Button klicken um dieses Ergebnis zu kriegen

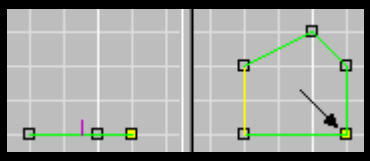


Triangulate Current Face

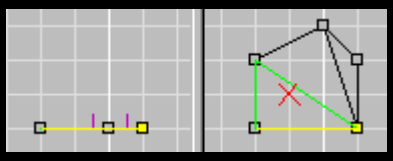


Dies teilt ein Face in Dreiecke, wobei die neuen Kanten (=Edges) dem current Vert entspringen.
Es funktioniert mit planaren als auch non-planaren Faces.

1. Schritt
Markiere die Verts die durch die Neue Kanten verbunden werden sollen



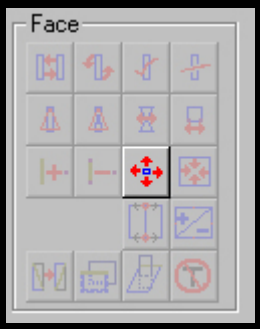
2. Schritt Gegebenenfalls das richtige Vert current setzen (**V**); Button klicken gibt das:



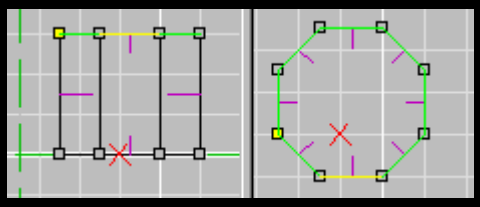
Triangulate Non-Planar Faces

Das splittet alle markierten Faces gleich wie ‚Face Planar Check', aber ohne vorher zu prüfen und die Änderung anzubieten. Es findet alle markierten Faces welche non-planar sind und trianguliert diese.

Expand Face

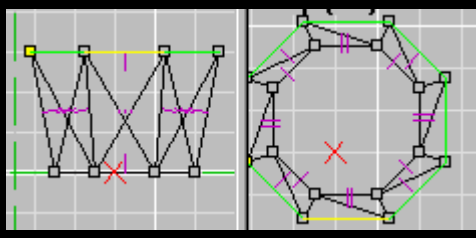


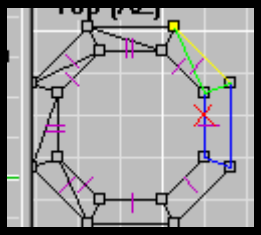
Vergrößert das current Face. Mach eins current und drücke den Button. Jeder Klick vergrößert es weiter, während die Proportionen gleich bleiben. Ganz gut, hat aber einen Nachteil, siehe Beispiel:



Wir beginnen mit dieser einfachen Säule hier. Mache das obere Face current.

Nach zweimaligem Klicken bekommst Du das hier. Alle Seiten, welche am vergrößterten Face anliegen wurden gesplittet. Man denkt, weil sie non-planar waren. Manchmal stimmt das, aber hier nicht.

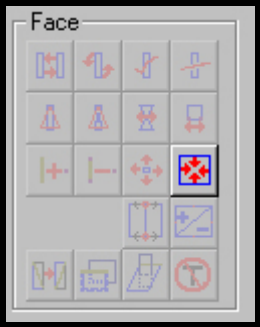




Du mußt zurückgehen und Combine Faces verwenden um sie wieder zu verbinden, um die Face-Anzahl nieder zu halten.

Update für v40:
Mittlerweile werden bei Expand und Contract Face die anliegenden Faces überwiegend nicht mehr gesplittet.

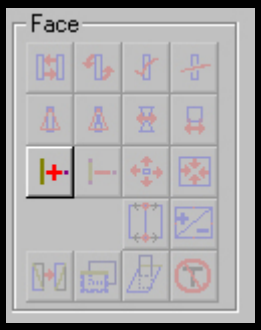
Contract Face



Kontrahiert das current Face unter Beibehaltung seiner Proportionen. Es hat die selbe Tendenz Faces zu splitten ob sie es bräuchten oder nicht, also aufpassen.

Expand und Contract funktionieren auch mit vollkommen fehlerhaften Faces, siehe dazu 129 - Räume teilweise skalieren.

Add Vertex

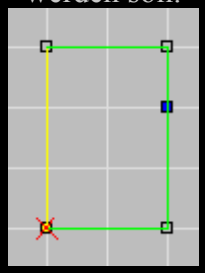


benötigt: genau ein markiertes Vert und ein current Face

Fügt ein markiertes Vert zur current Edge hinzu. Es ist gut um T-Joints zu fixen wie in diesem Beispiel:

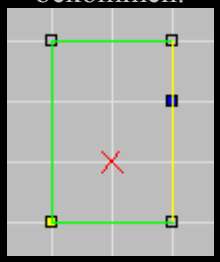
1. Schritt

Markiere das Vert das zum Face hinzugefügt werden soll.



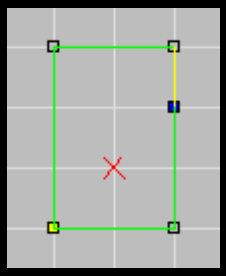
2. Schritt

Das Face current machen und **E** drücken um die Kante current zu bekommen.



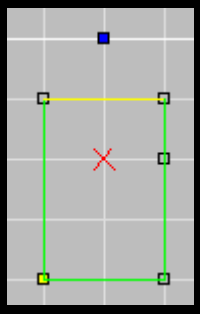
3. Schritt

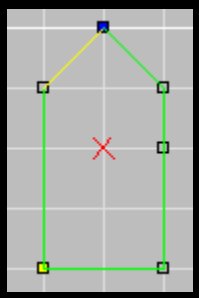
Tool ausführen.
Nun sind da fünf Kanten.



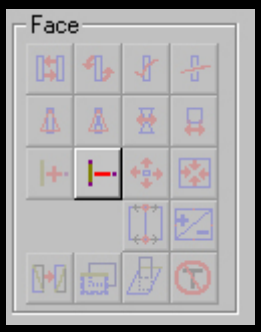
Der Vertex muß nicht an einer Kante des Faces sein. Du kannst das auch verwenden, um einem Face mehr Gestalt zu geben.

Sei aber vorsichtig. Das Tool prüft nicht auf non-planar oder Concavity.



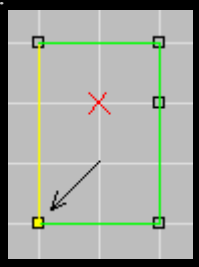


Remove Vert

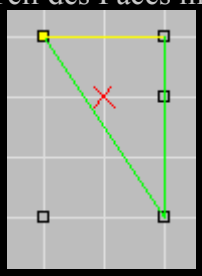


Dies entfernt den current Vert aus dem current Face.

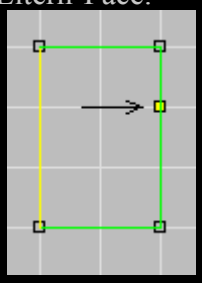
1) Mache das Face current und drücke **V** bis der Vertex current ist der entfernt werden soll.



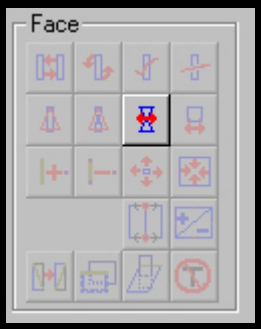
2) Führe das Tool aus und Du bekommst sowas wie unten. Beachte, daß der Vert noch immer da ist, nur kein Teil des Faces mehr.



3) Es kann auch zum T-Joint-fixen verwendet werden, durch Entfernten des ungebrauchten Verts vom Eltern-Face.

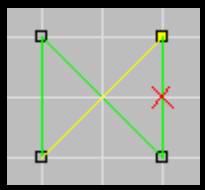


Twist Face

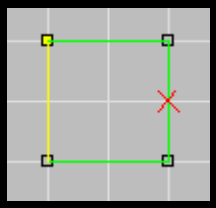


Repariert ein verdrehtes Face, das entsteht wenn ein Face erzeugt wird, dessen Verts nicht in einer fortlaufenden Reihenfolge eingefügt wurden. Meistens passiert das wenn man ein Face erzeugt und es an bereits Existierende anhängt. Für Faces mit fünf oder mehr Verts wirst Du 'Swap Face Verts' verwenden müssen.

Mache das Face current,



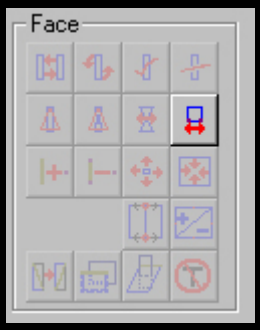
Button klicken gibt dann das:



¡Beachte!

1. Prüfe wohin die Normal danach zeigt.
2. Resette die Texturen zu Default UV's. Die sind sonst verzerrt.
3. Es prüft nicht auf Planarität – Du mußt.
4. Es prüft nicht auf Konkavität – Du mußt.

Swap Face Verts



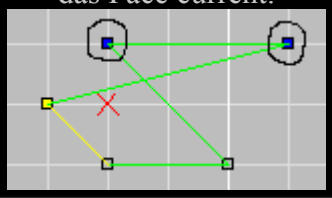
benötigt: zwei markierte Verts in der selben Edge

Um verdrehte Faces mit fünf oder mehr Vertices zu reparieren. Es vertauscht die zwei markierten Verts miteinander. Ich bevorzuge, Faces zu machen mit nicht mehr als vier Verts und die dann zu joinen, um das zu vermeiden. Mit fünf Verts ist es einfach; mit sechs oder mehr ist es möglich mehr als einen Dreher zu haben. Du bist besser dran wenn Du diese Situation vermeidest, so möglich.

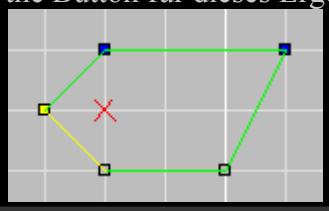
¡Beachte!

1. Prüfe wohin die Normal danach zeigt.
2. Resette die Texturen zu Default UV's. Die sind sonst verzerrt.
3. Es prüft nicht auf Planarität, Du mußt das machen.
4. Es prüft nicht auf Konkavität, Du mußt auch das machen.

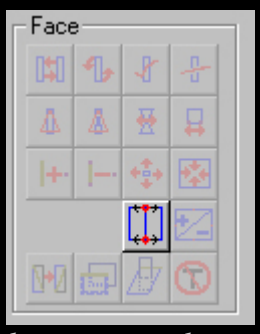
Markiere als Erstes die Verts die vertauscht werden sollen. Dann mache das Face current.



Press the Button für dieses Ergebnis:



Split Marked Faces Verts

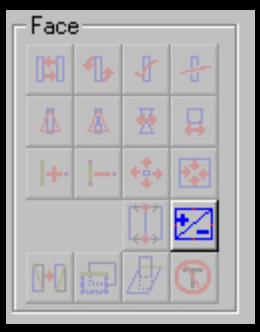


benötigt: markierte Faces

Ähnlich dem Button 'Split Current Vert', nur daß hier alle Vertices eines gegebenen markierten Face-Verbands gesplittet werden.

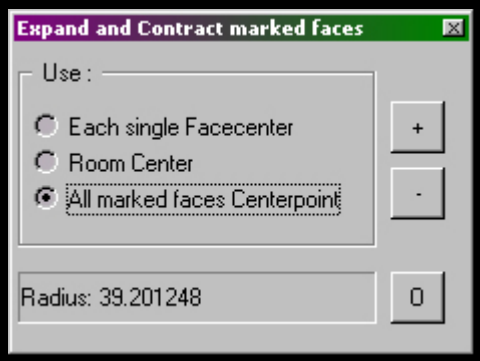
Etwas schwierig mit Screenshot darzustellen ☺

Expand / Contract Marked



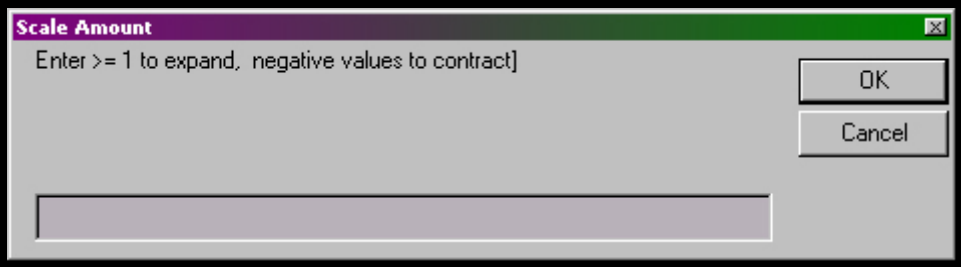
benötigt: markierte Faces

Diese Funktion ist noch recht neu und hat ihre Eigenheiten.
Sie ist dafür da, markierte Face-Gruppen zu verkleinern bzw. zu vergrößern. Ein klick auf den Button **O** läßt die direkte Eingabe von Faktoren zu.

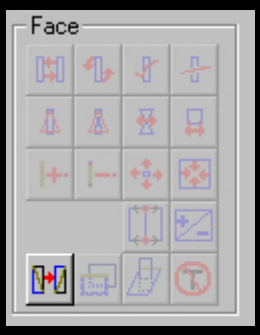


Ein Face-Verband wird allerdings auseinandergerissen, es muß danach Remove Extra Verts ausgeführt werden.

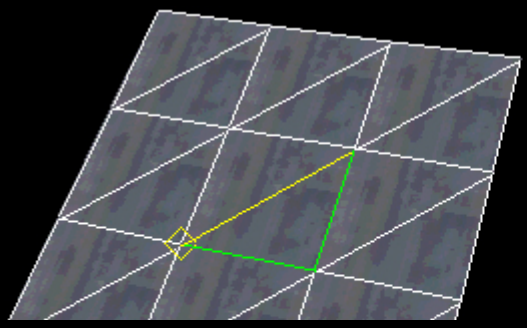
Update:
Ab v40 sieht der Dialog etwas anders aus, die dritte Option (All marked faces Centerpoint) gibt es nicht mehr.

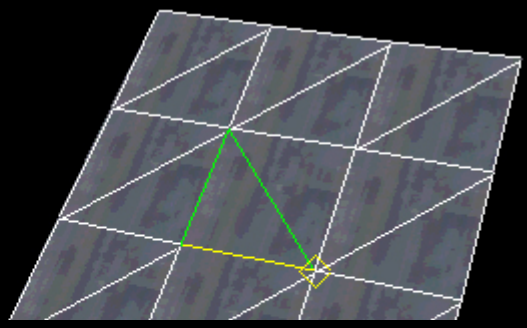


Turn Triangular Faces

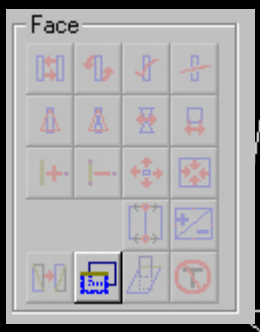


Ändert den 'Anschlag' von zwei dreieckigen Faces an deren gemeinsamer Edge. Diese muß current sein.

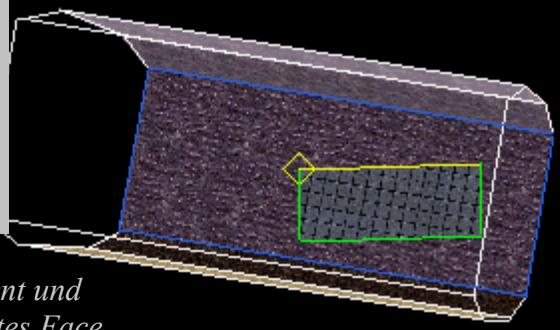


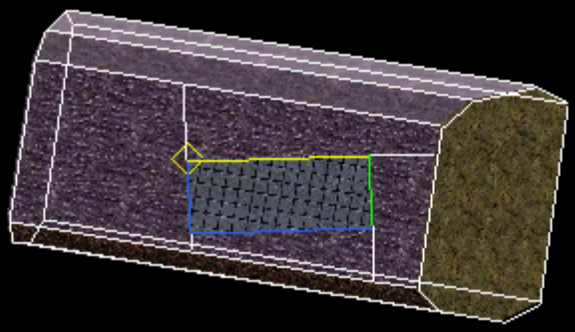


Clip Faces



Diese Funktion 'stanzt' ein Face gegen ein Anderes. Das markierte Face übernimmt hier die Rolle des 'Keksteiges', das current Face die Rolle der 'Keksform'.

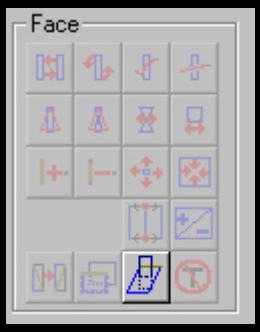




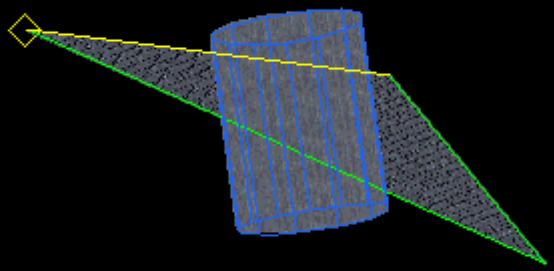
benötigt: ein current und genau ein markiertes Face

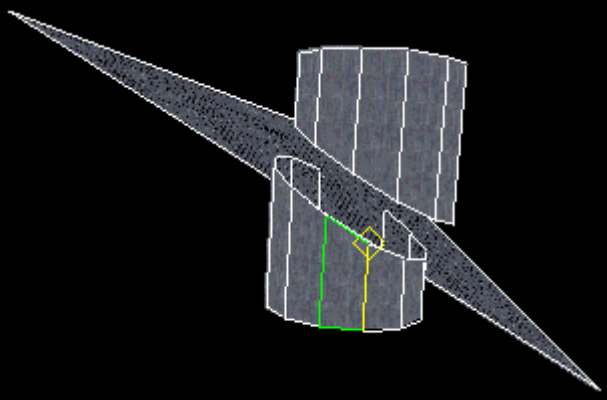
Wichtig ist hier, daß die beiden Faces planparallel sein müssen.

Split Faces at Plane



Eine der Tollsten Funktionen überhaupt - ein Faceverband wird mit einem anderen Face geschnitten:

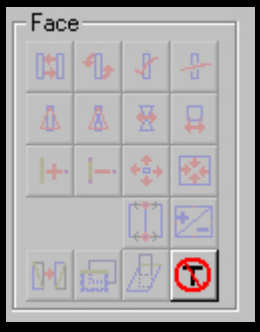




benötigt: zumindest ein markiertes Face und ein current

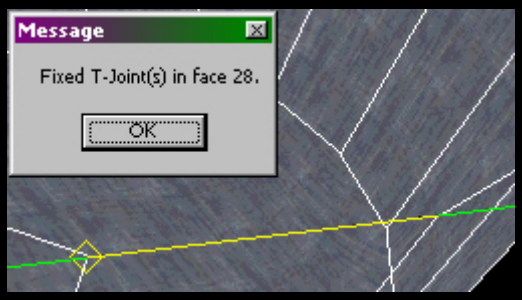
Markierte Faces sind das 'Schnittgut', das current Face übernimmt die Rolle der 'Klinge'

Kill T-Joint



Beseitigt T-Joints in Faces.

Face current machen, Button klicken -> dem current Face werden Verts hinzugefügt, bis keine T's mehr vorliegen.



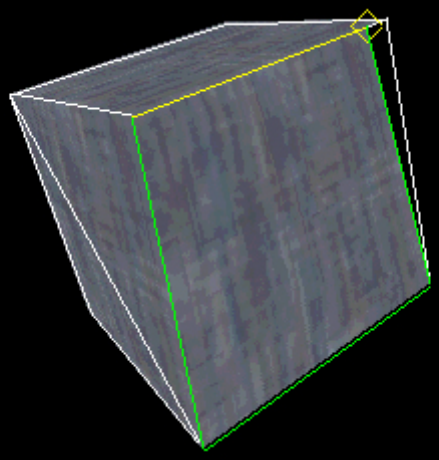
Edge:

In der Edge-Box gibt es nur drei Buttons, aber sie seien hier dennoch beschrieben:

Expand / Contract current edge

Diese Buttons verlängern bzw. verkürzen eine current Edge.

Wenn dabei Faces non-planar werden, splitten diese auf.

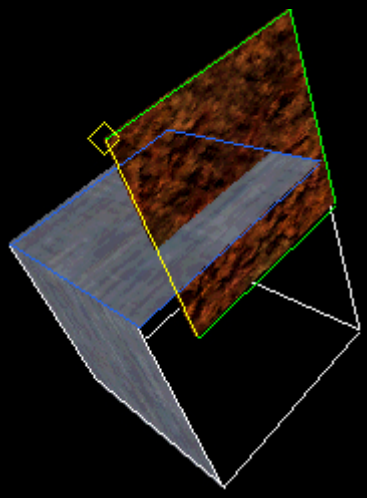


Intersect current edge whith marked Face

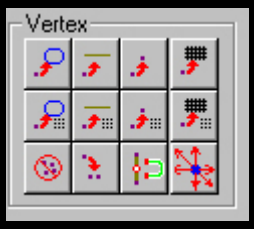
benötigt: genau ein markiertes Face und eine current Edge

Diese Funktion bestimmt den Schnittpunkt einer gegebenen current Edge mit einem markierten Face und setzt dort einen Vertex in die current Edge.

Links die Ausgangssituation, rechts der eingefügte Vertex.



Vertex:



In der Vertex-Box findest Du unter Anderem die ganzen ‚Snap'-Tools. Diese Tools sind sehr mächtig. Meistere sie, und das Bauen von komplexen Strukturen wird Dir zum Vergnügen anstatt zur Sisyphusarbeit. Verwende sie für die Präzision, um reparierte Faces und Bad Shells auf das Grid zurückzukriegen.

Die Snap-Tools bewegen den oder die ausgewählten Verts zu einem definierten Ziel (Kante, Vertex, Gitterpunkt oder Ebene eines Faces). Sie hängen sie nicht an, also mußt Du immer noch ‚Remove Extra Verts' oder ‚Add To Current Edge' verwenden (je nachdem welches passender erscheint).

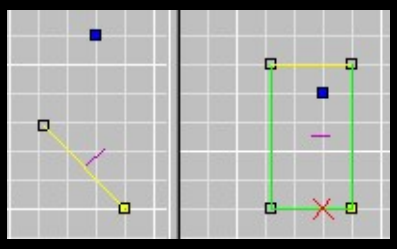
Die ersten vier bewegen das oder die markierten Verts zum Ziel. Die zweiten vier bewegen das current Vert zum Ziel und nehmen alle markierten Verts mit.

Snap Marked Vert To Plane Of Current Face

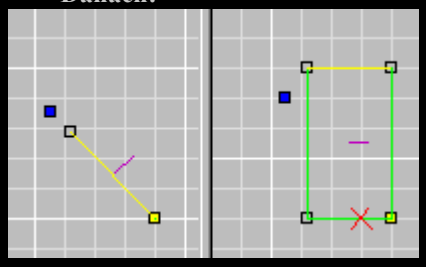
Wenn Du auf das Tool zeigst meint es ‚Snap marked to Face', aber lies immer die ganze Beschreibung in der Statuszeile. Es schnappt zur EBENE des Faces. Es kann treffen, muß aber nicht; siehe Beispiele.

Es ist gut für die Genauigkeit, aber auch praktisch um non-planare Faces plan zu kriegen.
Markiere den oder die Verts, und mache das Face auf dessen Ebene Du sie haben willst current. Button klicken.
Der/die Vert/s bewegen sich lotrecht zum current Face.

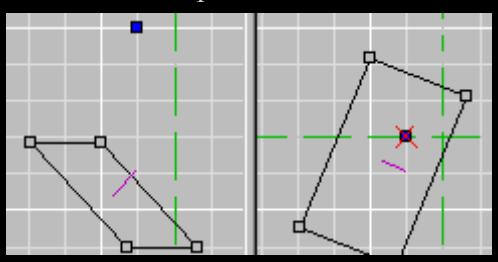
Vorher:

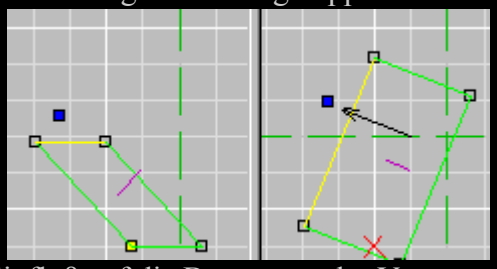


Danach:



In diesem Beispiel habe ich Das Face in Richtung des Verts gekippt. Dadurch





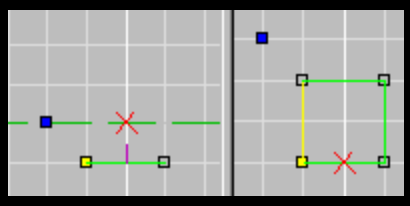
kannst Du sehen daß das Grid keinen Einfluß auf die Bewegung der Verts hat. Deswegen empfehle ich, daß Du immer speicherst bevor Du dieses Tool anwendest. Die Resultate sind nicht immer das was man erwartet.

Snap Vert To Line Of Current Face

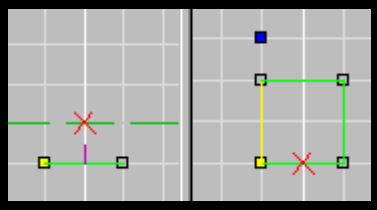
Dies bewegt den oder die markierten Vert/s zur Linie der current Edge, nicht notwendigerweise zur Kante selbst. Stelle Dir vor, die Kante reicht beiderseits bis in die Unendlichkeit; der/die Verts bewegen sich lotrecht zur Linie der Kante.

Markiere den/die zu bewegenden Vert/s. Mache das Face current und drücke **E** um die gewünschte Kante current zu machen. Button klicken.

Vorher

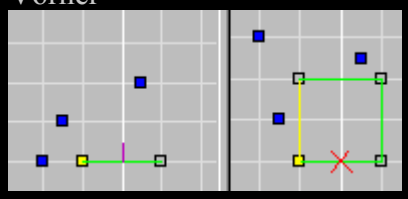


Nachher

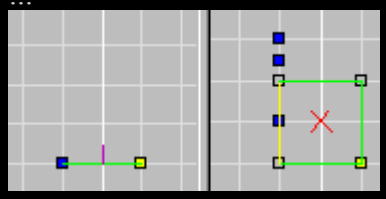


Wenn Du mehrere Verts bewegst, nimmt jedes seinen eigenen Weg, der nicht in Relation zu den Anderen steht.

Vorher



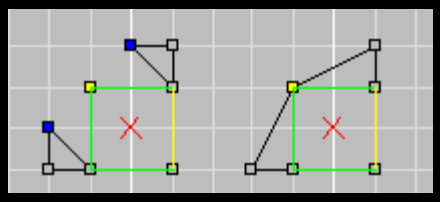
...



Snap Marked Vert(s) To Current Vert

Markiere Vertices die Du bewegen willst. Mache das Ziel-Face current. Verwende **V** um das Target-Vertex current zu machen. Button klicken.

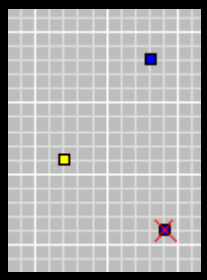
Im Beispiel kannst Du sehen, daß dies ein großartiger Weg sein kann, um Bad Shells zu reparieren (Links Vorher, Rechts danach).



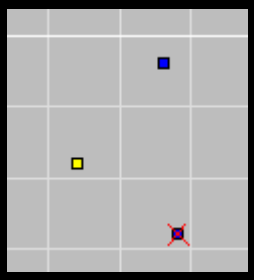
Snap Marked Vert To Grid

Verschiebt die markierten Vert/s zum nächsten *aktiven* Gitterpunkt. Gut um wieder auf das Gitter zu kommen, aber ich empfehle das nur einmal zu machen um die Kontrolle zu behalten. Siehe unten.

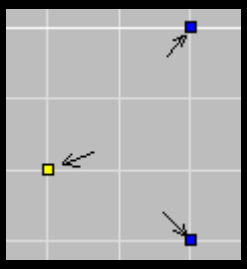
Ein paar eingesetzte Verts mit der Gittergröße 1.



Gridsize auf 10 geändert.



Tool anwenden. Hier siehst Du wie sie sich zum nächsten Gitterpunkt bewegen.



Die nächsten Tools fragen Dich, welches Face, welche Kante oder welches Vert Du als target (=Ziel) haben willst. Daher mußt Du die Informationen verstehen, welche in der Statuszeile stehen:

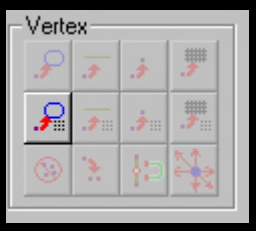
R: 45/56 F: 6/42 P: 0/3 E: 0/4 V (idx): 0/4 (11/48)	Grid: 1	2048, 250, 2430	0, 0, 0	Face mode	Marked: V: 0 F: 0

D3Edit beginnt Enumerationen mit 0; vier Faces werden von 0 bis 3 nummeriert.

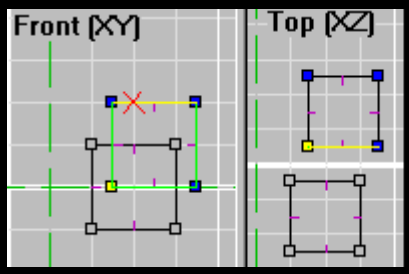
R: 45/56	Raum #45 von 56 ist current	Grid: 1	Gittergröße ist 1
F: 6/42	Face #6 von 42 ist current	2058, 250,2430	Gitterpunkt, wo die Maus gerade ist
P: 0/3	Portal #0 von 3 ist current	0,0,0	Aktueller Gitterpunkt (red **X**)
E: 0/4	Edge #0 von 4 (des current Faces) ist current	Face mode	Modusanzeige: face, vert, object
V:(idx) 0/4(11/48)	Vert # 0 des current Faces Vert # 11 des current Raums ist current	Marked V:0 F:0	Wieviele Verts und Faces markiert sind.

Du solltest Marked V:0 F:0 unter Beobachtung halten; es ist ein guter Weg sicherzustellen daß Du nicht versehentlich etwas markiert hast.

Snap Current Vert To Specified Face



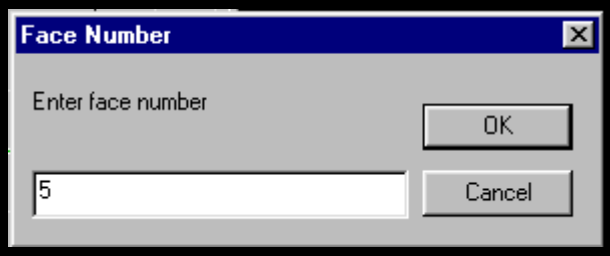
Dies ‚schnappt' das Current Vert zum spezifizierten Face unter Mitnahme aller markierten Verts. In Diesem Beispiel habe ich zwei Würfel in einem Raum.

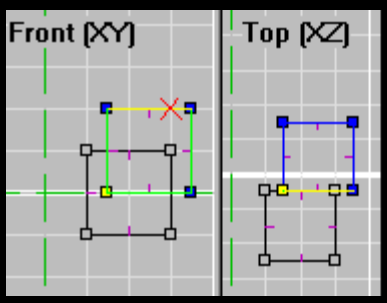


Bevor Du Verts markierst und eins Current machst, Finde die Nummer des Faces heraus zu dem Du ‚schnappen' willst.

Ich markiere alle Verts des Würfels, den ich bewegen will. Mache ein Face Current, dessen Vert Du current haben willst, welches ist egal. Dann verwende **V** um das Vert current zu bekommen.

Wende das Tool an. Du wirst dann nach der Face-Nummer gefragt, wohin Du die Verts bewegen willst (Vergiß nicht, es wird zur Ebene des Faces bewegt, nicht zum Face selbst). Du wirst auch nach einer Kante gefragt, das scheint aber keinen Einfluß zu haben (verifiziert mit v039 und v40).





Klicke **OK** und schau was passiert.

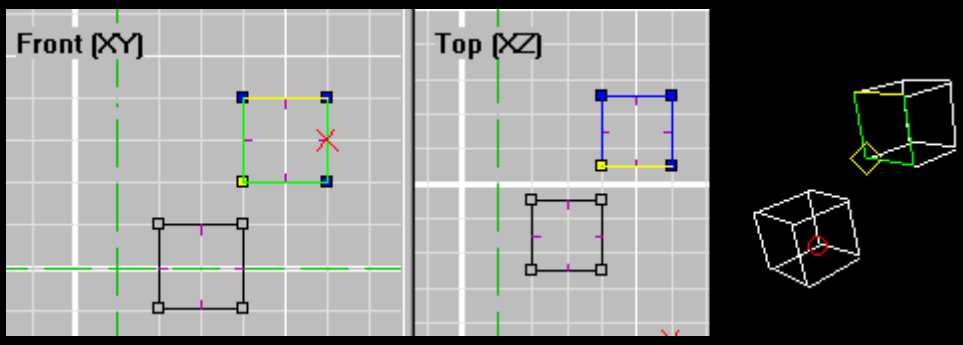
Snap Current Vert To Line Of A Specified Face

Für dieses Tool mußt Du die Nummer der Ziel-Kante und die Nummer des Faces, zu dem diese Kante gehört, wissen. Das Tool bewegt zur imaginären Linie (unter Mitnahme aller markierten Verts) dieser Kante, nicht notwendigerweise zur Kante selbst. Vergiß nicht, daß keines der Verts automatisch verbunden wird, sollte die Kante getroffen werden; das mußt Du schon selbst machen.

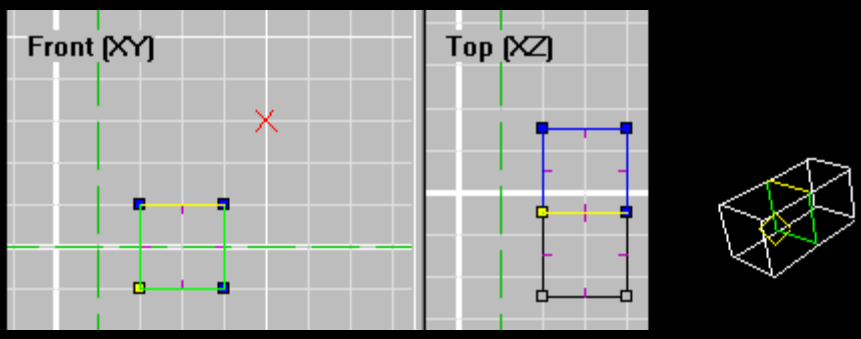
Snap Current Vert To Specified Vert

Möglicherweise wirst Du dieses Tool am häufigsten einsetzen. Um es zu verwenden mußt Du die Nummer des Ziel-Verts relativ zum Raum (nicht relativ zu seinem Face) wissen.

Markiere alle Verts die du bewegen möchtest.
Mach eins current.



Ich habe hier das Target-Vert eingekreist. Das wird die Würfel in eine Reihe bringen. Wähle das Tool an und erkläre ihm zu welchem Vert du das Current Vert bewegen willst. Vergiß nicht, sie werden nicht zusammengehängt bevor Du Remove Extra Verts ausführst.



Snap Current Vert To Grid Point

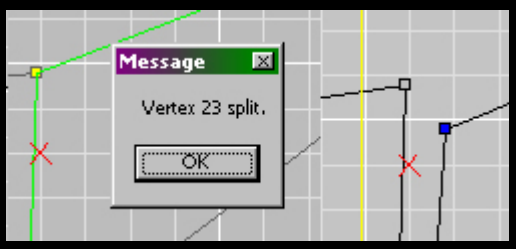
Bewegt das Current Vert zum nächsten Gitterpunkt unter Mitnahme aller markierten Verts. Das Ziel ist dann der nächste aktive (=sichtbare) Gitterpunkt der aktiven 2d-Ansicht (Top, Front oder Side)

Remove Extra Verts

Dürfte sich selbst erklären... entfernt überflüssige Vertices.
Wird diese Funktion ausgeführt, werden Konstrukte gewissermaßen 'verschweißt'.

Split Current Vertex

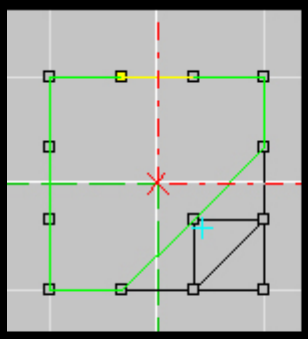
Diese Funktion tut das genaue Gegenteil: ein current Vertex wird in mehrere aufgespalten; damit kann man ein Face-Konstrukt quasi 'auftrennen':

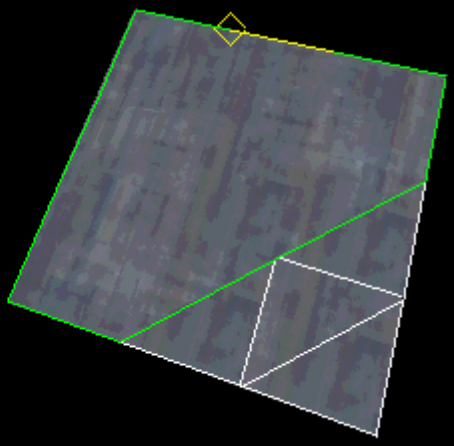


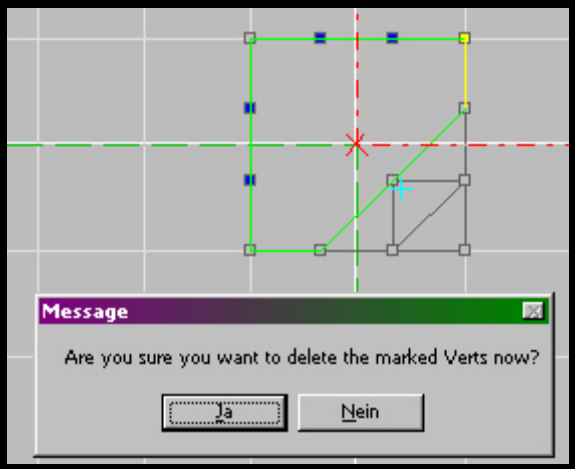
Links: vor Split,
Rechts: nach Split, ein Vert verschoben

Remove Unused Verts from Edges

Neu seit v40. Diese Funktion entfernt unbenötigte Vertices aus ansonsten geraden Edges:
Hier wurden Faces vereinigt, wobei Extra Vertices in den Edges übgriggeblieben sind.

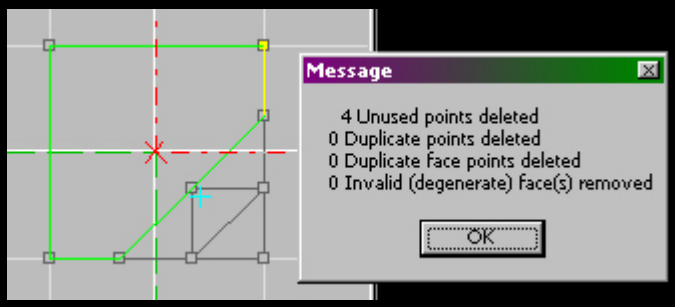






Führt man die Funktion aus, werden die nicht gebrauchten Verts markiert.

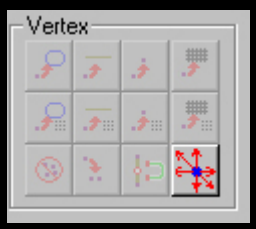
Bestätigt man das mit Ja, werden die Verts gelöscht und ein kurzer Bericht angezeigt.



Diese Funktion sorgt für Sauberkeit im Level

Auch seit v40 gibt es hier eine Neuerung, in der Statusleiste kann die 'Stärke' des Anti-Edge-Vert-Magneten bestimmen:

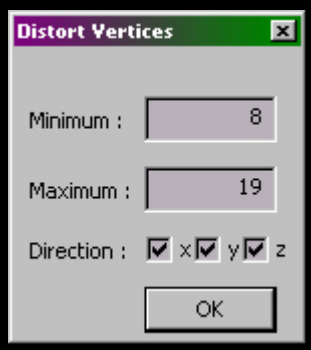
In der Statuszeile stehen rechts von Marked V:0 F:0 nun zusätzlich noch Conc: was die Toleranz für Konkavität anzeigt, und PtE: welcher der Parameter für die Funktion ist. Ein **Lklick** darauf öffnet ein Dropdown-Listenfeld, aus dem ein Wert gewählt werden kann. Dieser Wert bestimmt, wie weit ein Vert von einer gegebenen Edge entfernt sein muß, um als 'Ecke' zu gelten.
Der Wertebereich geht von 0.000001 bis 0.0025, Standardwert ist 0.000250. Dieser ist meist ausreichend, es kann aber passieren daß man durch fortgesetzte Bearbeitung der Geometrie Verts erzeugt hat, welche sich weiter von der idealen Kante entfernt befinden; dann hilft es hier den Wert zu erhöhen.
Hilft auch das nicht mehr, muß man es wieder händisch machen.



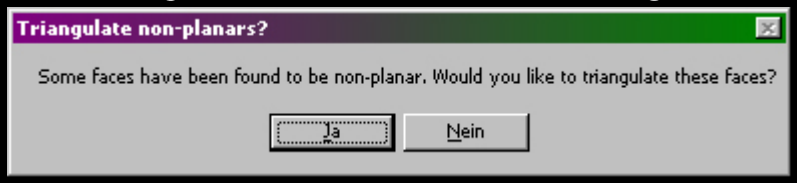
benötigt: ein oder mehrere markierte Verts

Ganz brandneu ist auch dieser Button: der Vertex-Shaker.

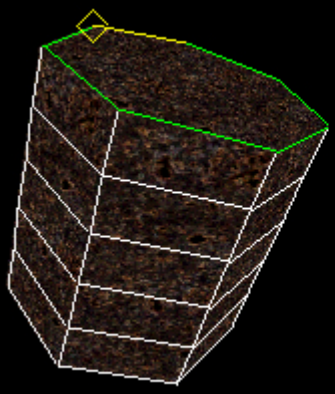
Diese Funktion erlaubt es, markierte Verts um einen zufälligen Betrag in beliebige Richtungen zu verschieben. Es poppt dieser Dialog auf:

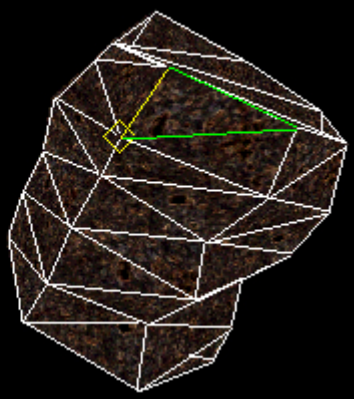


Unter Direction kann frei gewählt werden.
Sollten sich nicht-valide Faces ergeben wird man gefragt ob man diese splitten möchte. Damit wird das Erzeugen von unregelmäßigen Formen ein Kinderspiel!



Anwendungsbeispiel auf einen siebenseitigen Zylinder; links vorher - rechts danach:



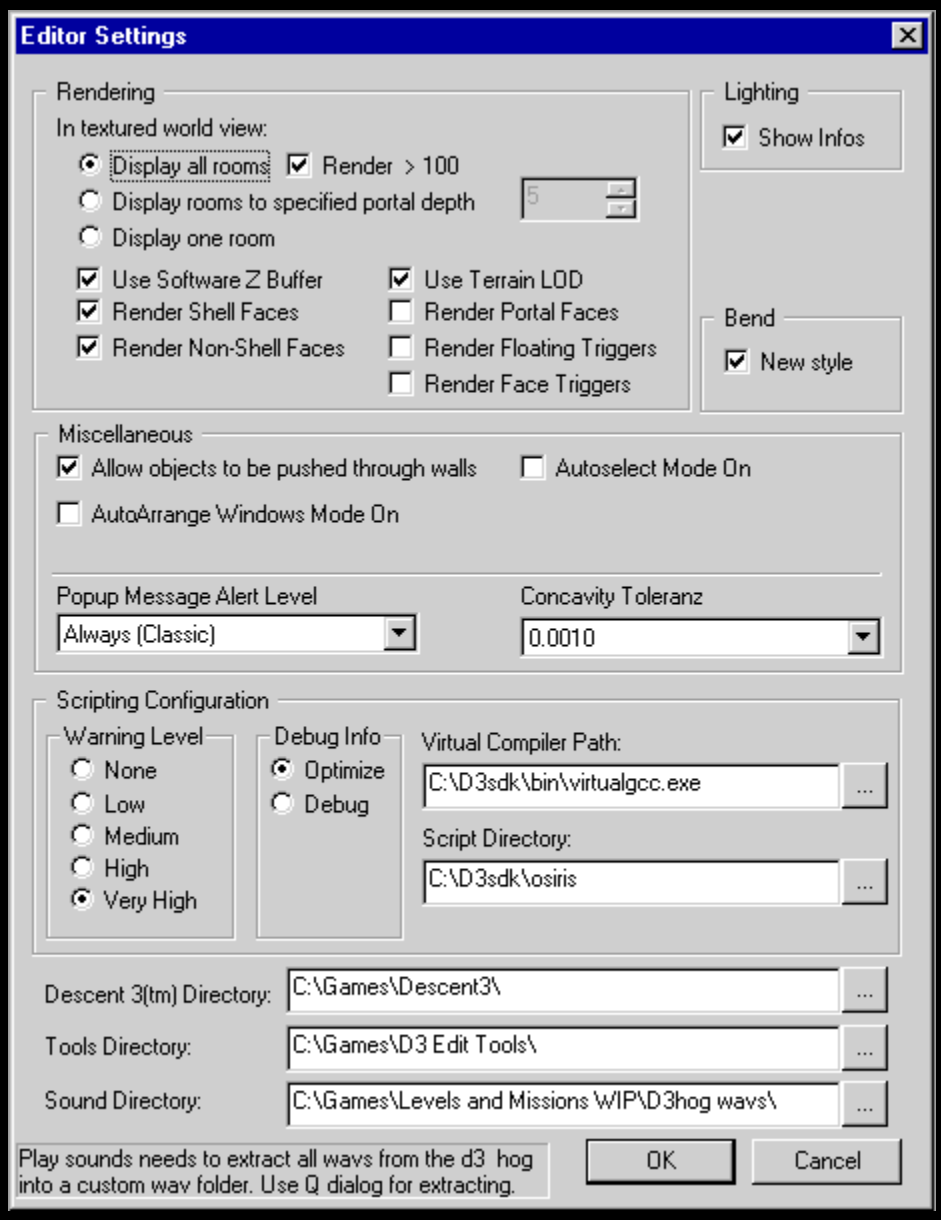


021 - Quicktest mit D3Edit einrichten

WillyP

D3Edit

Viele neue Features sind seit dem ersten Release von D3Edit hinzugekommen. Ich empfehle wärmstens, daß Du die neueste Version downloadest. Der beste Platz um sie zu bekomment ist www.descentforum.de/forum/, dort in der ‚**English Embassy**'-Sektion gehe in das Thema ‚**Wichtig: D3 Editing tools**'. Jaja, es meint ‚beta', ignoriere das, greif's Dir einfach, es ist um Längen besser als die Originalversion die mit ‚Mercenaries' mitgekommen ist.



Ok, jetzt hast Du einen Editor der meistens *nicht* abstürzt anstatt einem der meistens *doch* krepiert. Und die zusätzlichen Features sind wertvoll. Wieauchimmer, es gibt noch immer kein Autosave und Undo ist auf ein paar wenige Bereiche beschränkt. Der Ratschlag oft zu speichern und Backups zu machen ist noch immer gültig!

Nachdem Du ihn entpackt hast, sind nur ein paar Sachen einzustellen bevor du mit dem Levelbauen anfangen kannst. Unter File gehe nach Settings, was ein Dialogfeld namens **Editor Settings** öffnet. Fürs erste müssen wir nur das Descent 3(tm) Directory setzen, und zwar dorthin wo Du Dein D3 hininstlliert hast; der Standardpfad ist `C:\Games\Descent3\`.

Optional kannst Du auch Tools Directory setzen, um aus D3Edit heraus mit externen Tools und Dateien arbeiten zu können. Danach gehe nach Tools->Customize->Add und surfe zu Deiner Tool-exe, der `.gam`-Datei Deines Levels, deinem Lieblingstutorial (zum Beispiel dieser Datei ☺), kurz, jedem Dateityp dem eine entsprechende Anwendung zugeordnet ist. Du kannst sogar eine Linkseite in html erstellen, welche die Links zu Deinen favorisierten Descent-Seiten enthält! (Du wirst ‚All Files' auswählen müssen, um auch andere Dateien als nur die Ausführbaren zu sehen.). Klicke auf Open->OK. Wenn Du Dir nicht sicher bist welche Datei dein Tool repräsentiert, schau unter den Eigenschaften seiner Verknüpfung nach.

Du kannst jetzt beginnen Levels zu bauen, aber... öffne Dein Level noch nicht, ich schlage vor daß Du zuerst Quicktest einrichtest.

Quicktest

1. die Quicktest Toolbox: Fülle alle Felder aus.

Nun können wir Quicktest einrichten, ein gutes Tool das mit den neuesten Versionen von D3Edit mitkommt und aus dem Editor gestartet wird. Öffne es mit dem Button **Q** aus der Toolbar. Als erstes erstelle ein neues Profil: Klicke New und vergib einen Namen.

Der Dateiname der Mission kommt unter MN3 Name. Der Name, welcher in der Missionsauswahl von Descent3 sichtbar ist, kommt unter Mission. Wenn Du zusätzliche Dateien hast, wie etwa Texturen, Objekte, etc. kannst Du sie mit **Insert >>** laden. Du kannst unter einzelnen Dateien, `.hog`-Dateien und Inhalten ganzer Ordner wählen.

2. Data: Lade Table-Dateien und Missions-hog (überarbeiten)

Klicke Data->Tablefile Manager. Die Base Tablefile sollte `Table.gam` sein. Das ermöglicht Dir, die Original-D3-Texturen, -Objekte und andere Goodies in Deinem Level zu verwenden. Wenn Du eine Custom `.gam` für Deine Mission hast, lade sie als Mission Tablefile. Wenn Du *keine* hast, stelle sicher daß da auch keine steht. Falls Du bereits eine `.mn3` für dieses Level erzeugt hast, kannst Du diese nun laden: Data->Configure a Mission Hog->Mission Hog File:. Wenn Du *noch keine* Missions-`.hog` gespeichert hast, dann laß' das derweilen leer.

3. Öffne einen existierenden Level oder erzeuge einen Neuen.

Jetzt kanns Du Dein Level öffnen bzw. neu bauen. Speichere den Level mit einem kurzen Dateinamen. Nur (am besten Klein-)Buchstaben und Ziffern, ja?

4. Erzeuge eine Missions-.hog mit Quicktest

Zurück in Quicktest stelle sicher daß Generate MSN und SP angetickt sind, dann klicke Anarchy an und dann kannst Du Schlüsselwörter aus der Liste wählen, es sei denn der Level soll nur Single Player sein. Speichere Deinen Level und klicke Save MN3.

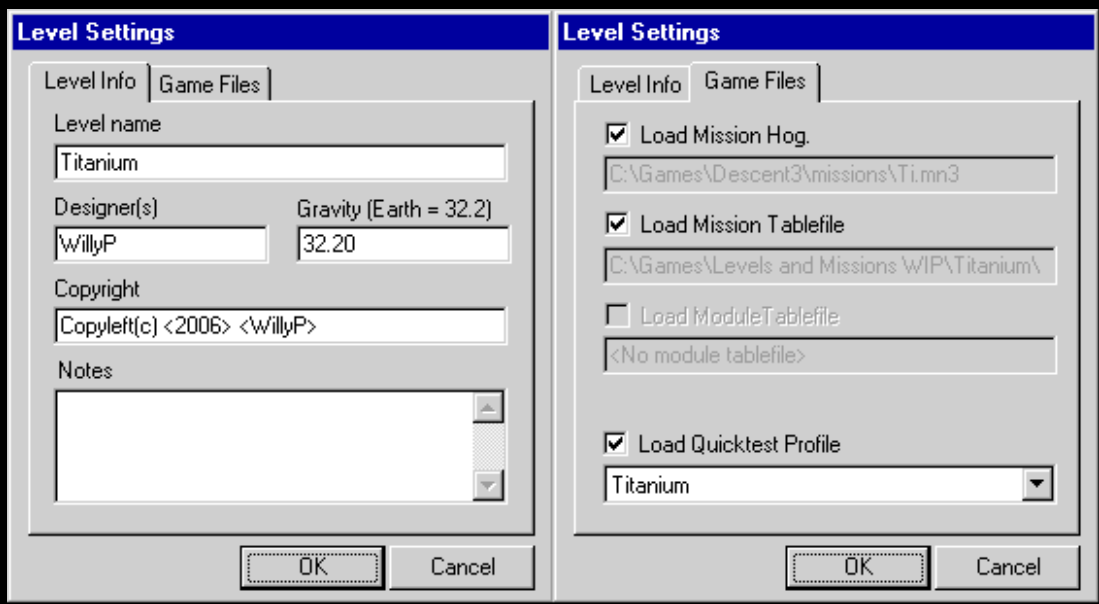
Nun hast Du eine Mission erzeugt! Wir sind fast fertig, nur noch ein paar Schritte.

5. Lade Deine neue `.mn3` als Mission Hog.

Wenn es eine neue Mission ist, mußt Du Deine `.mn3` vorher als Mission `.hog` laden. Als erstes schließe Deinen Level. Gehe nach Data->Configure a Mission Hog. Drücke den Button **Change**. Browse zu Deinem Descent-Missionsverzeichnis und wähle die `.mn3` die Du grade erzeugt hast und klicke **OK**. Wenn Du eine existierende `.mn3` in Schritt 2 geladen hast, überspringe das.

6. Öffne Dein Level, öffne die Level Settings.

Jetzt kannst Du Dein Level wieder öffnen. Wähle die World View. Öffne Level Settings unter Files und fülle die Felder unter der Level Info-Registerkarte aus. In der Game Files-Registerkarte kannst Du nun Load Misison Hog anhaken (hake auch Load Mission Tablefile an, sofern Du eine hast), dann hake Load Quicktest Profile an und wähle Dein Profil aus der Dropdown-Liste (Alles im **Level Settings-Dialog**). Das sind jetzt vielleicht viele Schritte, aber normalerweise muß man das nur einmal machen.



Testfliege Dein Level!
Wenn Du den Level Testen willst, stelle als erstes sicher daß Du es gespeichert hast und dann drücke den **Quicktest!**-Button. Quicktest updatet dann Deine `.mn3` und startet Descent3. Wähle Deinen Level aus der Liste (siehe dazu im Anhang: ingame testhilfen, feld machen!,haltmarke').

Custom-Elemente mit Quicktest laden
Wenn Du benutzerdefinierte (*offizielle Übersetzung von ‚custom', wirklich*) Texturen, Musik, Roboter, etc. in Deinem Level haben möchtest, müssen diese in die .mn3 inkludiert werden. Klicke den Button **Insert >>**, Du kannst sie dann einzeln laden oder gleich einen ganzen Ordner. (Das Einfügen einer neuen Datei mit gleichem Namen wird die alte überschreiben; aber Vorsicht, Du bekommst dann diesbezüglich keine Warnung angezeigt!) Hast Du deine Dateien eingefügt und **Save MN3** geklickt, schließe Deinen Level und öffne ihn erneut um die geupdatete `.mn3` zu laden.

Updaten Deiner Custom-Dateien
Von Zeit zu Zeit wirst Du während des Bauens deiner Levels in den Dateien, die Du in die `.mn3` eingefügt hast Änderungen vornehmen wollen. Zum Beispiel, wenn Du noch einen Eintrag in die `.gam`-Datei gemacht hast, oder die Animation einer Tür verändert hast. Deine Dateien müssen nicht erneut eingefügt werden, nachdem Du daran Änderungen vorgenommen hast. Speichere einfach deine `.gam`-Datei mit demselben Namen, oder drücke Speichern in DALLAS um eine geupdatete `.dll` zu kompilieren, etc... Wenn Du sie zuvor eingefügt hast, werden sie geupdatet! Für Texturen und Objekte speichere die `.mn3` und schließe-öffne Dein Level, um diese Einräge in den entsprechenden Listenfeldern zu sehen (*Beachte: Du wirst Deine* `.tga`-*Bilder in eine* `.ogf`-*Texturdatei und einen* `.orf`-*Raum zu einem* `.oof`-*Objekt konvertieren müssen, nachdem Du sie verändert hast.*)

022 - Editoreinstellungen und -profile sichern

(LL)Dark

Leider hat der D3Editor die unangenehme Eigenschaft alle seine Einstellungen in der Registry zu speichern. Dabei geht es leider nicht nur um die Einstellungen der Bedienoberfläche und um die paar Pfadangaben in den Settings, sondern auch um die vom User angelegten Quicktestprofile. Diese Profile können - je nach Level - sehr umfangreich sein. Bei größeren Singleplayerprojekten können hier durchaus hunderte Dateien enthalten sein.

Wenn es nun zu einer Neuinstallation von Windows kommt, wird eine neue Registry angelegt und alle Editorinstellungen und Profile sind verloren.
Abhilfe schafft nur die Sicherung der entsprechenden Registryzweige.
Bei einer bewußten Neuinstallation von Windows lässt sich das natürlich auch von Hand durchführen, aber wer denkt schon immer an solche Nebensächlichkeiten und bei einem Defekt der Registry ist oft sowieso nichts mehr zu retten.

Nach Rücksprache mit Atan sind wir zu dem Schluß gekommen, daß es zu aufwendig wäre eine Sicherungsfunktion in den Editor einzubauen da diese einfach zu selten benötigt wird.
Andere Aufgaben sind nun mal wichtiger.
Daher habe ich mal schnell eine Batchdatei zusammengebaut die diese Aufgabe bei jeder Beendigung des Editors automatisch durchführt.
Voraussetzung ist natürlich das man den Editor immer über diese Batchdatei startet.
Zugegeben – Batchdateien (auch Stapelverarbeitung genannt) sind ein Relikt aus DOS-Zeiten und sehen wegen fehlender grafischer Oberfläche auch nicht schön aus, aber sie können einiges und sind auch für programmiertechnisch unbedarfte User relativ einfach zu erstellen.
Für unseren Zweck also ideal 😊

Schritt für Schritt:

1. Öffnet den Ordner in dem eure `d3edita.exe` liegt.

2. Klickt mit der *rechten* Maustaste ins Fenster und wählt im Kontextmenü unter Neu-> Textdokument
   Es entsteht eine neue Datei mit dem Namen `Neu Textdokument.txt`

3. Benennt diese Datei in D3-Editor-`START.bat` oder `D3-Editor-START.cmd` um.
   Den daraufhin erscheinenden Dialog beantwortet ihr mit *Ja*.

4. Nun wieder mit der *rechten* Maustaste auf die Datei klicken und im Kontextmenü Bearbeiten auswählen. Daraufhin öffnet sich ein Texteingabefenster.

5. Hier gebt ihr nun folgenden Text ein:

```
echo off
if exist %EditorSettings-Save.reg goto 1
echo Keine gesicherten Editoreinstellungen gefunden
d3editAv.exe
regedit /e EditorSettings-Save.reg HKEY_Current_USER\SOFTWARE\Outrage
echo Editoreinstellungen wurden in der Datei EditorSettings-Save.reg gesichert
exit
:1
regedit /s EditorSettings-Save.reg
echo gesicherte Editoreinstellungen gefunden und in die Registry eingetragen
d3editAv.exe
regedit /e EditorSettings-Save.reg HKEY_Current_USER\SOFTWARE\Outrage
echo Editoreinstellungen wurden in der Datei EditorSettings-Save.reg gesichert
exit
```

6. Klickt nun auf Datei->Beenden und bestätigt die folgende Abfrage mit *Ja*.

7. Nun muss noch sichergestellt werden das ihr den Editor auch immer mit dieser Batchdatei startet. Legt dazu eine Verknüpfung der Datei auf dem Desktop an.
   Klickt mit der *rechten* Maustaste auf die Verknüpfung und wählt im Kontextdialog Eigenschaften.
   In der Karteikarte *Verknüpfung* stellt ihr unter Ausführen->Minimiert ein.
   In der gleichen Karteikarte könnt ihr auch ein anderes Symbol für die Verknüpfung festlegen. (zB. das in der `d3edita.exe` mitgelieferte Icon für den Editor)

8. Eure ursprüngliche Verknüpfung für den Editor solltet ihr löschen, damit es nicht zu Verwechslungen kommt. Nur wenn ihr den Editor immer über diese Batchdatei startet werden eure aktuellen Einstellungen auch gesichert!

Das wars schon.

Startet den Editor von nun an nur noch über diese Verknüpfung.

Ihr werdet bemerken das nun nach dem Starten des Editors über die Batchdatei in der Taskleiste ein zweiter Programmeintrag zu sehen ist.

Wenn ihr draufklickt seht ihr ein DOS-Fenster mit einer Statusmeldung.

Dieses Programm ist die Batchdatei und ihr dürft es NICHT beenden !!!

Es beendet sich von selbst wenn ihr den Editor beendet und sichert vorher noch eure Editor-spezifischen Registryeinträge in der Datei `EditorSettings-Save.reg`.
Diese findet ihr im gleichen Verzeichnis wie die Batchdatei.
Bei jedem Start der Batchdatei werden die in dieser Datei gesicherten Registryeinträge wieder in die Registry eingetragen.

Nach einer Neuinstallation von Windows ist daher nichts weiter nötig als den Editor über die Batchdatei zu starten und schon funktioniert euer Programm wie immer.

Erklärung der Befehle:

`echo off`	es werden standardmäßig keine Meldungen ausgegeben
`if exist %EditorSettings-Save.reg goto 1`	wenn die Datei `EditorSettings-Save.reg` im Programmpfad der Batchdatei gefunden wird springe zur Sprungmarke 1, wenn nicht geht es hier weiter. (dieser Befehl dient nur für einen sauberen Programmablauf beim allerersten Start der Batchdatei oder falls die Datei `EditorSettings-Save.reg` aus anderen Gründen nicht vorhanden ist)
`echo Keine gesicherten Editoreinstellungen gefunden`	Diese Meldung wird im DOS-Fenster angezeigt
`d3editAv.exe`	Starten des Editors
`regedit /e EditorSettings-Save.reg HKEY_Current_USER\SOFTWARE\Outrage`	Wenn der Editor beendet wurde werden die Einträge dieses Registryzweiges in der Datei `EditorSettings-Save.reg` gespeichert.
`echo Editoreinstellungen wurden in der Datei EditorSettings-Save.reg gesichert`	Diese Meldung wird im DOS-Fenster angezeigt.
`exit`	Beenden des Batchprogramms
`:1`	Sprungmarke 1
`regedit /s EditorSettings-Save.reg`	die Einträge aus der Datei `EditorSettings-Save.reg` werden in die Registry zurückgeschrieben.
`echo gesicherte Editoreinstellungen gefunden und in die Registry eingetragen`	Diese Meldung wird im DOS-Fenster angezeigt
`d3editAv.exe`	Starten des Editors
`regedit /e EditorSettings-Save.reg HKEY_Current_USER\SOFTWARE\Outrage`	Wenn der Editor beendet wurde werden die Einträge dieses Registryzweiges in der Datei `EditorSettings-Save.reg` gespeichert.
`echo Editoreinstellungen wurden in der Datei EditorSettings-Save.reg gesichert`	Diese Meldung wird im DOS-Fenster angezeigt.
`exit`	Beenden des Batchprogramms

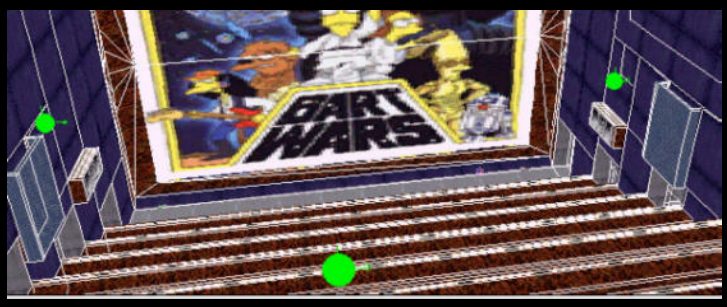
Selbstverständlich ist es auch möglich die Batchdatei woanders als im Editorverzeichnis abzulegen und auch die Registrysicherung kann in einem anderen Verzeichnis gespeichert werden oder auch gleich an mehreren Stellen.

Dann ist es nötig alle Dateien mit den entsprechenden Pfadangaben zu versehen.

023 - Standard - Mod's

[DCG]Roadrunner)

Ich möchte hier folgende Spielmod- Einstellungen ,an dem Beispiel '**Movieworld**' ,für den Leveleditor erklären.



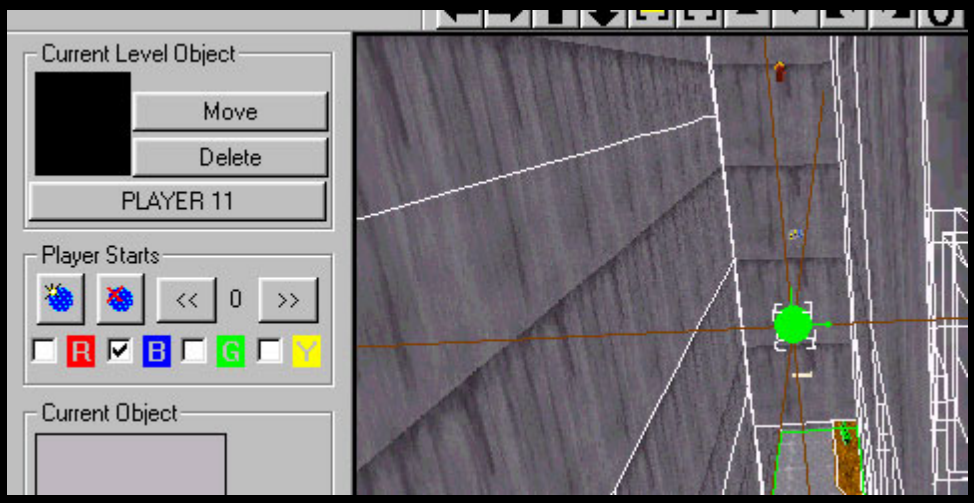
Hoard
CTF
Entropy

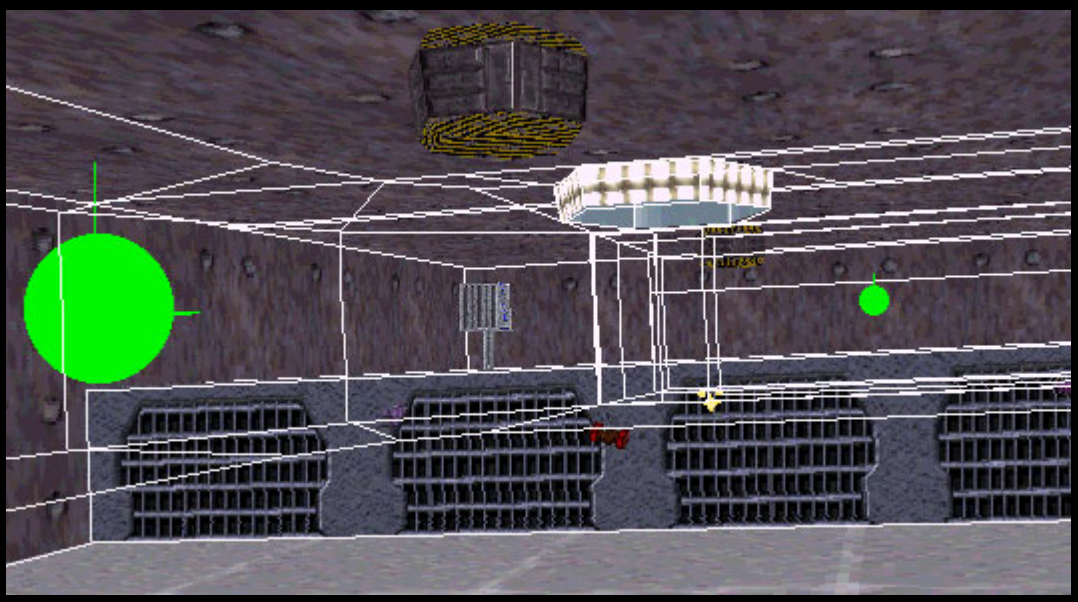
Grundsätzliches:
Jeder hat ein wenig andere Vorstellungen von dem was die Aufteilung der Räume dieser Spielarten betrifft. Daher ist folgendes meine Meinung zur Einstellung der Räume. Movieworld ist sicher kein gutes Beispiel für Hoard aber da die Einstellungen für Hoard und CFT ähnlich sind, ist der Mod nun mal da. Das wichtigste am ganzen ist das Einstellungsfenster 'Window - Room Properties'. Hier werden alle notwendigen Einstellungen vorgenommen.

Wähle nun einen Raum aus, der der Ablegepunkt für die Kugeln sein soll. Benenne ihn wie in Bild 2 z.B. **'Kino2 Blau'**. In Movieworld sind diese Ablegepunkte (es gibt Zwei) jeweils hinter den Bildwänden. Wenn Du nun den Raum gewählt hast, wähle in **'Room Properties'** Red Goal. Damit ist dieser Raum ein Goalroom. Möchtest Du nun in Deinem Goalroom nicht alle Flächen als Goal definieren, markiere (**nicht mark face!**) eine der Flächen und wähle in **'Room Properties', 'Current face is a goal face'.**

Das gleiche ist bei CTF, nur daß Du zwei benötigst, ein rotes Tor und ein blaues Tor. Dies war bei Movieworld kein Problem, da es zwei Kinos gib (Absicht). Hinter dem blauen Kino ist das Blaue Tor und hinter dem roten das Rote. Im Spiel sieht das dann so aus, das Du die eroberte Flagge quer durchs Level bringen mußt. Was wenn viele Gegner da sind nicht so einfach sein sollte. Es bringt also nichts, wenn zwei Torräume nebeneinander liegen. Ach ja, die Texturen für diese Mod's wird von Descent vergeben. Du kannst ganz normal texturieren, damit Deine Räume auch bei Anarchy gut aussehen. Bei der Auswahl der Spielerstartpunkte sollte darauf geachtet werden, das z.B. ein Spieler aus dem roten Team nicht direkt bei der blauen Flagge starten kann. Dies kannst Du erreichen, in dem Du den Startpunkten eine Team - Farbe zuteilst.

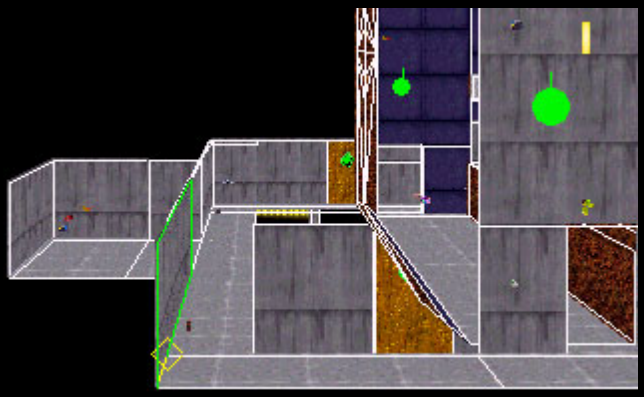
Dieser Startpunkt unten im Gang ist dem blauen Team zugeteilt (Bild rechts). Es gibt natürlich auch neutrale Räume wie im nächsten Bild. In Movieworld der Lüftungsraum.





Hier wählst Du einfach beide Farben um jedem Team die Möglichkeit zu geben hier zu starten.

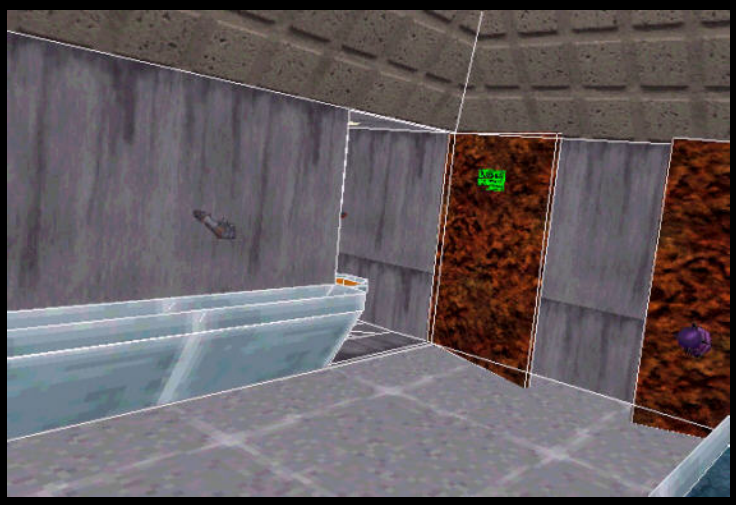
Jetzt zu Entropy!
Der Mod der seltener gespielt wird, aber wenn der Level gut ist viel Spass machen kann. Bei Entropy gibt es drei Typen von Räumen. Labors, Reparaturräume und pro Team den Energietanker. Die Alle werden auch über Window/Room Properties vergeben. In Movieworld habe ich pro Team 3 Labors, 1Energieraum und 1 Reparaturraum vergeben. Man kann natürlich auch jede Art der Räume mehrfach verteilen. In den Laboren liegen später die Viren, die man sammeln muss. Die anderen Räume sprechen für sich. In Movieworld ist der Raum hinter der Bildwand ,wie in CTF und Hoard Goalrooms, ein Labor pro Team.



Der Reparaturgang ist der Gang direkt davor. Und der Energietankraum ist Rechts vom Reparaturgang. Das nächste
Labor ist die Toilette und der Vorführraum.

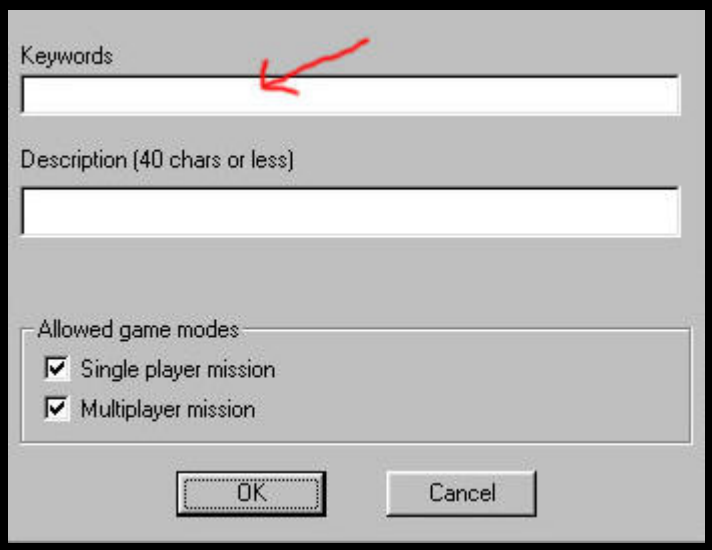
Die Einstellung dieser Räume sieht wie folgt aus:

S1=Labor Rot
S2=Energietanker Rot
S3=Reparaturraum Rot
S4=Labor Blau
S5=Energietanker Blau
S6=Reparaturraum Blau



Bei Entropy ist die Einstellung der Spielerstartpunkte **SEHR** wichtig. Es kann nicht sein, das ein roter Spieler in einem Labor vom Blauen Team startet.

Wenn du nun dein Level mit dem MN3 - Packer verpackst, musst Du noch folgende Key's eintragen (rechts):



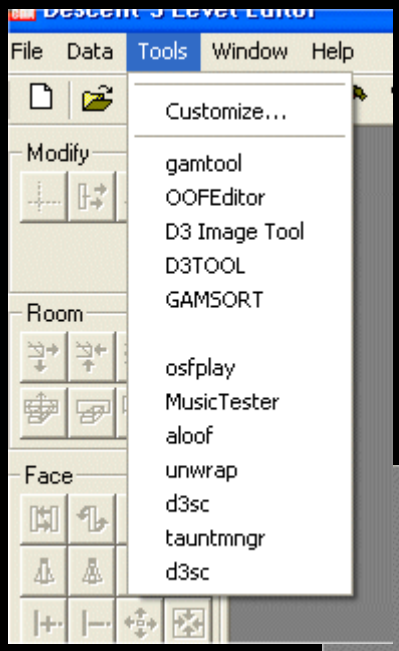
Für ein Hoard (Level mit einem Tor)	-	`GOALS1`
Für CTF	-	`GOALS2,GOALPERTEAM`
Für Entropy	-	`ENTROPY`

Für ein Level wie Movieworld würde das dann so aussehen:
`GOALS2,GOALPERTEAM,ENTROPY`

So das war's eigentlich schon.
Wie MUDDY immer so nett schreibt 'Ein T-Joint' freies Levelbauen.....
[DCG]Roadrunner

024 - Tools Menü anwenden

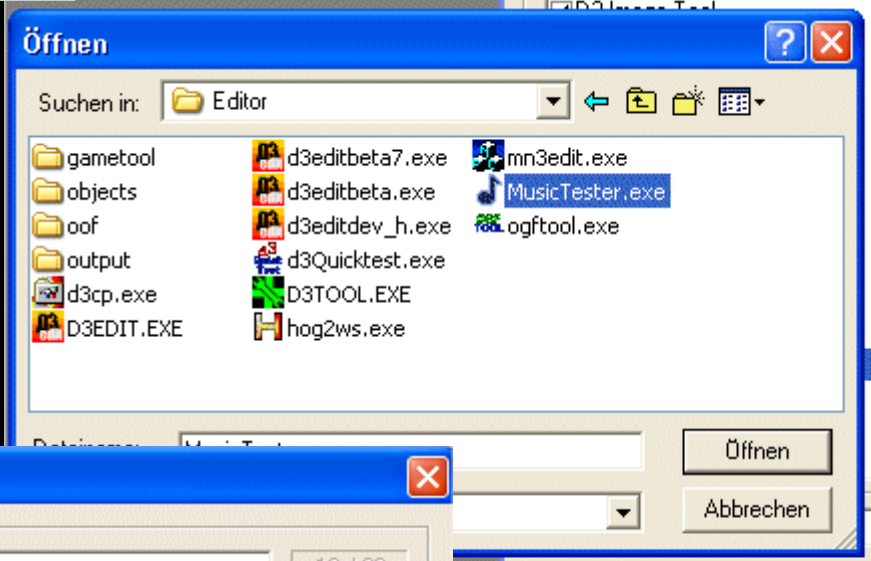
(LL)Atan

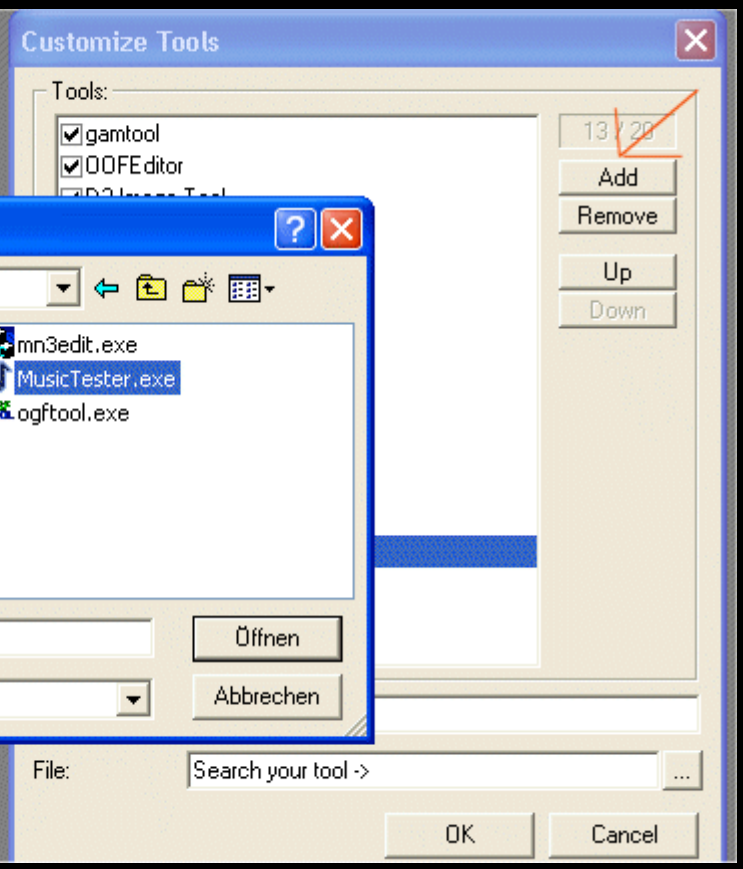


Es ist sinnvoll für die D3 Tools ein Verzeichnis anzulegen und dort die benötigten Tools abzulegen. Dann stellt man sich noch über das Tools Menü seine Tools für einen schnellen Zugriff zusammen und kann ohne lästiges Umschalten zur Windows Oberfläche seine Tools bequem starten. Ein Beispiel Menü zeigt das linke Bild.

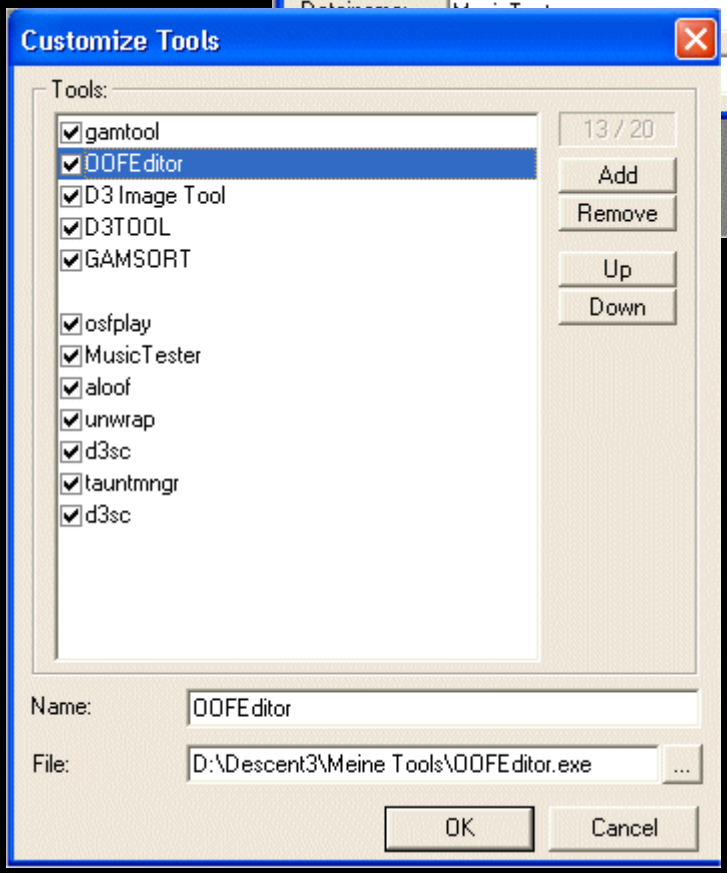
Beim ersten Start von D3Edit sind diese Einträge leer.
Über Customize kann man sich dann seine Tools zusammenstellen:

Add ruft ein Auswahlfenster auf mit dem wir uns durch unsere Ordner navigieren und das gewünschte Tool auswählen können. (unten)





Nach der Auswahl wird das Tool in die Liste eingetragen.



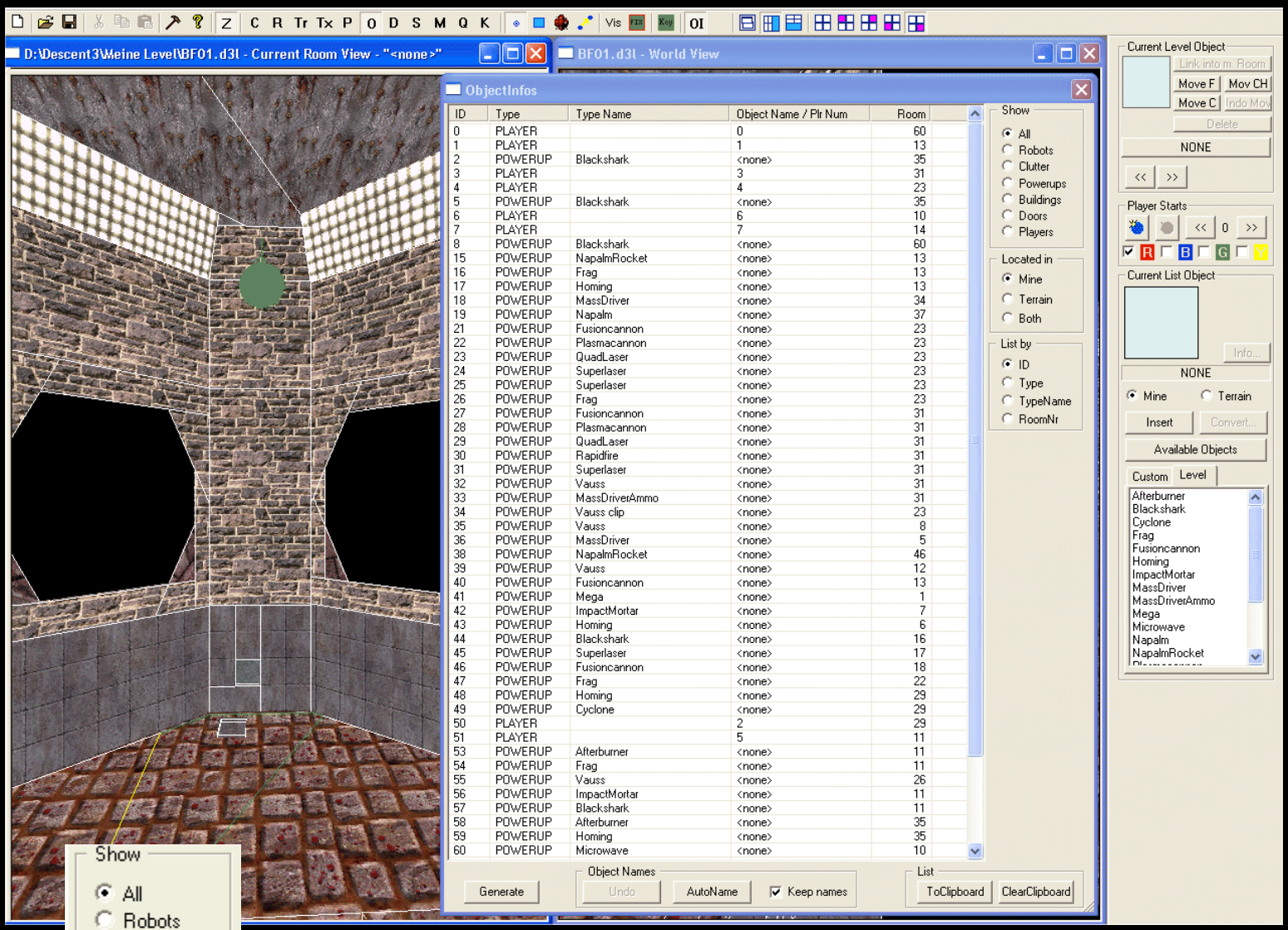
Es ist möglich einen eigenen Namen für jedes Tool einzugeben. Auch die Position innerhalb des Menüs ist einstellbar (links).
Abticken eines Eintrags bewirkt dessen Deaktivation im Menü, gleichwohl er sichtbar bleibt.

025 - Objekt Info Dialog (OI)

(LL)Atan

Der OI Dialog ist ein hilfreiches Tool mit dem man seine Level Objekte organisieren kann. Es ist vorwiegend zur Unterstützung beim Erstellen von SP Level gedacht, es ist aber auch hilfreich um einen Überblick über die Objekte in einem MP Level zu erhalten. Laden wir mal einen kleinen MP Level und schauen wir uns das Tool genauer an.

Dazu klicken wir auf den **OI** *Button* in der Tool Bar:



ID	Type	Type Name	Object Name / Plr Num	Room
0	PLAYER		0	60
1	PLAYER		1	13
2	POWERUP	Blackshark	<none>	35
3	PLAYER		3	31
4	PLAYER		4	23
5	POWERUP	Blackshark	<none>	35
6	PLAYER		6	10
7	PLAYER		7	14
8	POWERUP	Blackshark	<none>	60
15	POWERUP	NapalmRocket	<none>	13
16	POWERUP	Frag	<none>	13
17	POWERUP	Homing	<none>	13
18	POWERUP	MassDriver	<none>	34
19	POWERUP	Napalm	<none>	37
21	POWERUP	Fusioncannon	<none>	23
22	POWERUP	Plasmacannon	<none>	23
23	POWERUP	QuadLaser	<none>	23
24	POWERUP	Superlaser	<none>	23
25	POWERUP	Superlaser	<none>	23
26	POWERUP	Frag	<none>	23
27	POWERUP	Fusioncannon	<none>	31
28	POWERUP	Plasmacannon	<none>	31
29	POWERUP	QuadLaser	<none>	31
30	POWERUP	Rapidfire	<none>	31
31	POWERUP	Superlaser	<none>	31
32	POWERUP	Vauss	<none>	31
33	POWERUP	MassDriverAmmo	<none>	31
34	POWERUP	Vauss clip	<none>	23
35	POWERUP	Vauss	<none>	8
36	POWERUP	MassDriver	<none>	5
38	POWERUP	NapalmRocket	<none>	46
39	POWERUP	Vauss	<none>	12
40	POWERUP	Fusioncannon	<none>	13
41	POWERUP	Mega	<none>	1
42	POWERUP	ImpactMortar	<none>	7
43	POWERUP	Homing	<none>	6
44	POWERUP	Blackshark	<none>	16
45	POWERUP	Superlaser	<none>	17
46	POWERUP	Fusioncannon	<none>	18
47	POWERUP	Frag	<none>	22
48	POWERUP	Homing	<none>	29
49	POWERUP	Cyclone	<none>	29
50	PLAYER		2	29
51	PLAYER		5	11
53	POWERUP	Afterburner	<none>	11
54	POWERUP	Frag	<none>	11
55	POWERUP	Vauss	<none>	26
56	POWERUP	ImpactMortar	<none>	11
57	POWERUP	Blackshark	<none>	11
58	POWERUP	Afterburner	<none>	35
59	POWERUP	Homing	<none>	35
60	POWERUP	Microwave	<none>	10

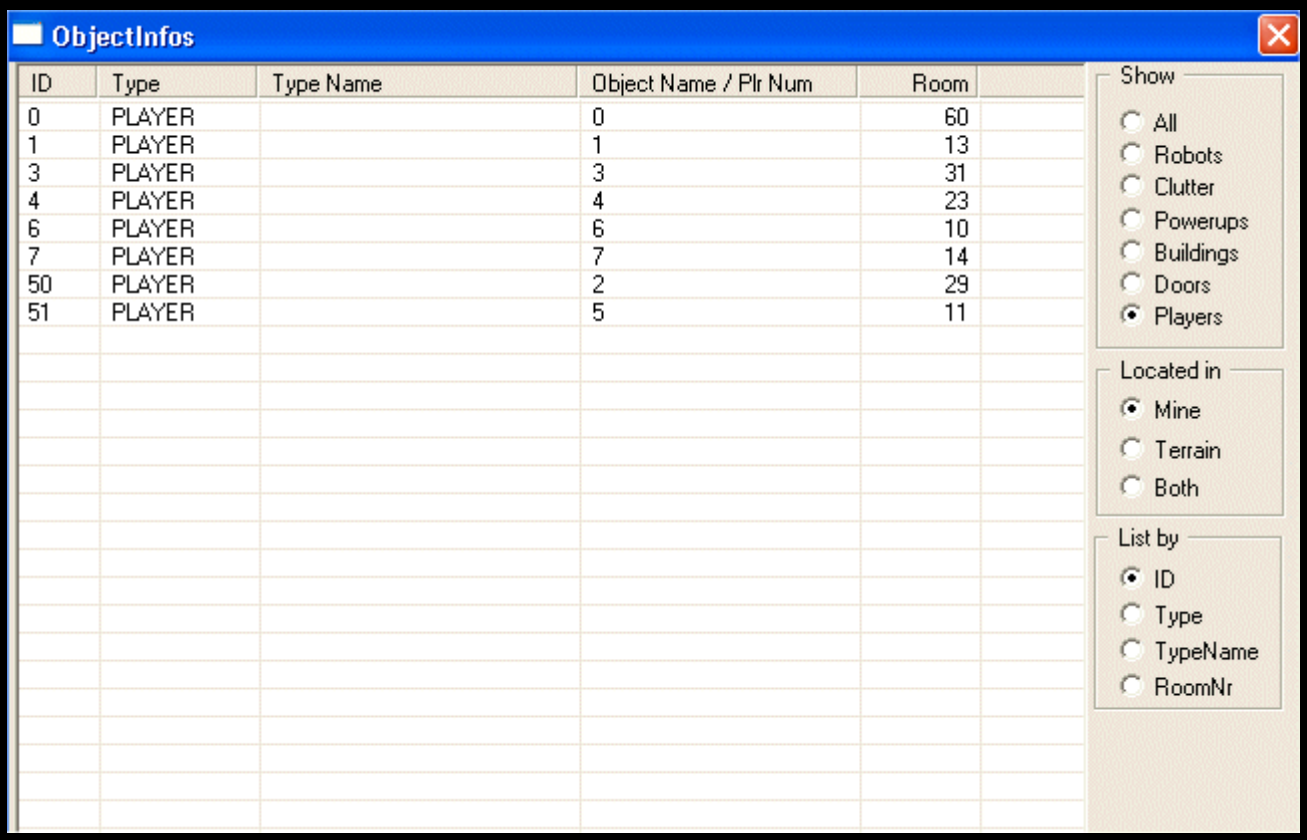
Der Dialog öffnet sich mit einer Vielzahl an Informationen.
Innerhalb des Ausgabe-Fensters befinden sich Spalten für:

ID	Type	Type Name	Object Name / Plr Num	Room

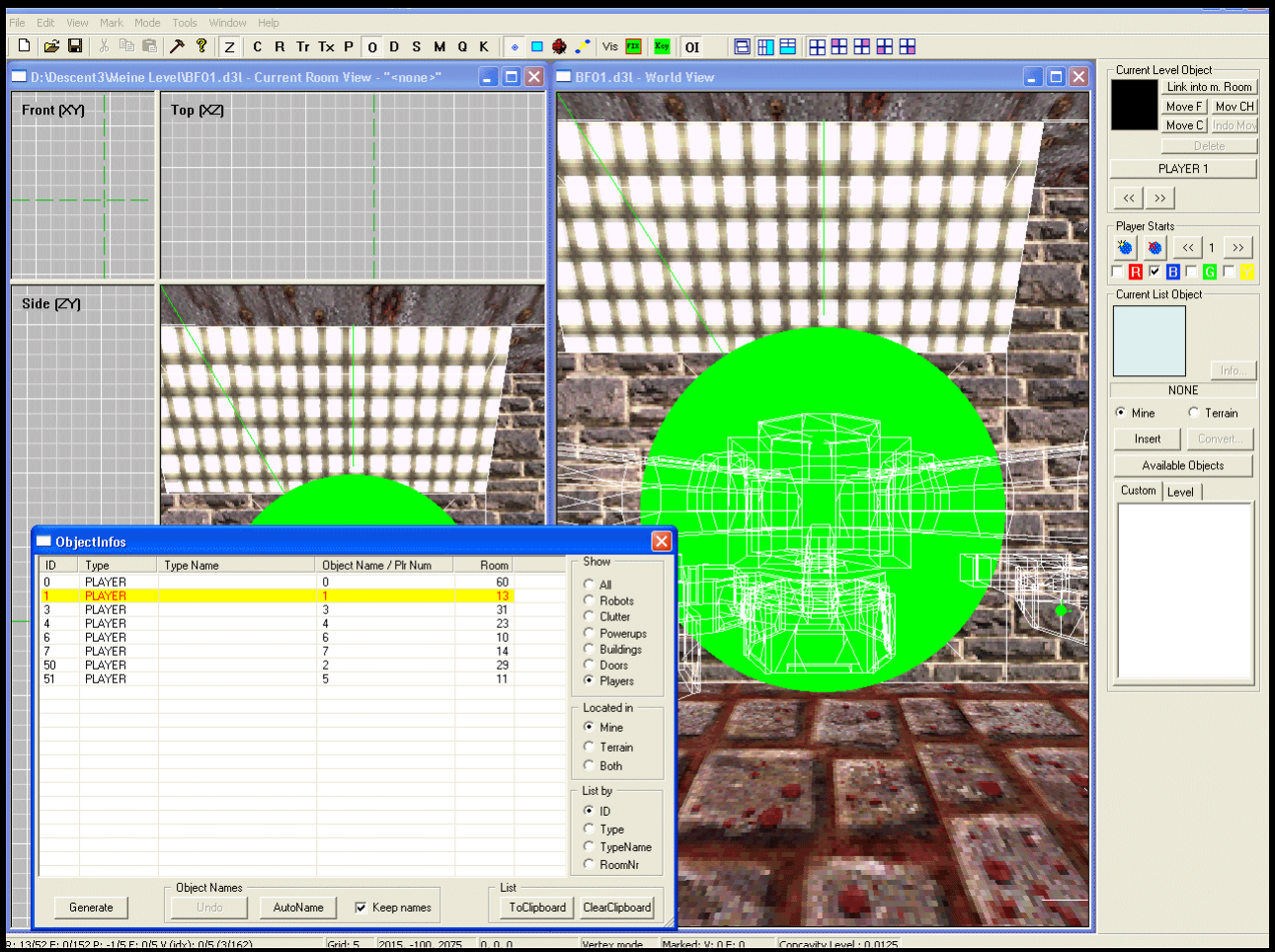
Wir haben verschiedene Möglichkeiten die Objekt-Informationen in dem Fenster anzuzeigen. In dem obigen Beispiel werden alle Objekte innerhalb der Mine fortlaufend ihrer ID angezeigt.

Einen schnellen Überblick über die eingesetzten Player-Objekte bringt unter Show ein Klick auf *Players* (Rechts).

Der OI Dialog zeigt daraufhin nur noch die Player-Objekte an:

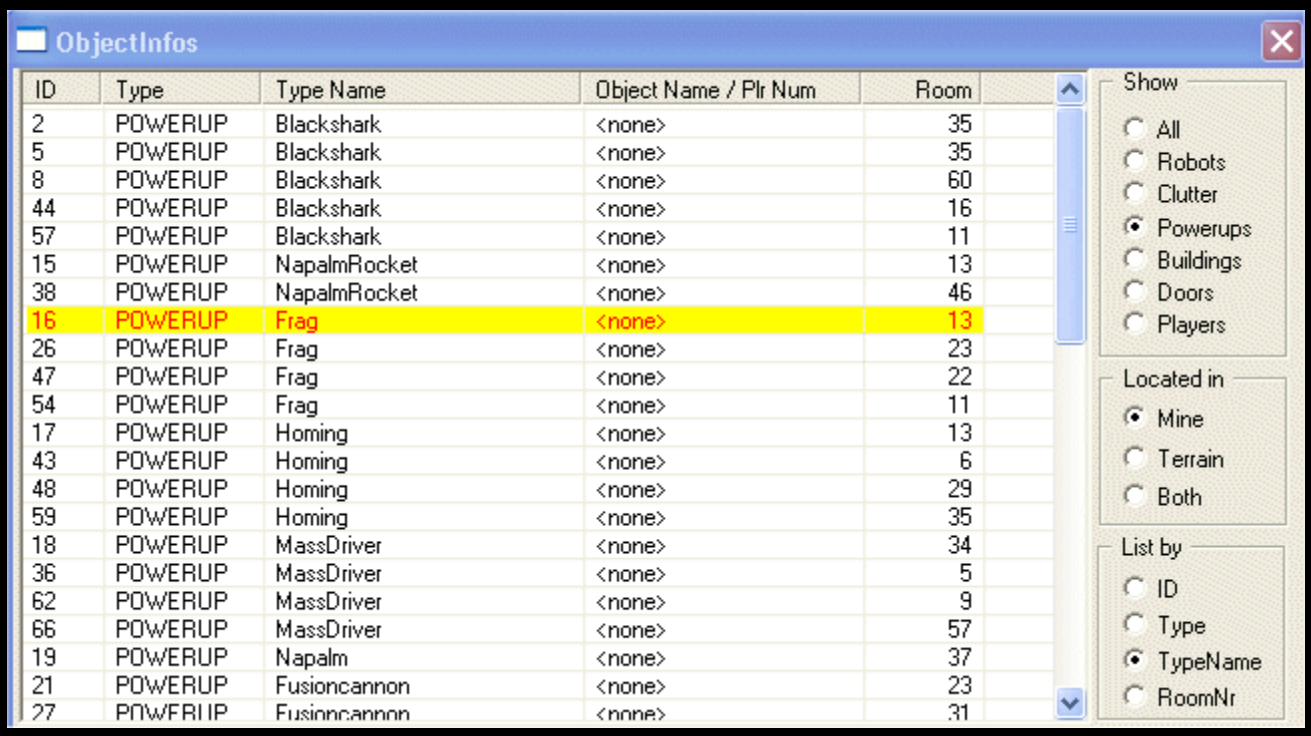


Die Reihenfolge ergibt sich aus der ID, also aus dem Index mit dem D3Edit dieses Objekt in den Level speichert. Interessanter ist da eher die Room-Nummer anhand der wir sehen können welcher Raum welches Player Objekt enthält.
Wollen wir uns eines der Objekte genauer ansehen, dann klicken wir in die entsprechende Zeile. D3Edit zeigt das gewählte Objekt sofort an und im OI Dialog wird die Zeile mit dem Objekt hervorgehoben.

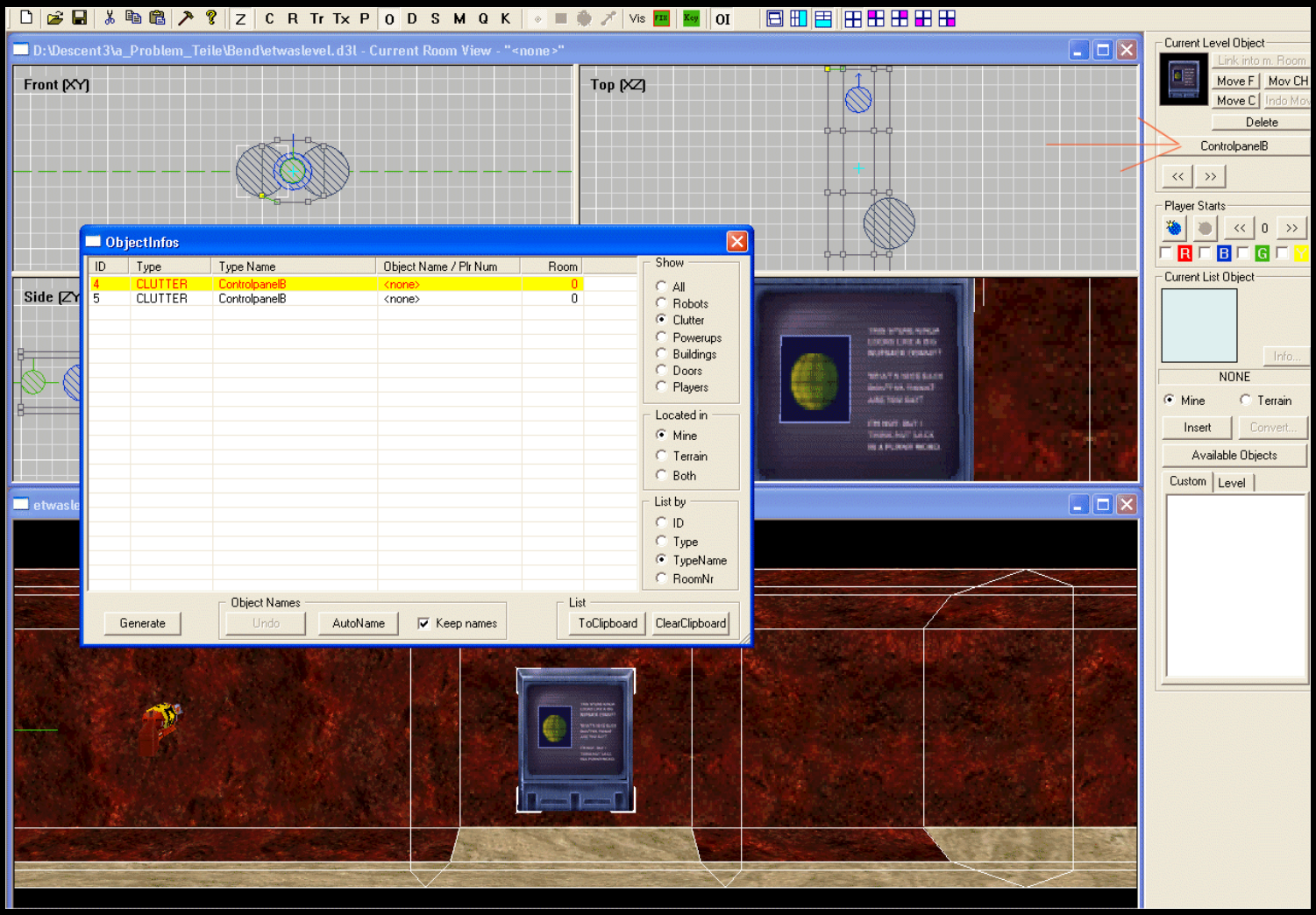


Nun wählen wir zum Beispiel die Powerups an und klicken dann auf das Objekt Nr 16. D3Edit befördert uns in den Raum der das Objekt enthält und zoomt direkt auf die Frag. Am rechten Rand sehen wir, dass auch die Objekt Bar entsprechend geändert wird. Das aktuelle Objekt wird auch dort sofort angezeigt.

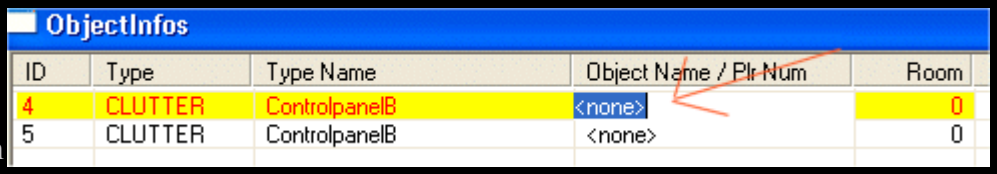
Wollen wir uns einen Überblick über die Anzahl der Frags in der Mine verschaffen, dann listen wir die Objekte nach ‚Type Name' auf.



So kann man schnell erkennen *wie viele* dieser Objekte *wo* eingesetzt sind und per Mausklick sofort zum Objekt springen.
Ein Objekt muss benannt werden um es per DALLAS anzusprechen. Um einen Namen einzugeben gibt es in der Objekt Bar einen Button (siehe Pfeil). Nach Betätigung erscheint ein Eingabefenster.



Wir können aber auch mit Mausklick links auf die Spalte *Objekt Name* klicken. Dadurch erhalten wir ein Eingabefenster in der entsprechenden Zeile und Spalte.

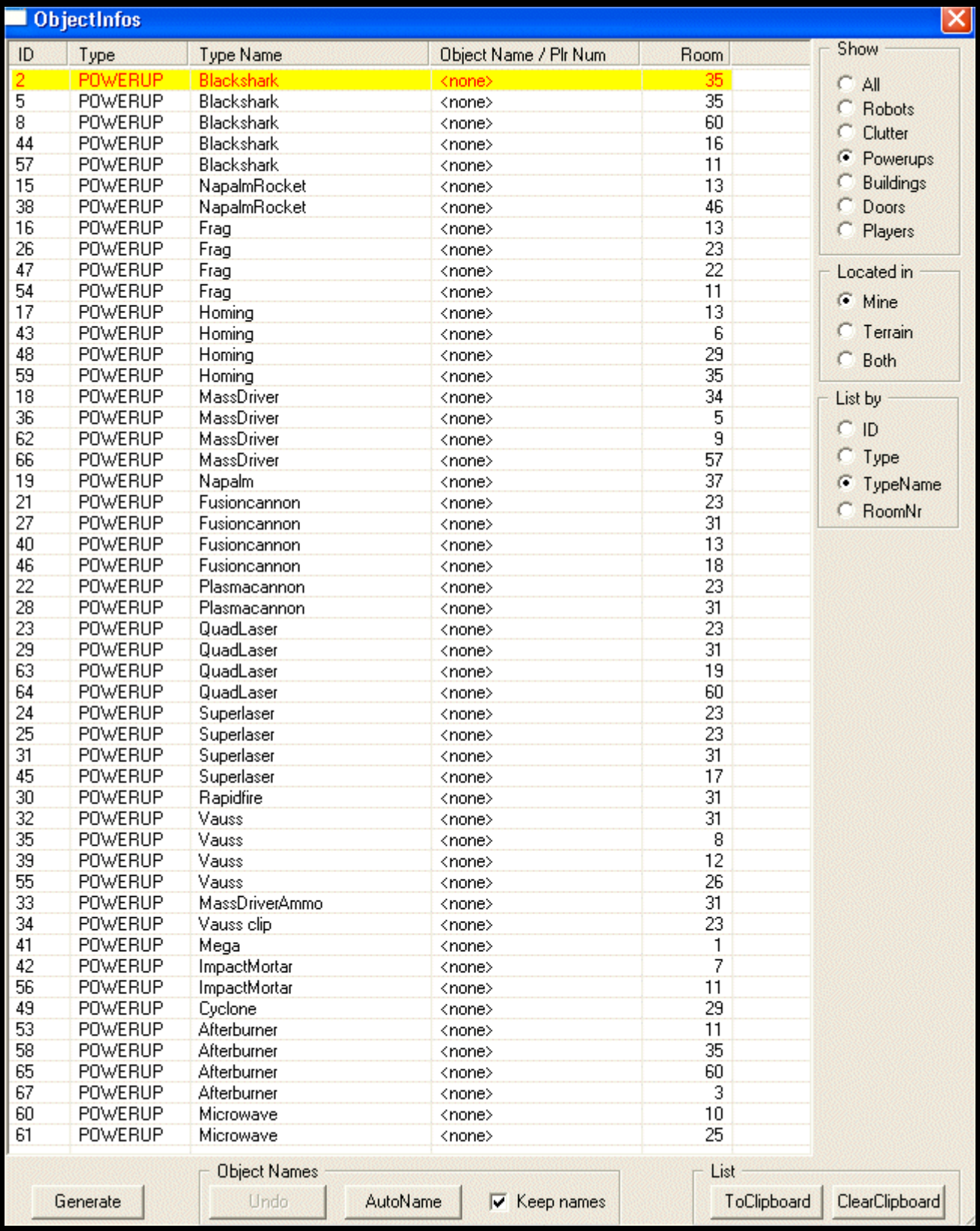


Hier geben wir dann einen Objekt Namen ein und bestätigen mit Return.

In der Objekt Bar wird sofort der neue Name angezeigt.

Bei einer Eingabe in der Objekt Bar aktualisiert sich die **OI**-Liste automatisch.

Es gibt in OI auch eine automatische Namensgebung. Sehen wir uns die Powerup Liste eines Levels mal an:



ID	Type	Type Name	Object Name / Plr Num	Room
2	POWERUP	Blackshark	<none>	35
5	POWERUP	Blackshark	<none>	35
8	POWERUP	Blackshark	<none>	60
44	POWERUP	Blackshark	<none>	16
57	POWERUP	Blackshark	<none>	11
15	POWERUP	NapalmRocket	<none>	13
38	POWERUP	NapalmRocket	<none>	46
16	POWERUP	Frag	<none>	13
26	POWERUP	Frag	<none>	23
47	POWERUP	Frag	<none>	22
54	POWERUP	Frag	<none>	11
17	POWERUP	Homing	<none>	13
43	POWERUP	Homing	<none>	6
48	POWERUP	Homing	<none>	29
59	POWERUP	Homing	<none>	35
18	POWERUP	MassDriver	<none>	34
36	POWERUP	MassDriver	<none>	5
62	POWERUP	MassDriver	<none>	9
66	POWERUP	MassDriver	<none>	57
19	POWERUP	Napalm	<none>	37
21	POWERUP	Fusioncannon	<none>	23
27	POWERUP	Fusioncannon	<none>	31
40	POWERUP	Fusioncannon	<none>	13
46	POWERUP	Fusioncannon	<none>	18
22	POWERUP	Plasmacannon	<none>	23
28	POWERUP	Plasmacannon	<none>	31
23	POWERUP	QuadLaser	<none>	23
29	POWERUP	QuadLaser	<none>	31
63	POWERUP	QuadLaser	<none>	19
64	POWERUP	QuadLaser	<none>	60
24	POWERUP	Superlaser	<none>	23
25	POWERUP	Superlaser	<none>	23
31	POWERUP	Superlaser	<none>	31
45	POWERUP	Superlaser	<none>	17
30	POWERUP	Rapidfire	<none>	31
32	POWERUP	Vauss	<none>	31
35	POWERUP	Vauss	<none>	8
39	POWERUP	Vauss	<none>	12
55	POWERUP	Vauss	<none>	26
33	POWERUP	MassDriverAmmo	<none>	31
34	POWERUP	Vauss clip	<none>	23
41	POWERUP	Mega	<none>	1
42	POWERUP	ImpactMortar	<none>	7
56	POWERUP	ImpactMortar	<none>	11
49	POWERUP	Cyclone	<none>	29
53	POWERUP	Afterburner	<none>	11
58	POWERUP	Afterburner	<none>	35
65	POWERUP	Afterburner	<none>	60
67	POWERUP	Afterburner	<none>	3
60	POWERUP	Microwave	<none>	10
61	POWERUP	Microwave	<none>	25

Die Objekte sind im Grundzustand alle mit <none> gekennzeichnet, sie enthalten also keinen Namen. Nach einem Klick auf den **Auto Name Button** werden die Objekte eines Typs automatisch durchnummeriert. Sind Objekte vorher benannt worden, kann mit **Keep Names** bestimmt werden, ob die Namen bei der automatischen Benennung beibehalten werden. Mit **Undo** kann der Vorgang rückgängig gemacht werden. Die Liste kann auch mit **ToClipboard** in die Zwischenablage zur Weiterverarbeitung gespeichert werden. Es ist möglich die Zwischenablage mit **ClearClipboard** zu löschen.

ObjectInfos

ID	Type	Type Name	Object Name / Plr Num	Room
2	POWERUP	Blackshark	Powerup_0	35
5	POWERUP	Blackshark	Powerup_1	35
8	POWERUP	Blackshark	Powerup_2	60
44	POWERUP	Blackshark	Powerup_3	16
57	POWERUP	Blackshark	Powerup_4	11
15	POWERUP	NapalmRocket	Powerup_5	13
38	POWERUP	NapalmRocket	Powerup_6	46
16	POWERUP	Frag	Powerup_7	13
26	POWERUP	Frag	Powerup_8	23
47	POWERUP	Frag	Powerup_9	22
54	POWERUP	Frag	Powerup_10	11
17	POWERUP	Homing	Powerup_11	13
43	POWERUP	Homing	Powerup_12	6
48	POWERUP	Homing	Powerup_13	29
59	POWERUP	Homing	Powerup_14	35
18	POWERUP	MassDriver	Powerup_15	34
36	POWERUP	MassDriver	Powerup_16	5
62	POWERUP	MassDriver	Powerup_17	9
66	POWERUP	MassDriver	Powerup_18	57
19	POWERUP	Napalm	Powerup_19	37
21	POWERUP	Fusioncannon	Powerup_20	23
27	POWERUP	Fusioncannon	Powerup_21	31
40	POWERUP	Fusioncannon	Powerup_22	13
46	POWERUP	Fusioncannon	Powerup_23	18
22	POWERUP	Plasmacannon	Powerup_24	23
28	POWERUP	Plasmacannon	Powerup_25	31
23	POWERUP	QuadLaser	Powerup_26	23
29	POWERUP	QuadLaser	Powerup_27	31
63	POWERUP	QuadLaser	Powerup_28	19
64	POWERUP	QuadLaser	Powerup_29	60
24	POWERUP	Superlaser	Powerup_30	23
25	POWERUP	Superlaser	Powerup_31	23
31	POWERUP	Superlaser	Powerup_32	31
45	POWERUP	Superlaser	Powerup_33	17
30	POWERUP	Rapidfire	Powerup_34	31
32	POWERUP	Vauss	Powerup_35	31
35	POWERUP	Vauss	Powerup_36	8
39	POWERUP	Vauss	Powerup_37	12
55	POWERUP	Vauss	Powerup_38	26
33	POWERUP	MassDriverAmmo	Powerup_39	31
34	POWERUP	Vauss clip	Powerup_40	23
41	POWERUP	Mega	Powerup_41	1
42	POWERUP	ImpactMortar	Powerup_42	7
56	POWERUP	ImpactMortar	Powerup_43	11
49	POWERUP	Cyclone	Powerup_44	29
53	POWERUP	Afterburner	Powerup_45	11
58	POWERUP	Afterburner	Powerup_46	35
65	POWERUP	Afterburner	Powerup_47	60
67	POWERUP	Afterburner	Powerup_48	3
60	POWERUP	Microwave	Powerup_49	10
61	POWERUP	Microwave	Powerup_50	25

Show
- All
- Robots
- Clutter
- Powerups
- Buildings
- Doors
- Players

Located in
- Mine
- Terrain
- Both

List by
- ID
- Type
- TypeName
- RoomNr

Generate

Object Names: Undo, AutoName, Keep names

List: ToClipboard, ClearClipboard

Ein besonders Feature verbirgt sich hinter dem **Generate**-Button. Wer sehr tief in die DALLAS Programmierung eingestiegen ist wird diesen Leckerbissen zu schätzen wissen. Diese Ausgabe erfolgt auch wieder in die Zwischenablage und kann dann eingefügt werden. Hier ein Beispiel aus dem *Mission to Saturn Part II*-Script.

```
/*
$$ENUM MyRobots
0:"B_P2Gr2"
1:"B_P3Gr2"
2:"B1"
3:"B2"
4:"B3"
5:"Bot_32"
6:"Bot_33"
7:"Bot_34"
8:"Bot_35"
9:"B_P2Gr1"
10:"B4"
11:"KameraDoorOpen"
12:"Bot_51"
13:"Bot_52"
14:"Bot_55"
15:"Bot_56"
16:"B5"
17:"B6"
18:"B_P1Gr2"
19:"Bot_53"
20:"Bot_54"
21:"Bot_43"
22:"Bot_44"
23:"Bot_45"
24:"Bot_59"
25:"Bot_46"
26:"Bot_60"
27:"Bot_61"
28:"Bot_62"
29:"Bot_63"
30:"Bot_64"
31:"Bot_65"
32:"Bot_66"
33:"Bot_67"
34:"Bot_68"
35:"Bot_69"
36:"B8"
37:"B9"
38:"B10"
39:"B_P3Gr5"
40:"B_P1Gr5"
41:"B_P2Gr5"
42:"B_P1Gr4"
43:"B_P2Gr4"
44:"B_P3Gr4"
45:"B_P3Gr6"
46:"B19"
47:"B20"
48:"B21"
49:"B22"
50:"B23"
51:"B24"
52:"B25"
53:"B26"
54:"B27"
55:"B28"
56:"Bot_70"
57:"Schweisser1"
58:"Bot_36"
59:"Bot_37"
60:"Bot_71"
61:"B29"
62:"Bot_38"
63:"Bot_85"
64:"Bot_86"
65:"B11"
66:"B12"
67:"B13"
68:"B14"
69:"B15"
70:"B16"
71:"B17"
72:"B18"
73:"Bot_87"
74:"Bot_88"
75:"Bot_47"
76:"Bot_48"
77:"Bot_49"
78:"Bot_72"
79:"Cam-Teleport1"
80:"B_P1Gr6"
81:"B_P2Gr6"
82:"B_P3Gr1"
83:"B_P1Gr3"
84:"B_P2Gr3"
85:"B_P3Gr3"
86:"B_P1Gr1"
87:"B7"
88:"CrashPlage"
$$END
*/
```

```
#define NUM_BOT_OBJECT_NAMES         89
char *Bot_Object_names[NUM_BOT_OBJECT_NAMES] = {
"B_P2Gr2",
"B_P3Gr2",
"B1",
"B2",
"B3",
"Bot_32",
"Bot_33",
"Bot_34",
"Bot_35",
"B_P2Gr1",
"B4",
"KameraDoorOpen",
"Bot_51",
"Bot_52",
"Bot_55",
"Bot_56",
"B5",
"B6",
"B_P1Gr2",
"Bot_53",
"Bot_54",
"Bot_43",
"Bot_44",
"Bot_45",
"Bot_59",
"Bot_46",
"Bot_60",
"Bot_61",
"Bot_62",
"Bot_63",
"Bot_64",
"Bot_65",
"Bot_66",
"Bot_67",
"Bot_68",
"Bot_69",
"B8",
"B9", "B10",
"B_P3Gr5",
"B_P1Gr5",
"B_P2Gr5",
"B_P1Gr4",
"B_P2Gr4",
"B_P3Gr4",
"B_P3Gr6",
"B19",
"B20",
"B21",
"B22",
"B23",
"B24",
"B25",
"B26",
"B27",
"B28",
"Bot_70",
"Schweisser1",
"Bot_36",
"Bot_37",
"Bot_71",
"B29",
"Bot_38",
"Bot_85",
"Bot_86",
"B11",
 "B12",
"B13",
"B14",
"B15",
"B16",
 "B17",
"B18",
"Bot_87",
"Bot_88",
"Bot_47",
"Bot_48",
"Bot_49",
"Bot_72",
"Cam-Teleport1",
"B_P1Gr6",
 "B_P2Gr6",
"B_P3Gr1",
"B_P1Gr3",
"B_P2Gr3",
"B_P3Gr3",
"B_P1Gr1",
"B7",
"CrashPlage",
};
```

Abschnitt C – Fortgeschrittenes Bauen

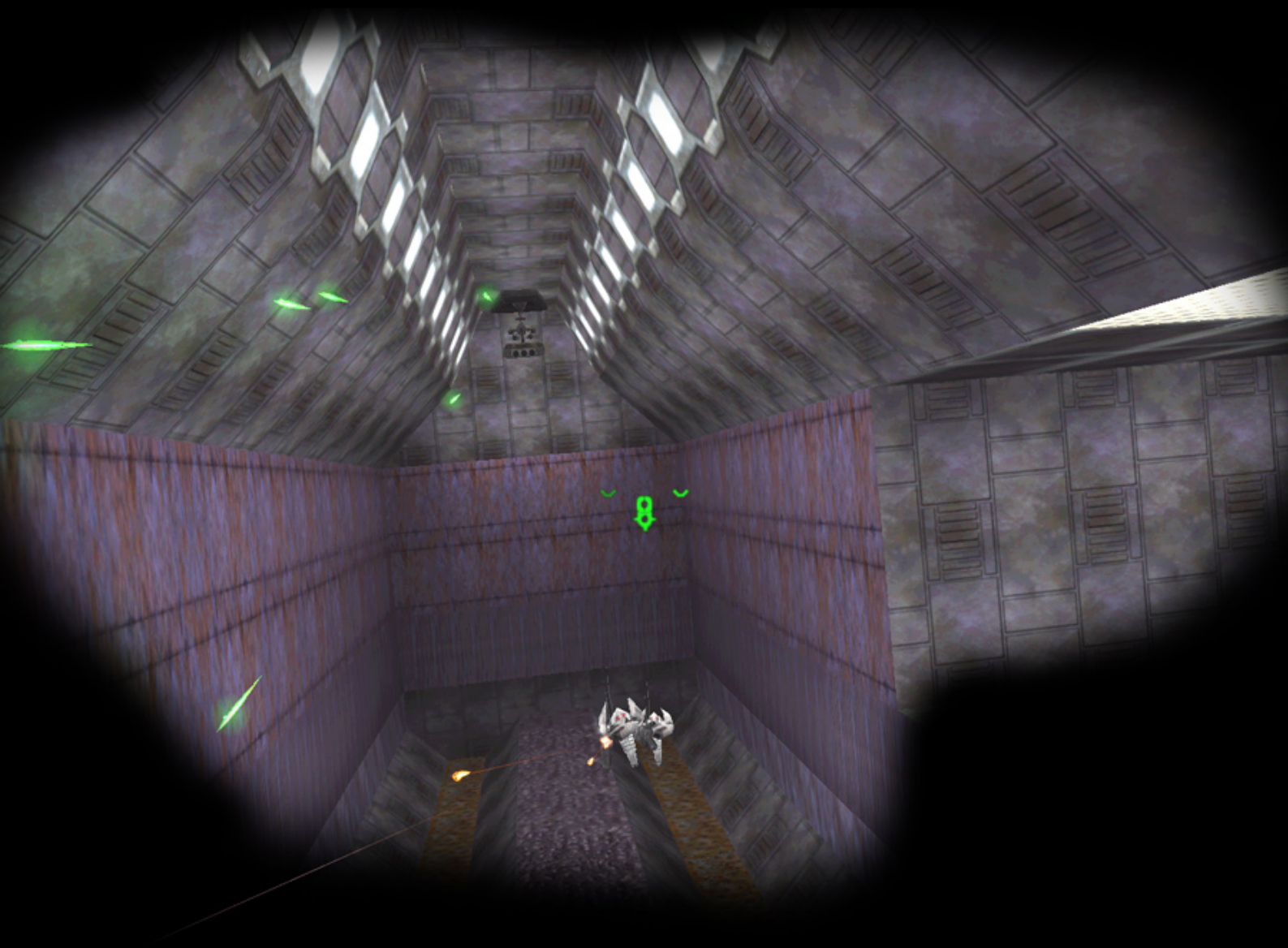
Für diejenigen, welche das mit dem Levelbauen wirklich ernst meinen sei empfohlen, sich die Winkelsätze des Dreiecks anzulernen. Sehr hilfreich beim Lathen, Benden oder allgemein um bestimmte Werte die man braucht zu errechnen. Und sooo kompliziert iss' auch net....

Die Sirianische Methode ist ein durchdachter Ablauf, um qualitativ hochwertig zu arbeiten				
026	**Bau einer hochwertigen Shell**	Lektion 1	Sirian	103
027	**Multiple Layers**	Lektion 2	Sirian	106
028	**Multiple Extrusion**	Lektion 3	Sirian	110
029	**Multiplanare Extrusionen kombinieren**	Lektion 4	Sirian	114
Die drei Tuts liegen thematisch etwa zwischen hier und dem vorigen Abschnitt				
030	**Raum Teilen**	Erklärt die Raumteilungs-Funktion	Atan	118
031	**Vertexoperationen an Faces**	Behandelt das Hinzufügen zu und Entfernen von Verts aus Faces.	Hydra	121
032	**Segmente anwählen**	Verwendung des K-Tools	Atan	122
Die nächsten fünf Tutorials befassen sich mit dem Verbiegen von Bauelementen in D3Edit.				
033	**Benden – Grundlagen**	Die Bendoperation aus der Sicht von D3Edit	(LL)Dark	124
034	**New Style Bend, v39**	v39: der New Bend Dialog	Ragil Ral	126
035	**Die Bend-Funktion**	Die Bendoperation aus praktischer Sicht	Schplurg	129
036	**Runderneuerte Bend-Funktion in v40**	v40: Bend - Runderneuert	Ragil Ral	131
037	**Bend: Mathematik**	Errechnen der benötigten Werte für das gewünschte Resultat	Papacat	133
038	**Benden, Objekte inklusive**	Objekte bei Bendoperationen mitnehmen	Atan	135
039	**Biegung ohne Bend**	Elegante Realisierung von Biegungen ohne die Bend-Funktion.	Fischlein	137
040	**Lathe – Vertiefung**	Hier werden die Extrude- und die Lathe-Funktion eingehender erläutert.	Hydra	138
041	**Extrude – Vertiefung**		Hydra	141
042	**Extrude: Zero**	Extrusion mit Null Weite	WillyP	147
043	**Spieglein, Spieglein...**	Wie man Spiegel in ein Level setzt sowie deren Einschränkungen	Ragil Ral	154
044	**Raum im Raum**	Einen inneren Raum anbinden	Fischlein	157
045	**Eine Sphäre machen**	Arbeitsschritte zur Erstellung einer Kugel	WillyP	159
046	**Komplexität ganz einfach**	Schritt für Schritt zu einem sehr komplexen Gebilde	Papacat	167
047	**Primitives**	Basisformen - ganz einfach	Ragil Ral	178
048	**Felsformationen erzeugen**	Unregelmäßige Formen erstellen: Objekt	Hydra	184
049	**Tunnel im Höhlen-Stil**	Unregelmäßige Formen erstellen: Raum	WillyP	186

The Sirian Method

Dieses sehr in die Tiefe gehende Tutorial wurde ursprünglich von Sirian - einer der Betatester von D3Edit - im ‚D3Edit Forum' gepostet; es besteht aus vier Teilen.

Hier wird erklärt, wie man hochwertige Shells baut, welche sich einfach weiterverarbeiten lassen. Es ist eine beinahe narrensichere Methode, wenn man sie sich erstmal angewöhnt hat und vereinfacht vieles.
Man sollte von den Werkzeugen in D3Edit zumindest ein wenig Ahnung haben, ansonsten wird man sich hier schwer tun, mehrmaliges Durchgehen kann auch nichts schaden.



Bedauerlicherweise habe ich nichts Weiteres mehr von diesem Autor auftreiben können...

026 - Bau einer hochwertigen Shell

Sirian

Der erste Schritt um einen Raum zu machen ist die Erstellung der Shell. Diese ist das Konglomerat aus Faces, welches das ‚Innen' von allem Anderen trennt. Die Shell muß luftdicht sein... keine doppelten Vertices, überhaupt keine Lücken, und alle Kanten paßgenau zwischen den Faces.

Nun, wenn Du einen großen Raum machst der größtenteils rechteckig ist kann Deine Shell ein einzelnes Face sein, welches Du extrudierst. So mache ich beinahe alle meine Räume. Üblicherweise mache ich den Boden, extrudiere den dann, sodaß der Boden und die Decke jeweils ein einzelnes Face sind, oder für konkave Räume mehrere Faces.

Meine Wände sind aufgeteilt, und das aus mehreren Gründen. Erstens, es ist absolut grundlegend, daß du eine gewisse ‚Eindämmung' für jedes Face hast, wo Du planst einen anderen Raum anzuhängen und ein Portal zu erzeugen. Wenn Du einen Raum an Deinen current Room anhängst und die zwei Faces nicht aneinanderpassen, ‚zerbricht' der Editor eines oder beide Faces um das Anhängen ohne konkave Faces zu ermöglichen. Das ist eigentlich sehr sehr hilfreich, aber dazu später mehr. Wenn er die Faces zerbricht, splittet der Editor sie in kleinere Faces auf und das fügt ZU ALLEN ANLIEGENDEN FACES Vertices hinzu, welche von den ‚Sprüngen' ‚getroffen' werden. Wenn es Dir passiert daß Du ein anderes Portal-Face an jenem Face anliegen hast, zu dem du gerade anhängst, kann es von einem ‚Sprung' getroffen werden und bekommt ein oder mehr zusätzliche Verts, was ein ‚mismatched portal' erzeugt, eine sehr sehr üble Sache. Also wenn Du Deine Räume planst, mußt Du planen WO Du deine Portale haben wirst, und als generelle Daumenregel wirst Du eine ‚Eindämmung' um die geplanten Portale haben wollen, sodaß beim Raumanhängen entstehende ‚Sprünge' von ‚Puffer'-Faces abgefangen werden, welche Du zwischen den Portalen hast.

Sagen wir zum Beispiel, Du machst einen Raum mit einer würfeligen Shell. Ein Face als Boden, eins als Decke, eins für jede Wand. Wenn du ein Portal in der Nordwand erzeugst, in ihrer oberen linken Ecke, und dann später einen Raum an die Westwand, in die Mitte des Faces, anhängst... hast Du höchstwahrscheinlich einen ‚Sprung' von der westlichen Anhängung ausgehend, welcher das Portal in der Nordwand trifft, und es schmeißt Dich königlich auf.

Aber was Du anstatt dessen machen kannst, ist die Westwand aus drei Faces zu machen anstatt nur einem. Die ‚Mitte' der drei Faces ist der Ort wo Du das eine Portal anhängst, dann ist der ‚Puffer' zwischen dem und der Nordwand, sodaß keine Sprünge die Nord- oder Südwand erreichen. Und Du bekommst keine mismatched portals, welche Dich dazu bringen ganze Räume zu löschen und zu restrukturieren.

Vertrau mir, ich hab's auf die harte Art gelernt.

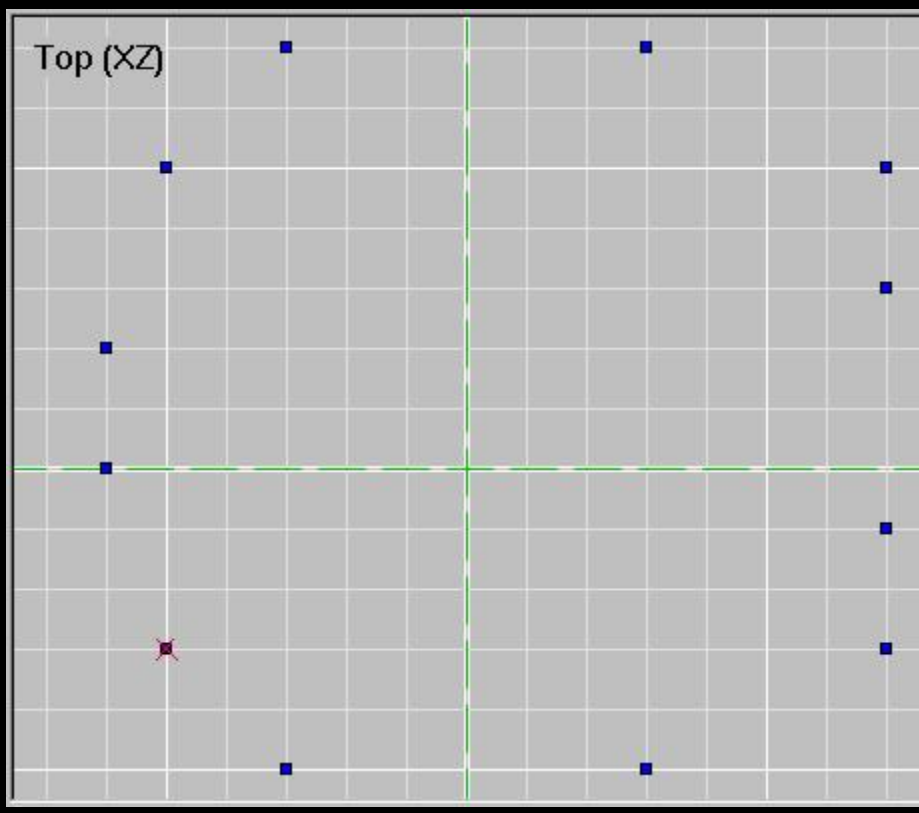
Außerdem, das Erzeugen von aufgeteilten Wänden gibt Dir einen Ort, wo Du die Wandtexturen variieren kannst, um den Raum interessanter aussehen zu lassen.

Gehen wir also nun durch die Schritte des Baues einer Shell, mit der Sirianischen Methode.

1) Du mußt in den Vertex Mode. Der Editor startet in diesem Modus, aber wenn Du zuvor etwas anderes bearbeitet hast könntest Du in einem anderen Mode sein.
2) Wähle das Face aus, von dem Du extrudieren wirst. Bei mir ist das üblicherweise der Boden, aber gelegentlich eine Wand. Wir werden im Beispiel den Boden verwenden.
3) Wähle die passende 2d-Ansicht aus (Für den Boden wäre das die Top-Ansicht [X-Z]).
4) Bekomme ein Bild von der Form des Bodens vor Deinem geistigen Auge (es muß nicht-konkav sein – man kann konkave Formen machen, aber das involviert multiples Extrudieren von multiplen Faces).

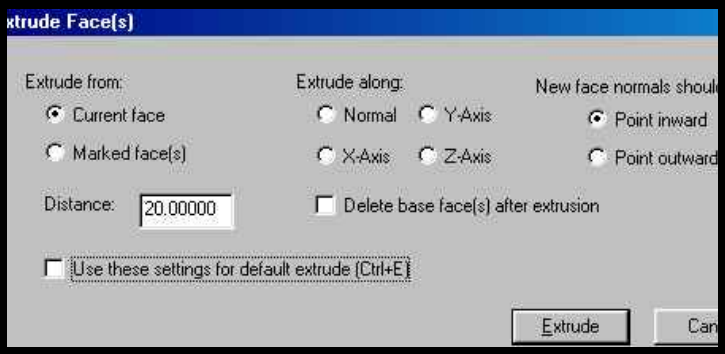
4a) Ich empfehle *STRENGSTENS* daß Du für alle 2d-Operationen den 3d-Cursor (das rote **X**) verwendest. Wenn Du die Maus verwendest und klickst, verlierst Du die Kontrolle über die Tiefe, wo der Cursor ist. Wenn Du den Cursor nur mit dem Keyboard bewegst, behältst Du die volle Kontrolle über seine genaue Position im Raum/Level. Um den Cursor zu bewegen, verwende **Strg-Cursor**tasten (oder, WENN Dein Numlock aus ist, **Strg** und **2**, **4**, **6** und **8** auf dem Nummernblock). Der Cursor wird immer immer immer am Grid bleiben, sodaß Du die Gridsize ändern müssen wirst um ihn dorthin zu bekommen wo Du ihn haben willst. Und in manchen Fällen wirst Du zwischen den 2d-Ansichten wechseln müssen, um die Tiefe als auch die Höhe und Weite zu kontrollieren.

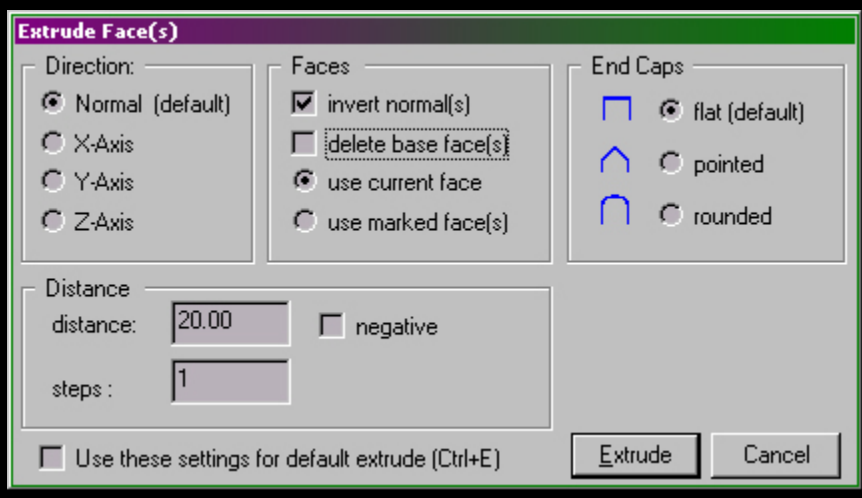
4b) Also setze den Cursor dorthin wo Du Dein erstes Vert platzieren willst, dann drücke **Ins** um es zu setzen. Bewege den Cursor zum nächsten Ort, **Ins**, Cursor bewegen, **Ins**, etc. Dies kann mehrmaliges Ändern der Gittergröße involvieren, wenn Du es originell willst. Wenn Du einen Boden machst, empfehle ich die Verts entgegen dem Uhrzeigersinne einzufügen, sodaß die Normal nach OBEN schaut, wenn das Face erzeugt wird. Gridsize im Bild ist 10.
(Anmerkung: ab D3Edit v39 ist es andersrum)



5) Nachdem Du alle Verts platziert hast, bist Du bereit das Bodenface einzufügen. Verwende **Shift-Ins** um das zu machen. Du kannst zum jetzigen Zeitpunkt den Boden textururieren, wenn Du willst.

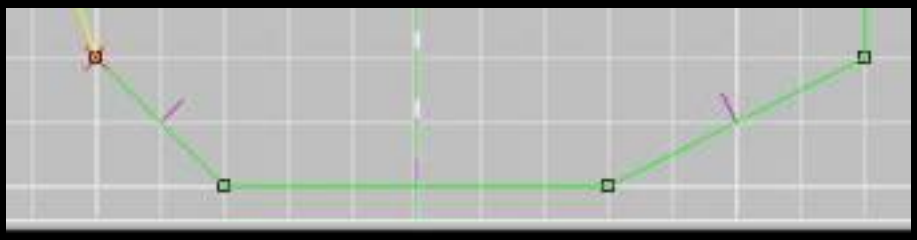
6) Nun bist Du bereit zum Extrudieren. Stelle Sicher, daß deine Normal in die richtige Richtung zeigt, wenn nicht, gehe in den Face Mode und flippe das Face vorher. Dann extrudiere vom current face, pointing inward, mit beliebiger Höhe. In diesem Beispiel habe ich 20 verwendet; bedenke, daß

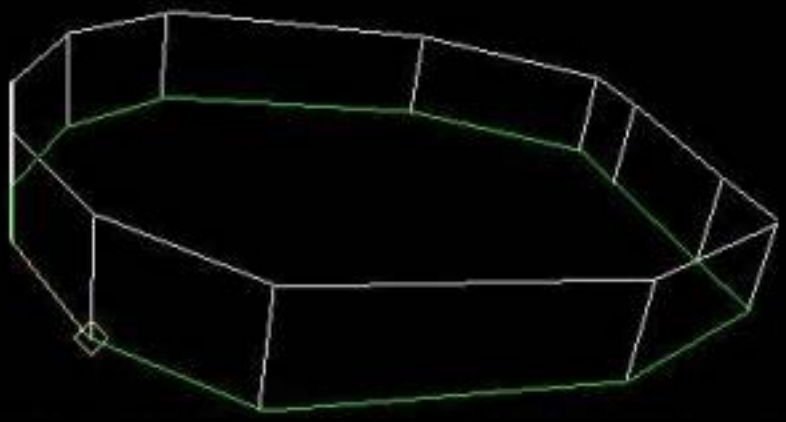




Oben: Dialog der v39, Rechts: Extrude-Dialog der v40

20 die Größe aller Texturen ist. Ein 20x20-Face paßt mit einer Textur. 20x30 sind dann eineinhalb Texturen, und mit manchen sieht das schlecht aus. Also mußt Du Deine Texturen mit deinen Raumgrößen und -formen mitplanen oder zumindest schauen, daß wenn Du die Texturen wählst (wenn Du dort ankommst, falls Du nicht vorplanst) diese gut in den Raum passen, WENN nicht alles aus netten 20x20 Faces (oder Vielfache davon) aufgebaut ist.



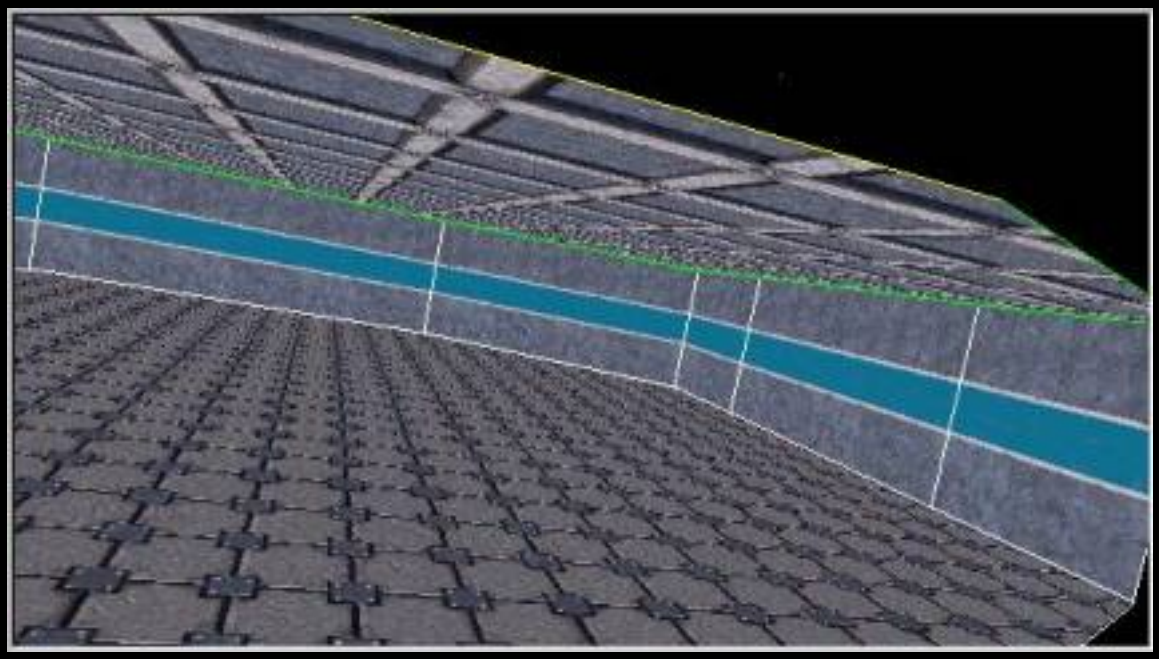


6a) Merke, daß Du in D3Edit 0.9 jedes Face händisch texturieren müssen wirst, nach der Extrusion. In neueren Versionen wird aber automatisch die current Texture aus der Texture Bar appliziert, auf alle extrudierten oder gelatheten Faces, genauso auch wenn Du Faces manuell erzeugst, also wenn Du die Textur für die Wände vor dem Extrudieren auswählst, sind sie fertig und Du mußt nur die Decke texturieren oder vielleicht Einige der Wände mit einer alternativen Textur ändern. Texturieren während des Vorankommens ist meine Methode... Ich kenne einige Andere, die es bevorzugen erst die Struktur zu bauen und später zu texturieren, aber dann wird das zur Sisyphusarbeit. Es wird weniger nervig, wenn man während des Fortschritts einige oder alle erledigt.

Hast Du erstmal extrudiert, hast Du eine komplette Shell Wenn sie die Größe hat, die Du wolltest kannst Du die Arbeit an den Dingen anfangen, die Du IN den Raum stellst (Lichter, Säulen, Glasfenster, Teilungswände, Inventar, wasauchimmer).

Merke, Du kannst Wände in der selben Ebene haben, wie die Rechte hier im Beispielraum. Dies ist keine Konkavität. Ich mache das IMMER, aus einer Reihe von Gründen welche hauptsächlich das texturieren oder die Portal-Einfassung betreffen.

Hier ist eine Ansicht der texturierten Shell. Bei Verwendung der sirianischen Methode kannst Du eine Basis-Shell fertig und texturiert in einer Minute oder zwei haben, Ohne eins der Verts zu bewegen oder später irgendwelche Shell-Texturen erneut ausrichten zu müssen.



026-sirian1.rar

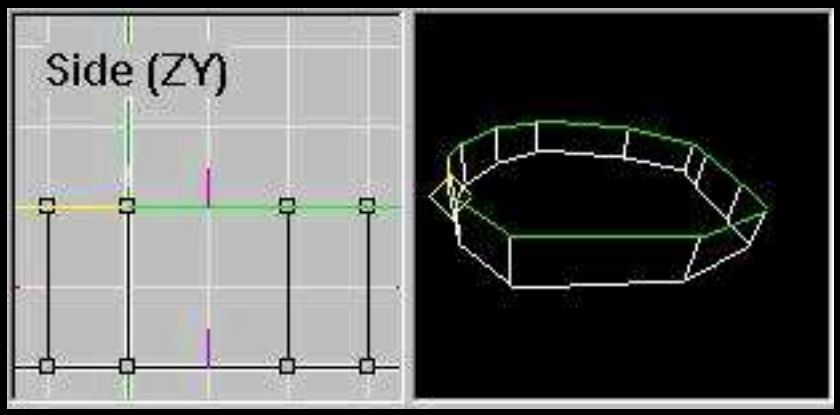
027 - Multiple Layers

Sirian

Nun da Du eine Basis-shell bauen kannst, laßt Uns zu etwas Komplexeren kommen: Layers (=Ebenen) Du kannst Layer für verschiedenste Zwecke verwenden, aber wieder schließt das hauptsächlich Texturierung und/oder Portal-Einfassung ein. Wieauchimmer, sie können auch die Möglichkeit beinhalten, Verts in Deinen Wänden zu manipulieren um Deine Shell zu ändern und abgefahren geformte Shells zu machen die nicht würfelig sind. Und das geht nicht mit einer einfachen Extrusion, weil das konkave Faces machen würde. Gegenwärtig ist es so, daß wenn Du es so machst der Editor dazu tendiert einige nonplanar Faces zu machen, aber Du kannst diese mit Triangulation fixen. Der Editor berichtet exakt welche Faces ein Problem haben, wenn Du die Verify-Funktion im File-Menü verwendest.

Wir machen dort weiter, wo wir in der vorigen Lektion geendet haben, und fahren mit dem Beispielraum, welchen wir dort erstellt haben fort. Er ist 20 Einheiten hoch. Nun bringen wir ihn auf 50 Einheiten in drei Ebenen.

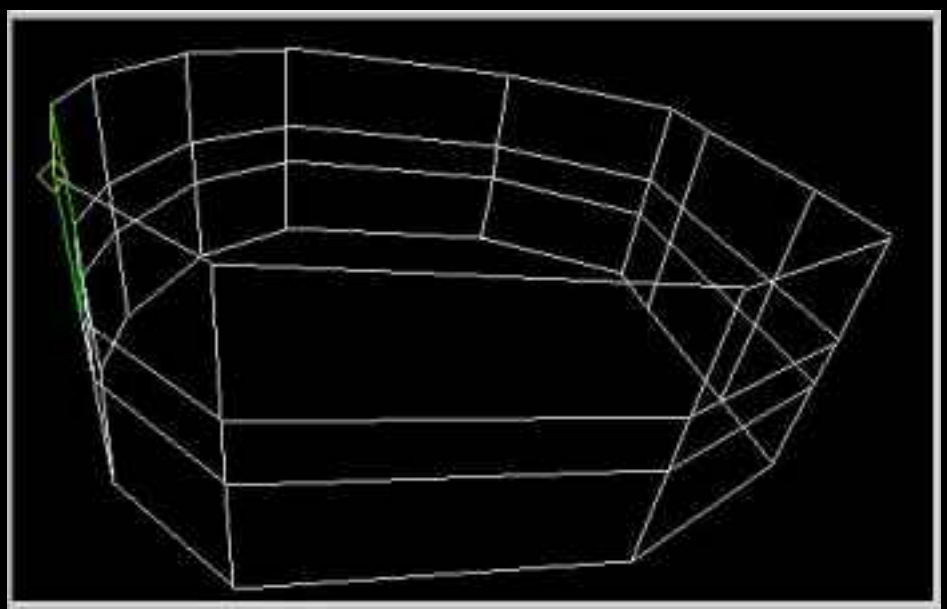
1) Das erste was Du nach der ersten Extrusion machen mußt, ist das Basisface abzumarkieren. Gehe in den Face Mode und drücke U um alles abzumarkieren. Als nächstes wähle das Deckenface. Du kannst das auf viele Arten machen, Klicken in die Top-Ansicht bis es hervorgehoben ist eingeschlossen, oder unter Verwendung der 3d-Ansicht, herumschwenken / zoomen bis Du das Face klicken kannst (seine Unterseite in diesem Fall, die Seite die auf den restlichen Raum herunterschaut)



2) Du mußt das Deckenface markieren, es wird das Basisface für Deine nächste Extrusion sein. Dann flippe das Face sodaß seine Normal nach oben zeigt, wie im Bild rechts.

3) Nun bist Du bereit erneut zu extrudieren. Mach es, und setze die richtige Höhe für Deine nächste Layer, dann extrudiere. Anmerkung: Du kannst die Box für Delete Base Face(s) after Extrusion anticken. Anderenfalls wirst Du sie manuell löschen müssen, da sie weggehören.

4) Wiederhole 2) und 3) für jede Ebene. In diesem Beispiel machen wir nur eine Ebene mehr. Anmerkung: Jede Ebene bedeutet eine Multiplikation der Gesamtzahl an Faces in Deinem Raum, Zwei Ebenen haben DOPPELT soviele Shell-Faces wie nur eine Ebene. Die meisten Ebenen ich jemals gemacht habe waren sieben, n einem Raum mit zwölf Faces per Ebene, das waren also 86 Faces *nur* für die Shell (inklusive Boden und Decke). Du mußt ein Auge auf deine Facemenge halten. Wenn sie zu groß wird, Wird Dein Raum hart für Low-End-User sein, und könnte ihn sogar für einige Spieler unspielbar machen.



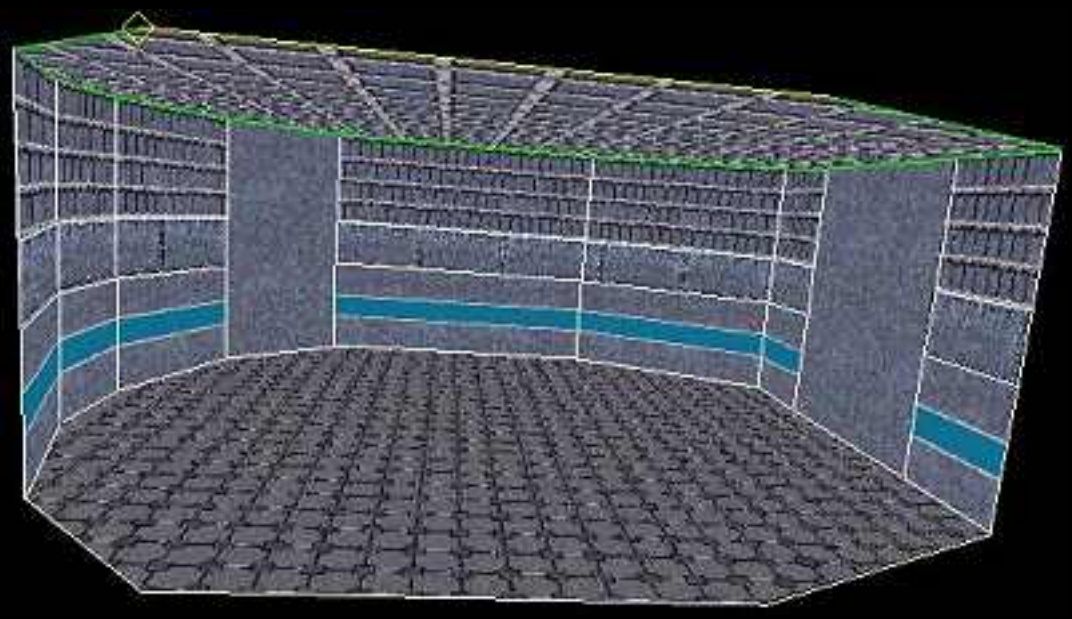
Hier ist ein Bild der fertigen Shell.

Nun hast Du einen in Ebenen aufgeteilten Raum. Er ist ein wenig klein, aber könnte als guter Wegpunkt zwischen größeren Räumen dienen, oder als ein seitlich gelegener Raum. Er ist hoch genug, daß Du ihn mit einer Vielzahl von Dingen füllen kannst, eingeschlossen Deckenlichtern, eine erhöhte Plattform, ein Altar (mit einem Powerup drauf), was Du auch hast.

Du könntest nun einerseits entscheiden daß Du ein paar vertikale Streifen haben willst um die horizontalen aufzubrechen, andererseits daß Du einen anderen Raum anhängen willst, aber nicht oben oder unten in einer Wand, sondern in der Mitte. In jedem Fall kannst Du Faces vereinigen um Dein Ziel zu erreichen.

Vereinigungen von Faces machen Dir NIEMALS T-Joints. Das ist, weil keine Verts gelöscht werden, und keine Kanten Vereinigt werden. Die Kanten, welche von beiden Faces gemeinsam genutzt werden, werden gelöscht, aber alle anderen bleiben wie sie sind, und das schützt die Integrität Deiner Shell. So kannst Du völlig frei alle Faces vereinigen die Du willst, solange dies sich nicht mit anderen Plänen überkreuzt die Du noch haben magst.

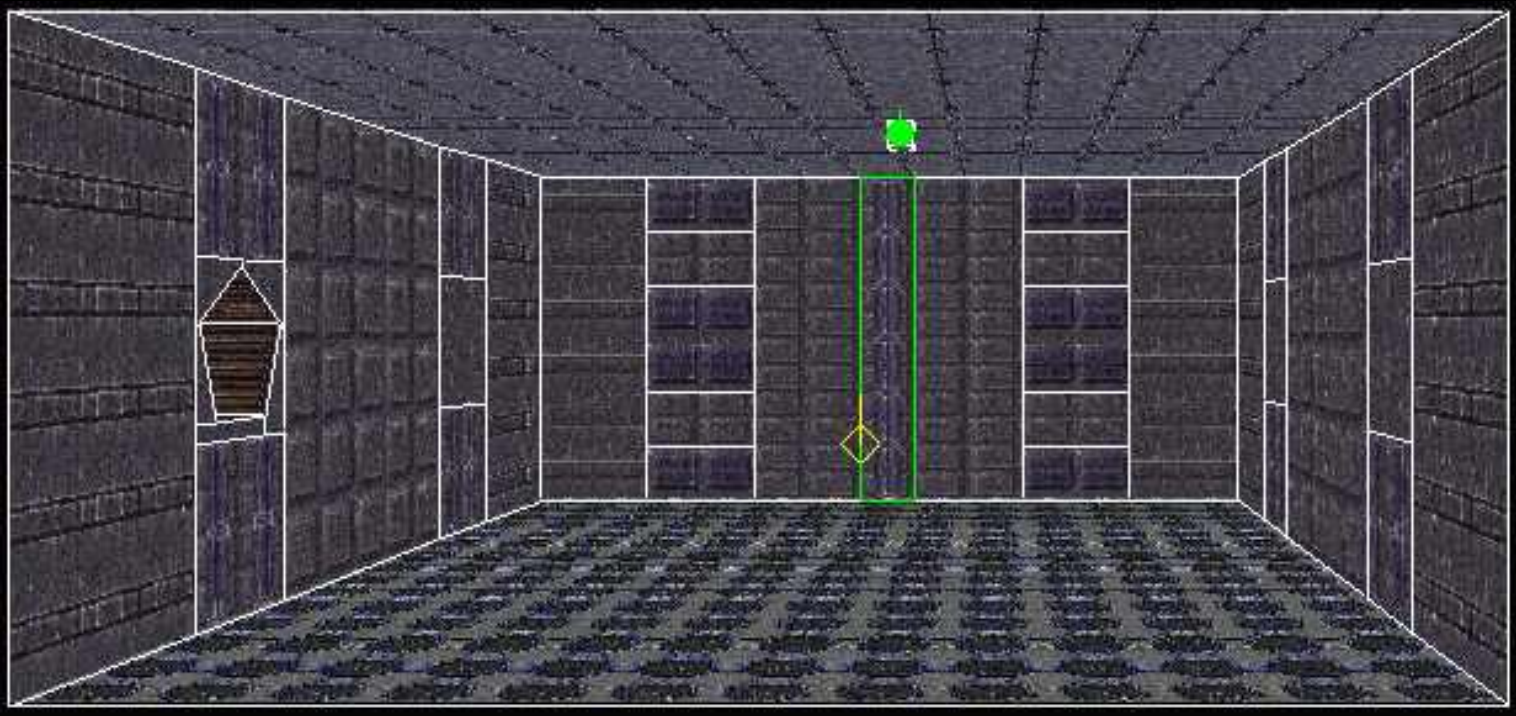
Um Faces zu vereinigen, sei als erstes im Face Mode und stelle sicher daß kein Face markiert ist. Dann wähle das Face mit jener Textur, die das neue Face dann haben soll (wenn keine behalten werden soll ist es egal welches Face Du wählst). Markiere dieses Face. selektiere das andere Face, dann wende den Button Combine Faces (unter Face) aus der Geometry-Bar an. Das neue Face bleibt markiert, sodaß Du eine ganze Reihe von Faces recht schnell vereinen kannst.



Hier ist ein Blick auf unser Beispiel, wo drei der Wände für Portale vereint wurden. Beachte daß ich sie auch unterschiedlich texturiert habe. Der Raum könnte nut als Hub dienen, der zu drei anderen Räumen führt. Natürlich braucht er noch immer Lichter und vielleicht noch andere Aufpeppung, aber die Shell ist fertig, texturiert und fehlerfrei.

In meinem Entropy-Level Siege Maximus, war der Hauptraum, der zwischen den zwei Festungen liegt, dazu bestimmt kollossale 16 Portale zu haben, welche 16 verschiedene Räume anschließen sollen. Das inkludiert vier Entropy-Basen, vier Sniper-Ports, vier Eingänge zur roten Base und vier zur Blauen. Um dies zu erreichen, habe ich eine Extrusion in fünf Ebenen gemacht, und ich hatte sieben Wandsegmente in den Nord- und Südwänden und fünf segmente in den Ost- und Westwänden. Das waren 24 Faces per Layer und 12 Faces für fünf Ebenen in meinem Raum, nachdem ich mit dem Extrudieren fertig war. Gut, ich habe nicht ALLE Faces gebraucht, Ich brauchte nur die Ebenen für die Portal-Einschließung, also war es an der Zeit zu optimieren.

Ich hatte vier Portale die in eine Ebene gingen, nochmal vier eine Ebene höher, vier mehr in der Ebene darüber, und ich wollte eine Ebene jeweils ober- und unterhalb der Portale. Also habe ich meine Layers gemacht, und begann dann Reihen von Faces zu vereinen, wo keine Portale hinkämen. Wirf einen Blick auf die Shell des Raumes hier im Bild. Dieser Screenie wurde unter Verwendung der speziellen ‚Render Shell Faces'-Option im Einstellungsdialog erstellt, wobei die Nicht-Shell-Faces abgetickt und nicht angezeigt werden. Alle Lichter und Befestigungen sind IN dem Raum, aber nur die Shell wird in dieser Ansicht angezeigt.



Beachte, daß ich in Wänden wo kein Portal sein wird alle fünf Ebenen zu einem Face vereinigt habe. Und auch eine Reihe andere Faces. Dadurch war ich fähig von 122 Faces auf 58 runterzukommen, in der Basis-Shell, indem ich Faces vereinigt habe wo ich konnte, ohne die Portaleinfassungen zu kompromittieren.

Schaue zum fünfeckigen Anhang in der Westwand. Das ist eine Entropy-Base, und da ist eine auf jeder dieser ‚Mittelebenen'-Faces sowohl in der West- als auch Ostwand. Das sind nur vier der 16 Portale, was erklärt warum die Ost- und Westwand nur neun Faces in sich haben während die Nord- und Südwand jeweils 15 haben. Siehst Du die Sprünge, die durch anhängen des Pentagon-Faces der Entropy-Base an das größere, quadratische Face des Hauptraumes verursacht wurden? Wenn diese gesamte Westwand nur ein Face wäre anstatt dieser Einfassungsstruktur die Du da siehst, wäre es unmöglich das zweite Portal mit Präzision anzuhängen, da die Sprünge die vom ersten Portal verursacht würden die gesamte Wand in ein paar Stücke splitten und neue Verts einführen würde und ALLE Normals herum bewegen würde, sodaß ich nicht einfach Place Room Attach room machen kann und fertig wäre. Aber durch designen der Wand mit einer Einfassung, ist dann wenn ich das Level assembliere, alle Räume anzuhängen ein Stück Kuchen, und ich bekomme niemals mismatched Portals. Eindämmung, Eindämmung, Eindämmung: Sie kann dein Level retten!

Nun Schau nochmals. Siehst Du das current Face, daß ich im Screenie ausgewählt habe? Es ist in grün, während ein kleiner Teil gelb ist. Das gelbe ist die current Edge des current Face, und ist der Beweis der gelayerten sirianischen Methode, welche ich verwendet habe um diesen Raum zu machen. Diese Kante ist die zweite Ebene vom Boden, und sie paßt mit einer Kante des anliegenden Faces. Wie ich sagte, könntest Du besser dran sein wenn Du Ebenen machstt und dann Faces vereinigst, als zu versuchen eine Ebene zu machen und dann einige der großen Faces in kleinere Stücke zu splitten. Und wenn Du es so machst... dann wirst Du viele T-Joints zu fixen haben.

Beachte auch, daß das current Face im Screenie selbst eine Eindämmung hat, links und rechts beiderseits, sodaß es zwischen den zwei Reihen, wo die Portale zu den vier Basen-Eingängen hinkommen (In Ebene zwei und vier), gepuffet ist. Diese Mittelreihe ist es, wo die zwei Sniper-Ports angehängt werden. Das ist es, warum ich sieben Reihen in der Nord- und Südwand gebraucht habe. Reihe eins ist ein Puffer, Reihe zwei hat zwei Portale, in Ebene zwei und vier, Reihe drei ist ein Puffer, Reihe vier hat zwei Portale, welche klein und rechteckig sind und nicht vertikal eingeschlossen werden müssen, aber horizontale Eindämmung brauchen, Reihe fünf ist ein Puffer, Reihe sechs hat zwei Portale in Ebene zwei und vier, und Reihe sieben ist ein Puffer. Whew! Hier geht viel ab aber... ich hatte Erfolg, sechs Portzale an dieser einen Wand zu haben, ohne Fehler, und ich hatte während des Prozesses ein wenig nette Texturierung zu machen.

Layered Extrusion hat ihre Anwendungen. Dieser komplexe Raum hat 16 Portale, und es ist bloß eine Box. Ich habe ihn jedoch mit ein paar Brücken und fantasievollen Lichtern ausgestattet, sodaß er nicht dermaßen fade aussieht wie in dieser Ansicht, die NUR die Shell zeigt. Aber das war das beste Beispiel das ich hatte, was Du mit Layers machen kannst, und Extrusion um Deine Shell zu machen. Weil, realisiere, daß ich diesen GESAMTEN Raum ohne Markieren und Bewegen irgendeines Verts gemacht habe. Sogar die Befestigungen habe ich komplett mit Extrude oder Lathe gemacht, oder in diesem Fall diverse Lichter, unter manueller Einfügung von Verts und Einsetzen der Faces um ein Licht zu machen, und dann die anderen Seiner Art vom Ersten zu Copy/pasten.

Du *KANNST* keinen großartigen Raum haben ohne eine großartige Shell. Lerne gute Shells zu machen, und Du bist auf halbem Wege Henker-Level zu bauen die sogut aussehen wie sie sich (so hoffe ich) spielen.

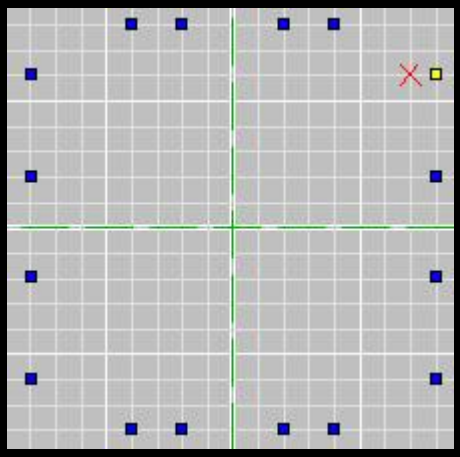
Die sirianische Methode beruht auf intelligenter Verwendung von Extrude und Lathe um Deine Shells zu machen. Du kannst eine ganze Menge auf diese Art machen, mit minimaler Arbeit, teilweise auf dem Gebiet der Textur-Ausrichtung, sodaß es schnell vorwärts geht und immer scharf aussieht.

027-sirian2.rar

028 - Multiple Extrusion

Sirian

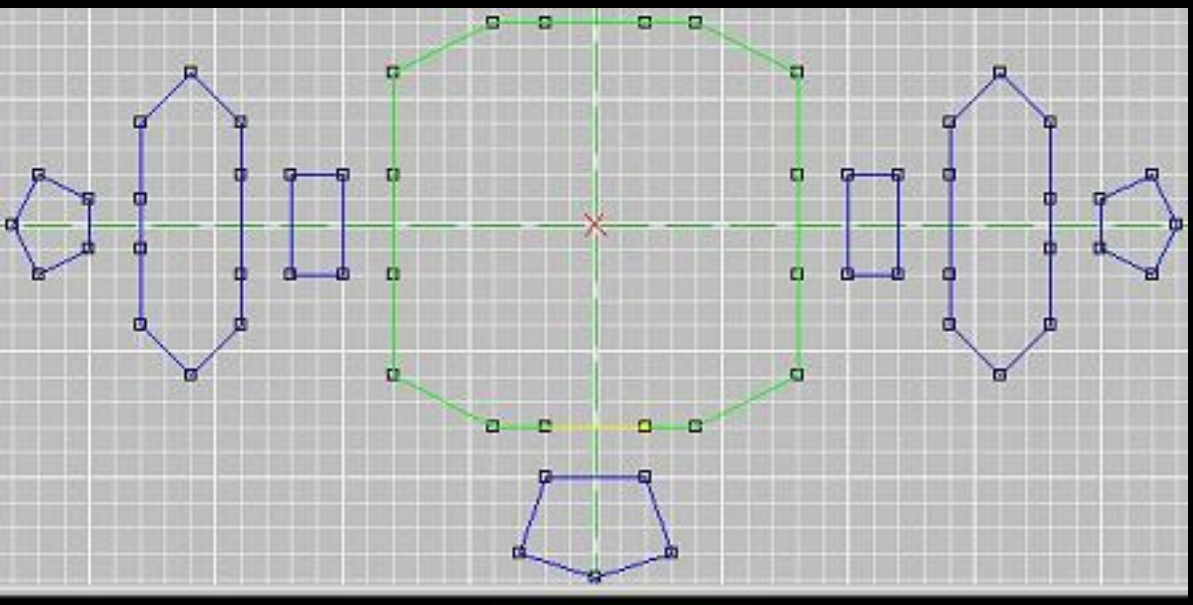
Manchmal kannst Du einfach die Shell, die Du haben willst nicht aus einem einzelnen Boden- oder Wand-Face machen. Das ist ein gefährlicher Prozess, wenn Du nicht weißt was Du machst, in dem Sinne, daß du links und rechts T-Joints erzeugst und dein Level mit Fehlern anfüllst. Du willst Deine Shells immer luftdicht und fehlerfrei haben, um das Risiko von Abstürzen oder anderen Problemen während Spielen zu eliminieren.

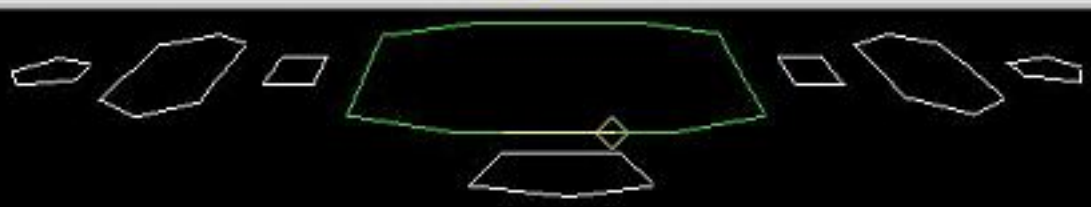


Also laß uns durch den Prozeß gehen, eine fehlerfreie Shell mit multipler Extrusion zu machen. Du mußt sorgfältig planen, sodaß Du keine T-Joints produzierst, welche Löcher in deiner Shell lassen. Guck Dir dieses Vertex-Muster an (rechts).

Nun mache ich die anderen Faces separat, **nur** um die Teile zu zeigen und wie Du die Kanten und Verts anpassen mußt

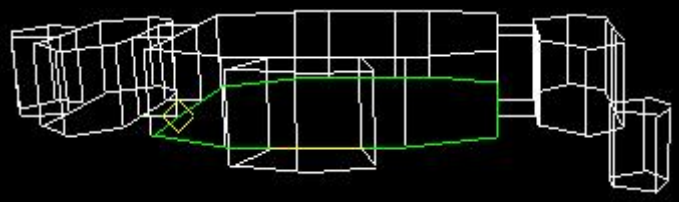
Der Grund warum Du soviele Faces haben mußt ist daß du keine konkaven Faces haben darfst. Also mußt Du über die Form die Du haben willst nachdenken, und dann den Boden in sowenige Faces wie möglich aufgliedern, ohne ein konkaves zu haben. Beachte auch die gelbe Kante auf dem current Face: diese Kante MUSS exakt mit der Kante des Pentagon-Faces passen, da dies eine gemeinsam benutzte Kante im Boden ist, und bevor es nicht exakt paßt hast Du T's und eine Bad Shell.



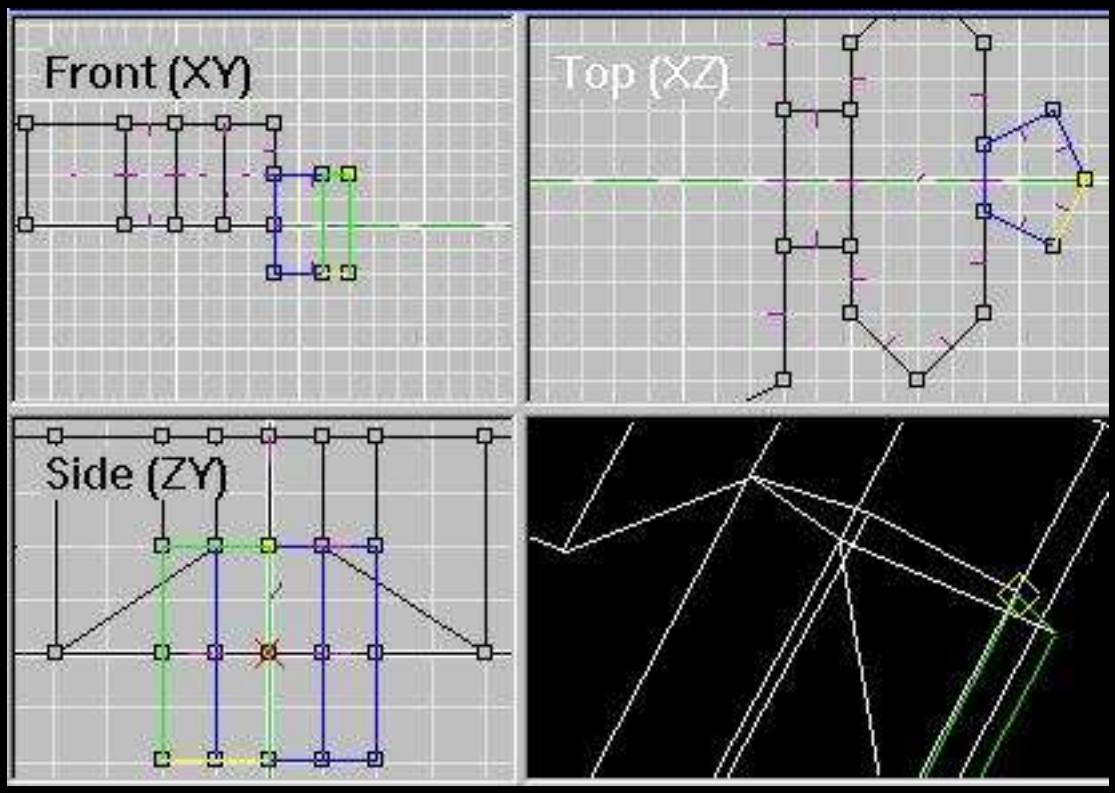


Nun gehe ich daran alle diese Faces zu ihren richtigen Positionen zusammenzubewegen und zu extrudieren. (Siehst Du das Gitter? es ist ein 10er Grid, nicht 5. Du kannst in den Gittern mit dem **Mausrad** rein- und rauszoomen bzw. mit **Shift-Mausklick** schwenken)

In aktuellen Versionen des D3Edit kannst Du von allen Faces aufeinmal extrudieren, indem Du ‚Extrude from: Marked Faces' im Extrude-Dialog anwählst. Dann peng, es werden alle in einem Rutsch extrudiert, auf die selbe Länge. Wenn Du noch immer mit einer 0.9 Beta arbeitest, wirst Du sie einzeln extrudieren müssen.

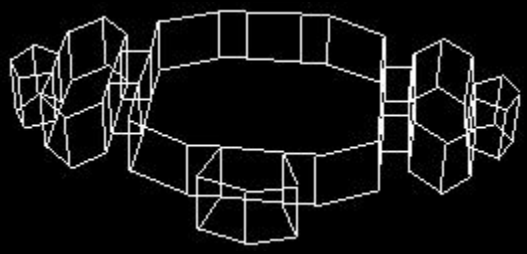


So sollte es nach dem Extrudieren aussehen. Siehst Du den Block ganz rechts? Er ist runter- anstatt raufextrudiert worden. Laß Dir das nicht passieren! Stelle immer sicher, daß du deine Boden-Verts entgegen dem Uhrzeigersinn (Anm: im Uhrzeigersinn bei 0.1.39 beta 10) einsetzt, wenn Du Boden-Faces machst, sodaß deine Normals nach oben schauen. Und prüfe es bevor Du extrudierst, daß keine Normal nach unten zeigt. Wenn Du doch eins haben solltest, mußt Du ALLE Faces abmarkieren, diejenigen markieren die falschrum schauen, flippen, und die Bodenfaces erneut markieren. Yum.

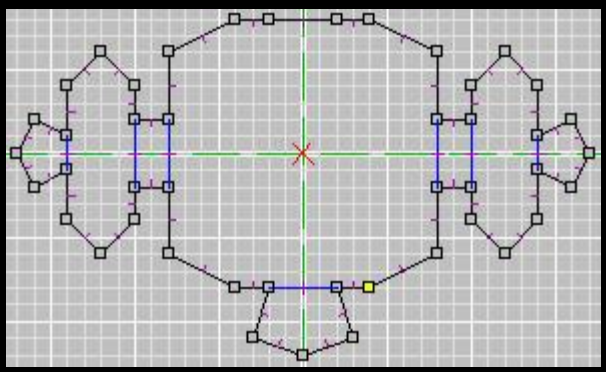


Es ist ebenso eine gute Idee, die Verts nicht zu verbinden vor der Extrusion. Dazu ist genug Zeit, wenn die Gestalt erst einmal paßt. Wenn Du die Verts verbindest, kannst Du die Blöcke nicht unabhängig voneinander bewegen. Sie werden zusammengehängt, und einen zu bewegen erzeugt Verzerrungen in anderen, wie hier:

In aktuellen Editor-Versionen kannst Du nun markierte Faces im Face-Mode herumbewegen. Also KÖNNTE die Verschiebung das Problem gelöst haben, WENN die Verts nicht verbunden worden wären. Also entferne keine Extra-Vertices, bis die Shell fertig ist, und DANN mach das. Und vergiß nicht drauf! Du kannst massive Fehler erzeugen wenn du dabei versagst die Verts zu verbinden um diese geteilt genutzten Kanten zu versiegeln.



Laß uns nun den Raum anschauen, wie er gemeint war: Wir sind aber noch nicht fertig. Da sind einige Faces zwischen den Segmenten, welche entfernt werden müssen um das Ding zu einem einzelnen Areal zu öffnen. Du kannst das am besten in der Top-Ansicht machen. Stelle zuerst sicher, daß Du im Face Mode bist, dann markiere alle Faces ab, und dann verwende die Maus um jene Faces zu markieren die weg müssen. Ziehe dazu einen Rahmen über die Naht, einer nach der anderen. Wenn Du alle Nähte markiert hast, sollte es etwa so aussehen (links):



Stelle sicher daß Du nicht versehentlich eines der Bald-Shell-Faces markierst. Du willst nur die Nähte zwischen den extrudierten Segmenten löschen. Sind erst mal alle markiert, drücke **Entf** oder wähle Delete aus dem Edit-Menü.

JETZT ist Deine Shell fertig, und Du solltest ‚**Remove Extra Verts**' machen um all die Kanten zu versiegeln.
Und da Bist Du:
Ein nicht-trivial aussehender, nicht-würfeliger Raum komplett mit Extrude gemacht.

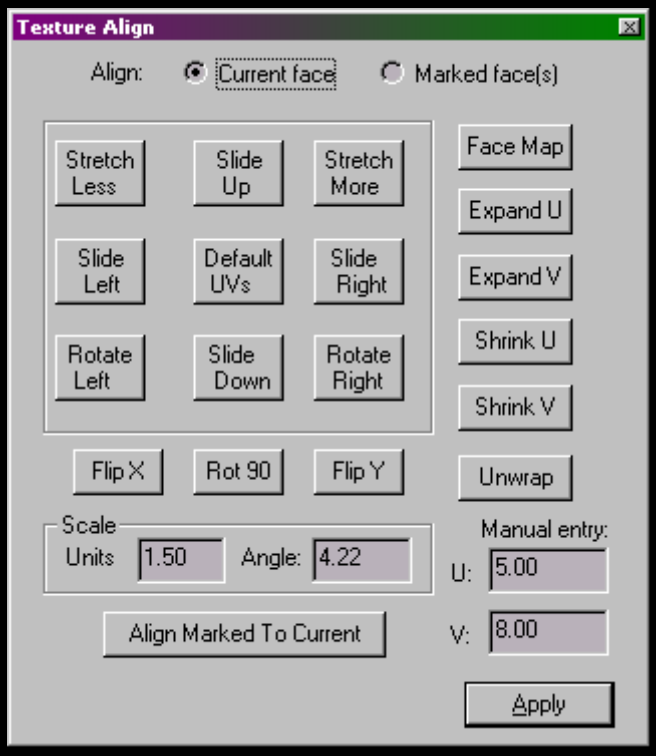
Texturiere wie Du willst. Dann ist Deine Shell komplett und fehlerfrei, und Du kannst beginnen sie mit Non-Shell Faces zu bevölkern.

Eine andere Anmerkung:
Dieser Beispielraum ist *NUR* ein Beispiel. Ich würde eigentlich dieses Layout nicht in dieser Art machen, wenn ich es in einem Level verwenden wollen würde. Ich würde das in drei Räumen haben wollen, da diese Flügelbereiche an den Seiten keine Sichtlinien zu den meisten andern Räumen hätten. Sie sollten Räume für sich sein. Also sollte dieser Beispielraum eigentlich drei Räume sein: Die Mitte und die beiden Flügel. Und das würde Dir auch ein wenig Zeit sparen, weil Du einfach einen Flügel als eigenen Raum erstellst und diesen Raum zweimal verwendest wenn Du Deine `.orf`'s zusammensetzt um das `.d3l` zu machen. Nun, das ist nicht der Hauptgrund, diesen Raum in drei aufzubrechen. Der Hauptgrund ist, weil es das Beste für die Vis Table und die Game-Engine wäre, und die Performance Deines Levels optimiert. Aber das Fakt daß es auch weniger Arbeit bräuchte, ist ein netter Bonus.

Addendum - Textur-Ausrichtung
Mit dem 0.9 Beta Editor ist das Texturen ausrichten ein ermüdender Prozeß. Du wirst die Faces händisch ausrichten müssen, eins nach dem anderen, wenn du nach dem Extrudieren Dinge erst wieder-ausrichten mußt. Eine Sache Die Du in diesem Fall machen kannst, ist ein einfacher Step-By-Step-Prozeß um die Arbeit zu verringern:

1) Wähle ein Face zum Ausrichten.
2) Klicke den mittleren Button um das Face zu ‚Default UV's' zu resetten (dies eliminiert den werfenden, verdrehenden Effekt den Du bekommst wenn Du Die Verts bewegst oder wenn Du bendest – wenn Du nicht willst, daß sich die Texturen mit den Verts mit-benden, mußt Du sie anschließend rücksetzen.)
3) Richte die Textur aus, begutachte sie in der 3d-Ansicht eher in der texturierten Ansicht als mit Wireframe. Beachte die U- und V-Werte.
4) Gehe zu einem ähnlichen Face, das auf exakt die selbe Weise ausgerichtet werden muß, setze die UV's mit dem mittleren Button zurück, dann gib die Werte vom Original (die Du aufgeschrieben hast) ein. Diese Face wird nun zum Ersten passen, daß Du gemacht hast.
5) Sooft als nötig wiederholen.

Im aktuellen Editor gibt es alle Arten von Ausrichtungs-Tools die Du verwenden kannst. Hier ist wie man sie verwendet.
Align: Current face oder Marked face(s): Dies ist ein Schalter. Align current richtet nur das aktuelle Face aus. Aber Du kannst den Prozeß, den ich oben für den 0.9 erwähnt habe, nun auf schnellere Art verwenden: Du markierst jedes einzelne Face, das Ausrichtung braucht, dann schalte um auf Align Marked, und wenn Du eins ausrichtest, werden sie alle ausgerichtet. Hübsch, nicht wahr?
Und Du kannst den Align Marked dazu zu verwenden, um einen ganzen Raum in wenigen Momenten zu resetten: Markiere alle Faces, Reset marked, abmarkieren: Peng, Du hast fertig. Keine welligen, tanzenden, blasigen, verworfenen Texturen überall mehr (Wenn Du es magst, Verts viel herumzubewegen.)



Align Marked To Current: Du kannst einen ganzen Streifen von Faces auf einmal ausrichten (z.B. Alle Wände einer Ebene eines Raumes: sagen wir, Du hast viele Nicht-20x20-Faces und Du willst daß es professionell aussieht...) Also was Du machst, ist daß Du ein ‚Quell'-Face auswählst, alle markierst welche Du daran ausgerichtet haben willst, und dann auf das Quell-Face klickst sodaß es current wird, dann klicke Align Marked to Current. Dies richtet alle markierten Faces aus, sodaß alles glatt aussieht. All die markierten Faces müssen aneinanderliegend sein, obwohl sie sich gemeinsame Kanten teilen.
Ctrl-Klick: Du kannst Texturen manuell neu ausrichten, entweder im Textured Mode oder im Textured With Outline Mode (ich persönlich bevorzuge mit Outline). Du wählst das Quell-Face als dein current aus, und dann **Ctrl-Klick**e auf ein anliegendes Face (das sich eine Kante mit dem current Face teilt) und die anliegende Face-Textur wird zur current hin ausgerichtet. Das neu ausgerichtete Face wird automatisch zum current Face, so daß Du um einen ganzen Raum herumgehen kannst, oder Wand, oder Boden, oder gelathetem Objekt, oder Wasauchimmer, in relativ flotter Manier, immer im Blickfeld habend und die Änderungen in ECHTZEIT zu sehen während Du vorwärtskommst. THIS IS AN AWSOME FEATURE! Stelle sicher daß du weißt wie man es einsetzt, da seine Mächtigkeit atemberaubend ist. Du kannst sogar über Räume hinweg **Ctrl-klick**en. Traurigerweise kann man das nicht zwischen `.orfs`, aber zwischen Räumen in einem zusammengestellten `.d3l`... was es Dir erlaubt, ganze Böden und Decken und Wände in Deinem Level. Ich habe dieses Feature ausgiebig in „Divided We Fall" getestet, und Gwar kommentierte nachdem er es gesehen hatte, *„Das wird einen komplett neuen Standard für Texturenausrichtung setzen."*

Der Grund für diesen Teil der Übung ist einfach: die Böden und Decken im Beispielraum sind fehlausgerichtet. in 0.9 ist das ein Ärgernis, aber im aktuellen Editor machst du nur ein paar **Ctrl-Klick**s und es wird nahtlos aussehen.

Soon these tools will be in your hands.

`028-sirian3.rar`

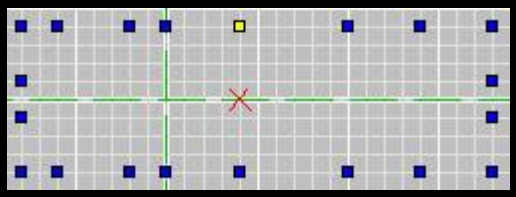
029 - Multiplanare Extrusionen kombinieren

Sirian

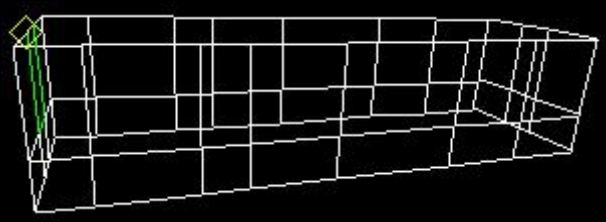
Nun werden wir uns in das Land der T-Joints wagen und (so hoffe ich) unversehrt wieder herauskommen. Startest Du erst einmal entlang verschiedener Achsen zu extrudieren und die Ergebnisse zu verbinden, bist Du prädestiniert für T-Joints und Bad Shells, außer Du nimmst GROSSE Qualen auf dich um sie zu vermeiden. Und da Du die in der 0.9er Version des Editors nicht einfach reparieren kannst, wirst Du sie dann unter ALLEN Umständen vermeiden wollen. In aktuellen Versionen kannst Du Verts zu Faces und dessen Kanten hinzufügen, was es Dir erlaubt T-Joints zu richten, aber das braucht immer noch Arbeit und ein wenig einschlägige Erfahrung, also zeige ich Dir einen Weg dies zu machen ohne irgendwelche T's zu erzeugen.

Laß uns mit dem Vert-Pattern anfangen daß Du für Portal-Eindämmungen brauchen wirst. Wir machen einen Raum mit vier drangehängten Portalen: eins in der Westwand, zwei in der Nordwand und eine gewinkelte Erweiterung zum Süden hin, die zu einem anderen Portal führt.

Wir werden drangehen die vorigen Lektionen zu kombinieren, da wir einen Raum in Ebenen machen aber mit multiplen Extrusionen, die aus unterschiedlichen Richtungen kommen. Fange mit dem Containmentpattern in der ‚primären' Ebene an:

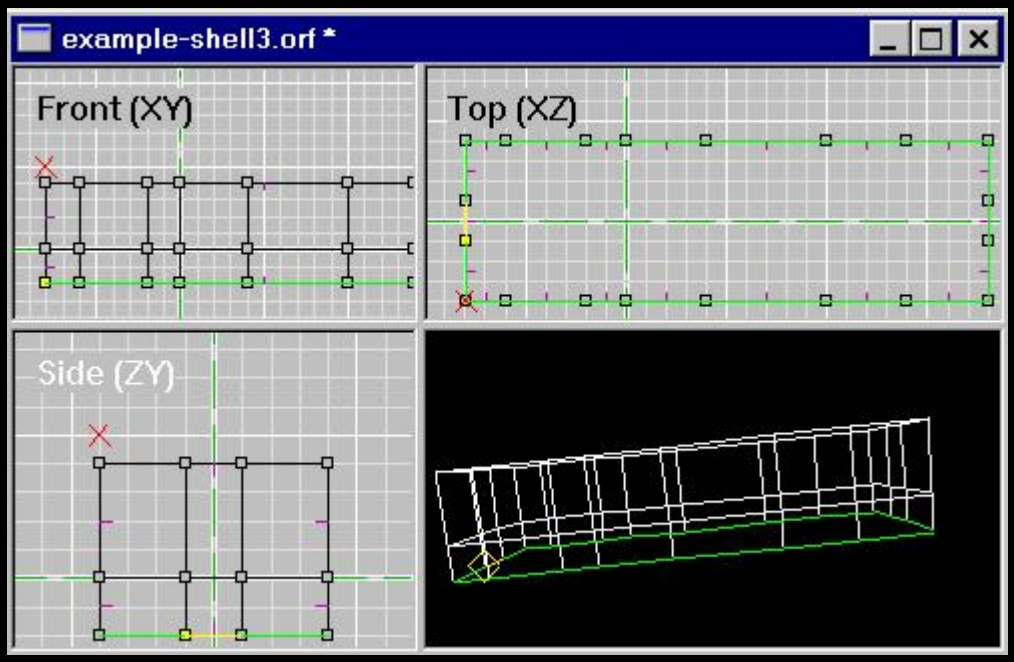


Nun, der Schlüssel zur Vermeidung von T-Joints ist es, Deine Verts zu spiegeln. Wenn Du nur ein Vert auf einer Seite brauchst, schön, aber Du mußt ein gegenüberliegendes Vert auf der anderen Seite platzieren, auch wenn Du es nicht brauchst. Ich möchte ein Portal zwischen den Verts zwei und drei als auch zwischen den Verts vier und fünf an der Südwand. Also technisch betrachtet, brauche ich die Verts vier und fünf an der Nordmauer ebenso nicht wie die Verts zwei, drei, sechs und 7 an der Südwand, aber ich mußte sie spiegeln, auf jeder Wand. Warum, das wirst Du gleich sehen.
Beachte daß dies ein 10er Gitter ist, und ich zu einer weiten Ansicht rausgezoomt habe so daß das ganze Ding betrachtet werden kann. Also alle zwei Gitterpunkte repräsentieren 20 Einheiten, was die Größe einer Textur-Bitmap ist. Wenn Du in 20er-Einheiten-Blöcken arbeitest, hast Du's leicht beim Texturieren. Wenn Du beginnst abseits der Winkel zu arbeiten, mußt Du Deine Texturen ausrichten, oder der Level wird Scheiße aussehen.



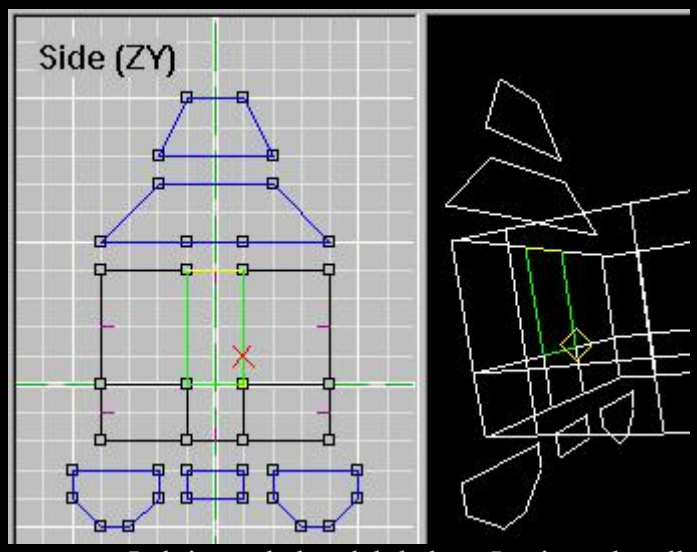
In diesem Bild habe ich das Anfangsface eingefügt, 40 Einheiten nach oben extrudiert, den Boden geflippt und 20 Einheiten nach unten extrudiert um eine zweite Ebene darunter zu machen. Nach dem Löschen der Nähte zwischen den Ebenen habe ich den ‚Körper' des Raumes fertig, und fürs erste die Extrusionen in der Top-Ansicht.

Nun ist es Zeit in die Seitenansicht zu gehen und die Extrusion von dieser Ebene vorzubereiten. An diesem Punkt MÜSSEN Wir mit dem Koordinaten-Cursor arbeiten und eine gute Kenntnis unseres Aufenthaltsortes in allen drei Dimensionen.
Dies ist ein Bild, wo der Cursor zu den exakten Koordinaten bewegt wurde, wo ich beginnen werde neue Vertices zu platzieren. Du mußt den Cursor zur korrekten Weite und Höhe in der Seitenansicht setzen, und dann in einer der anderen 2d-Ansichten nachgucken ob die Tiefe paßt. Wenn Du NICHT bei der richtigen Tiefe bist, klicke in eins dieser Fenster und bewege den Cursor mit **Ctrl-Cursor**tasten. WICHTIGE ANMERKUNG: das Klicken in ein aktives Fenster *bewegt* den Cursor. Das Klicken in ein inaktives Fenster aktiviert es lediglich. Also mußt Du ein Auge darauf haben, welche der 2d-Ansichten Du aktiv hast, falls überhaupt eine. Vertraue mir, wenn Du's nicht schon weißt, findest Du bald mehr über die Frustration heraus wenn man versehentlich in den falschen Platz zur falschen Zeit klickt und dann den Cursor manuell wieder zurücksetzen muß.



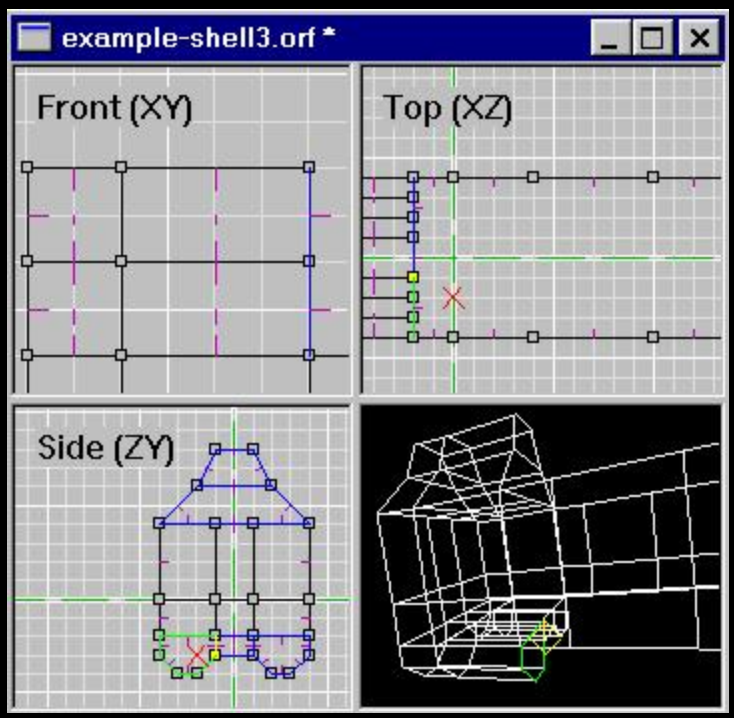
Ist der Cursor erst einmal am richtigen Gitterpunkt in allen drei Dimensionen, brauchst Du die Seitenansicht aktiv, denn da werden wir unseren nächsten Schritt ausführen.

Vert einfügen, Ctrl-Cursor um den Cursor zum nächsten Punkt zu bringen, Vert einfügen, wiederholen. Du kennst den Drill. Nur jetzt kommst Du mit Schummlerei und Verwenden der Maus nicht davon: Du *mußt* den Cursor verwenden, da wir nicht länger im Refernzrahmen arbeiten (Den Bezugspunkt für Kleinigkeiten wie das hier zu versetzen ist es meiner Meinung nach nicht wirklich wert – kann mehr Probleme machen als es löst). Das linke Bild zeigt das komplettierte Face-Muster. Wie Du in der 3d-Ansicht sehen kannst, sind sie mit der Westwand an einem Ende des Raumes ausgerichtet. Ich habe sie auseinandergezogen, sodaß Du jedes einzelne Face sehen kannst. Normalerweise hast Du keinen solchen Abstand dazwischen.

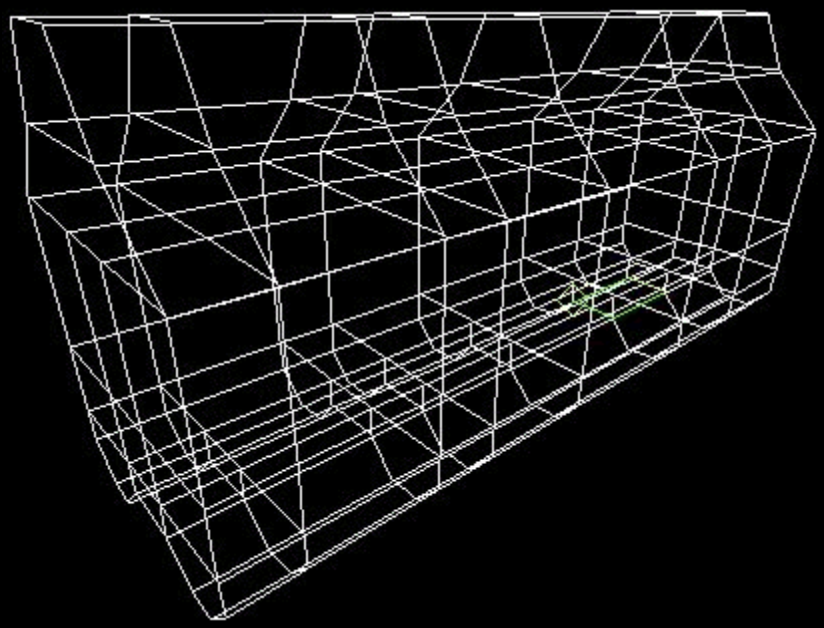


Als nächstes mußt Du ihre Normals prüfen, stelle sicher daß alle in die richtige Richtung schauen, dann ist es Zeit eine gelayerte multiple Extrusion zu machen, das kombinierend was wir in den ersten zwei Lektionen behandelt haben. In einer aktuellen Editor-Version kannst Du alle Faces aufeinmal extrudieren. In der alten 0.9er Version mußt Du die Extraarbeit machen, sie einzeln zu extrudieren, und in diesem Falle *KÖNNTEST* Du die Faces wirklich separieren wollen, wie ich das hier zu Anschauungszwecken gemacht habe, um einfacheren Überblick über Dein Fortkommen zu haben.

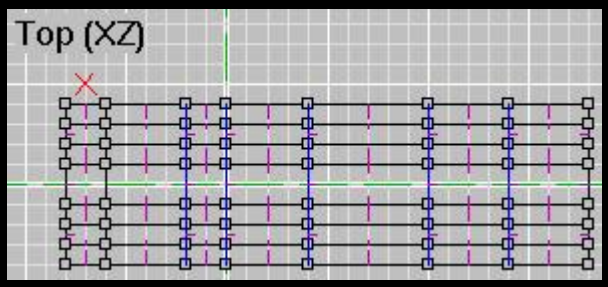
Hier kannst Du sehen, daß ich zwei Ebenen Extrudiert habe. Du *mußt* die Extrusionstiefe mit den Verts der Wände des Raum-‚körpers' passend haben. Das ist das Geheimnis beim Vermeiden von T-Joints: viel Planung. Wenn Du hinter ein Wandvertex extrudierst, hast Du ein T. Wenn Du zu einer Länge extrudierst wo kein Wand-Vert ist, dann hast Du ein T. Wenn Du die Länge des Raumes in einem Rutsch extrudierst, hast Du eine Schiffsladung voll T's. Das kannst Du dir in der 0.9er Version nicht leisten. Nun sind die eigentlichen Schritte beim Durchführen der Extrusionen die selben, die ich Dir in früheren Lektionen gezeigt habe. Du markierst alle Faces ab, dann markierst Du die welche Du extrudieren willst, flippst sie gegebenenfalls, extrudierst die richtige Länge, und das wiederholst Du, bis Du durch die Länge des Raumes marschiert bist. Beachte, daß Du einfach die gesamte Face-Gruppe in der Top-Ansicht markieren kannst, nur klicken



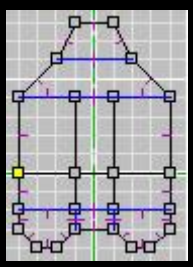
und ziehen, und ohne Gefahr von ungewollten Markierungen, außer Du wirst wirklich wirklich schlampig. Beachte auch, daß ich im Bild bereits die nächste Face-Gruppe markiert und bereit zum Extrudieren habe, und sie wurden auch schon geflippt (siehst Du die Normals?)



Hier links siehst Du wie es aussieht wen das Extrudieren erledigt ist. Nun müssen wir hingehen und alle Nähte löschen, und wir müssen das in zwei Schritten machen.

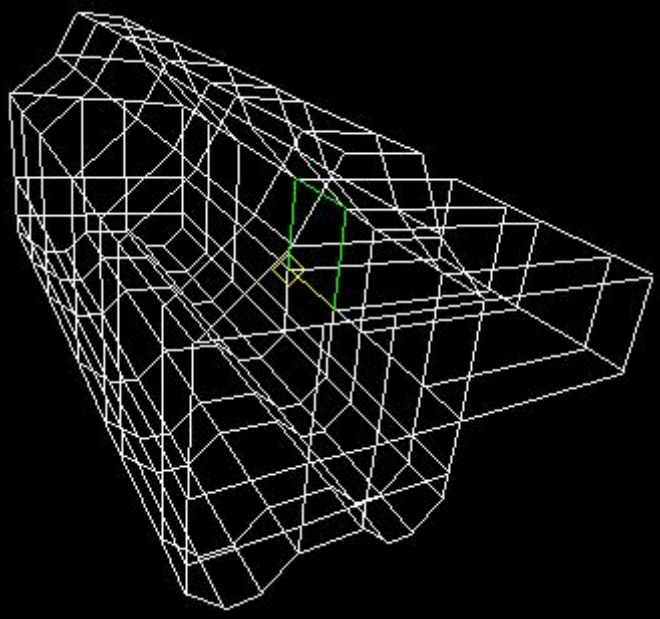


Hier habe ich die Nähte markiert welche in der Top-Ansicht sicher markiert werden können. Da sind welche die nicht markiert sind, aber die habe ich bereits früher gelöscht. Zeit diese zu löschen.



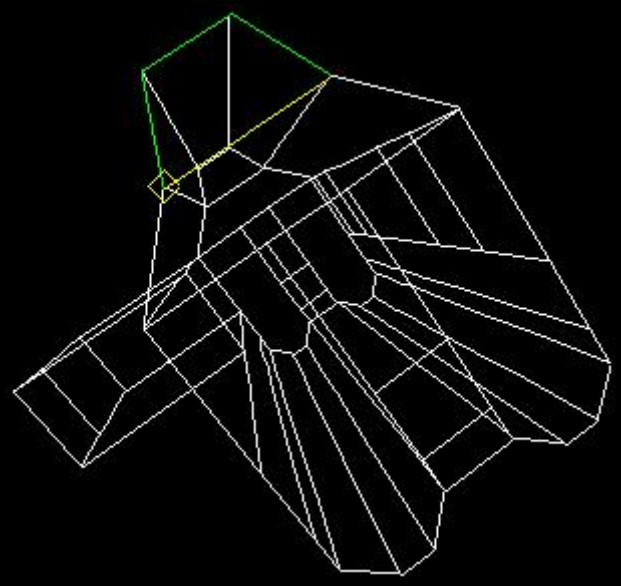
Und hier links kannst Du die verbleibenden Nähte sehen, welche in der Seitenansicht markiert werden. Sind diese erst mal gelöscht, haben wir diese Stufe des Prozesses abgeschlossen.

Nun gehts an die gewinkelte Sektion die ich früher erwähnte. Wir werden für diese Stufe von der Front-Ansicht extrudieren. Wieauchimmer, diesesmal werden wir nicht neue Faces machen um davon zu extrudieren, sondern bereits existierende Faces verwenen, die Teil der Südwand sind. Diese Faces müssen markiert, dann geflippt werden, dann kann die Extrusion stattfinden. Etwa so:

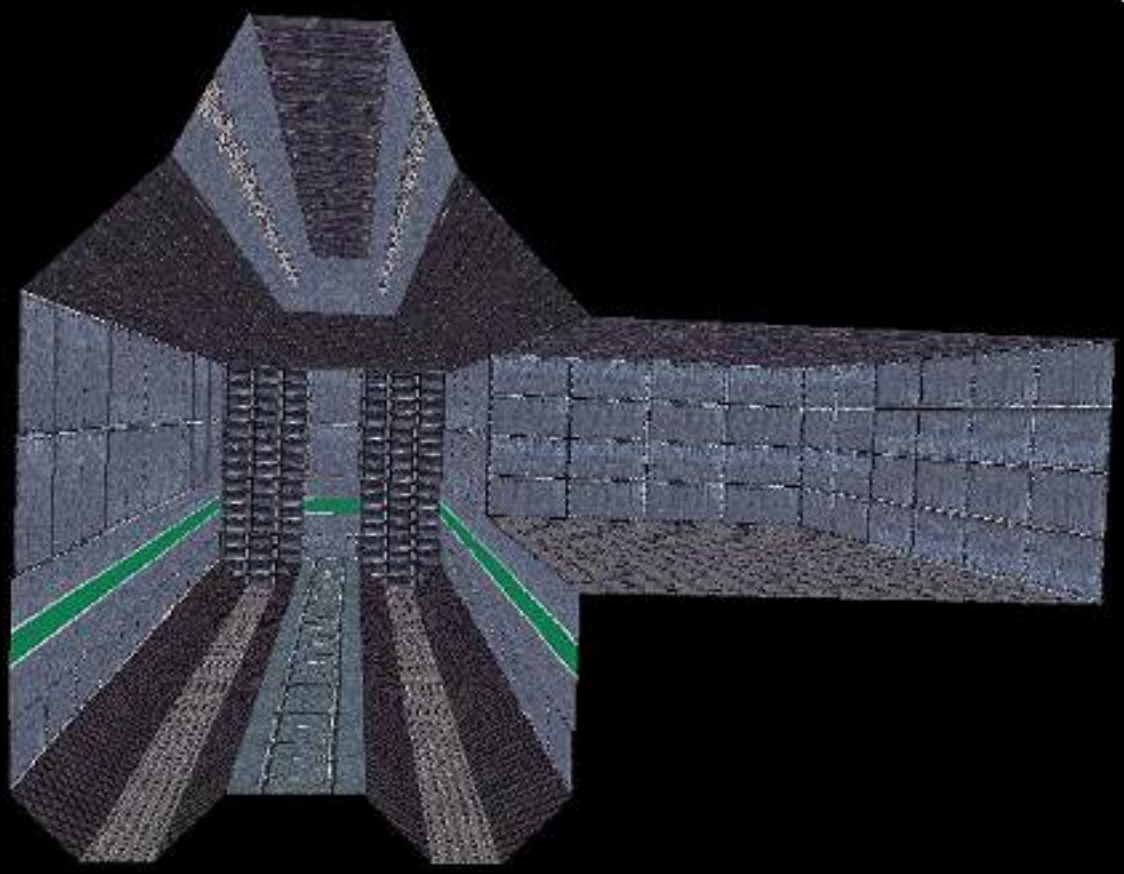


Nach der Extrusion mußt Du die Nähte löschen. Da sind nicht viele, aber Du kannst in die texturierte Ansicht wechseln um sicherzustellen, daß Du keins übersehen hast.
Nun haben wir eine Shell, und ungeachtet der Tatsache daß wir von elf verschiedenen Faces in drei verschiedenen Richtungen extrudiert haben, wir haben ÜBERHAUPT KEINE T-JOINTS. Das ist der ‚sichere' Weg, um mit der sirianischen Methode komplexe Operationen Durchzuführen – in D3Edit 0.9 der einzige Weg überhaupt. JETZT, da alle Shell-Faces erzeugt sind und alle Naht-Faces aus dem Innenraum gelöscht wurden JETZT ist die Zeit um die Extra-Vertices zu entfernen, sodaß diese Kanten an all diesen Faces verschmolzen werden und deine Shell luftdicht wird!

Defakto ist es nötig die Verts nun zu mergen, sodaß wir zum nächsten Schritt kommen können. Und was ist der? Ein Texturschema zu wählen und dann wo es möglich ist Faces zu vereinigen um den Polygon-Count für den Raum niedrig zu halten und die FPS zu erhöhen.
Der Grund, warum Du Dein Texturschema zuerst wählen solltest ist weil es eine königliche Pein ist ein Face splitten zu müssen, daß Du nicht vereinigen hättst sollen. Texturiere den Raum, dann schau welche Faces Du sicher vereinigen kannst, unter Achtnahme auf die Texturen als auch auf die Portal-Einfassungen. Das kam bei mir heraus, nach sieben Minuten Faces vereinigen:



Und in texturierter Ansicht:



Nochmals, dies ist der ‚Sichere' Weg um Multiplanar-Extrusionen zu kombinieren, der einzige Weg es zu machen ohne T-Joints zu erzeugen.

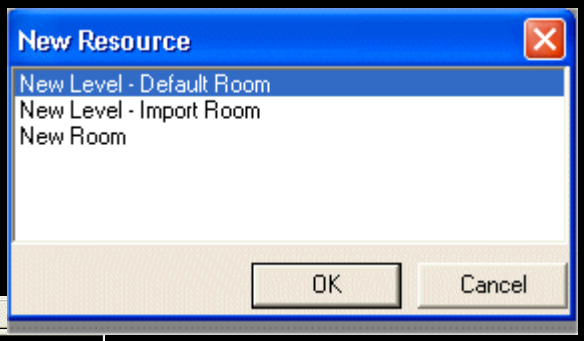
029-sirian4.rar

030 - Raum Teilen

Atan

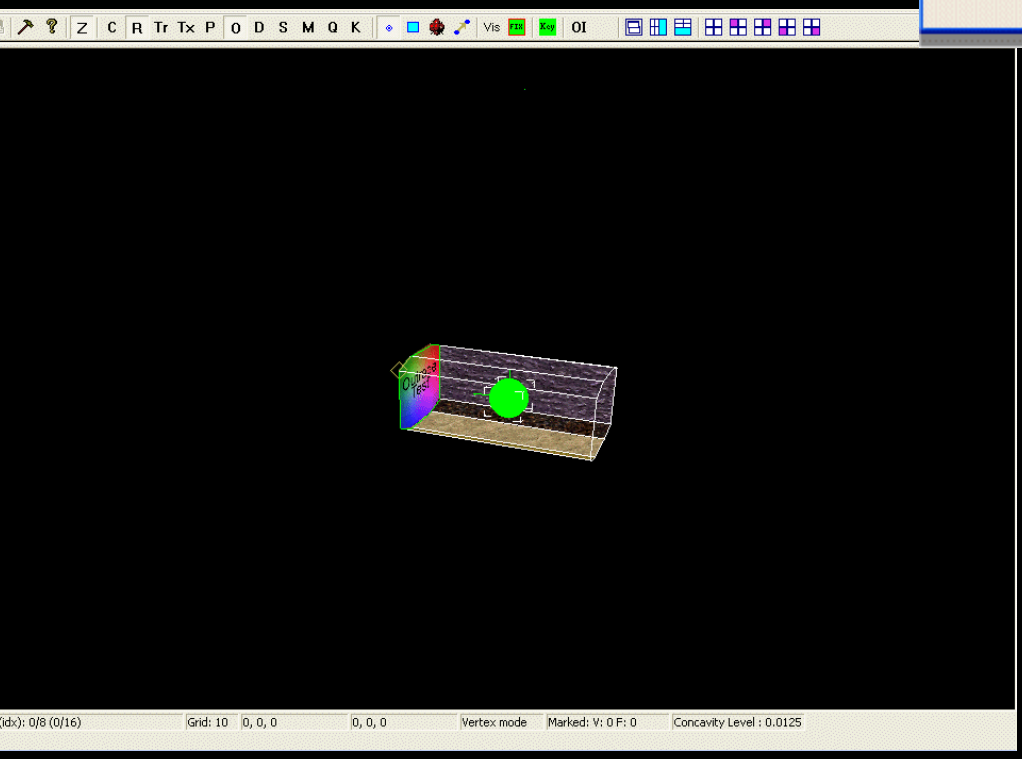
v1.0

Starten wir den Editor und erzeugen einen Raum:



Das Ergebnis sollte etwa so aussehen:

Das Player Objekt im Raum brauchen wir nicht weiter zu beachten.



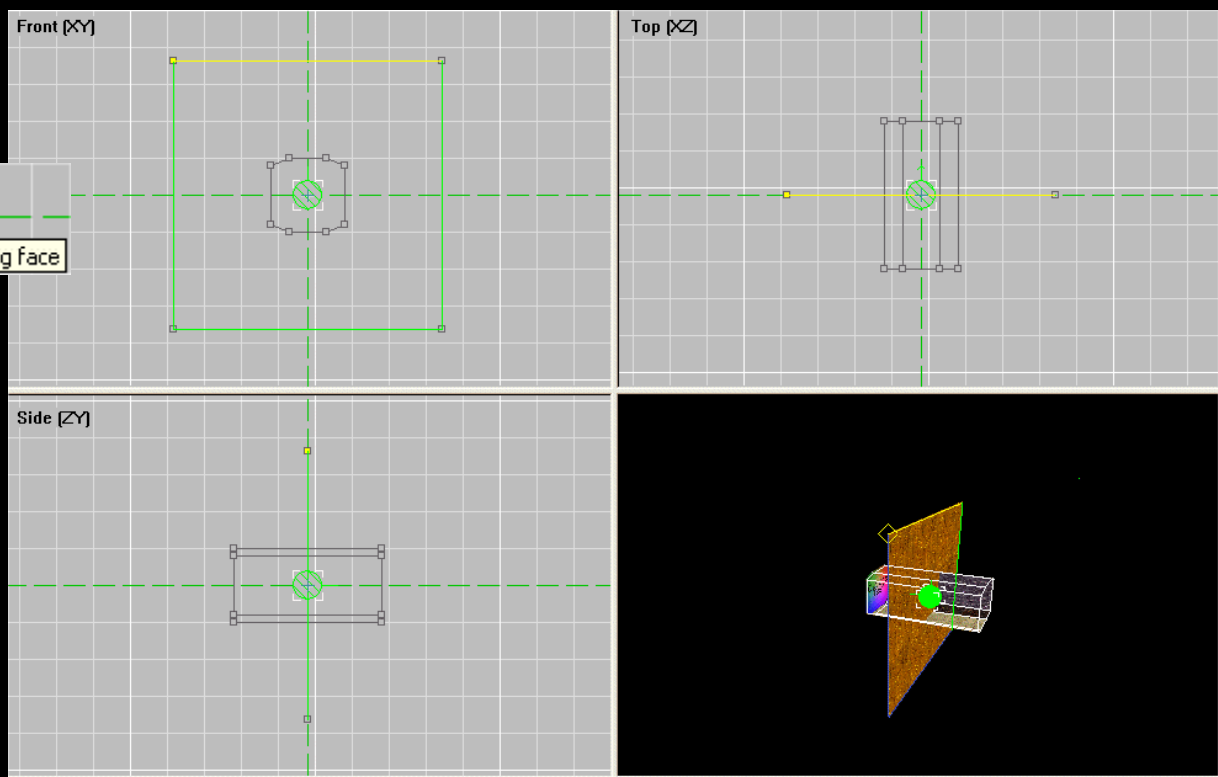
Noch ein Hinweis:

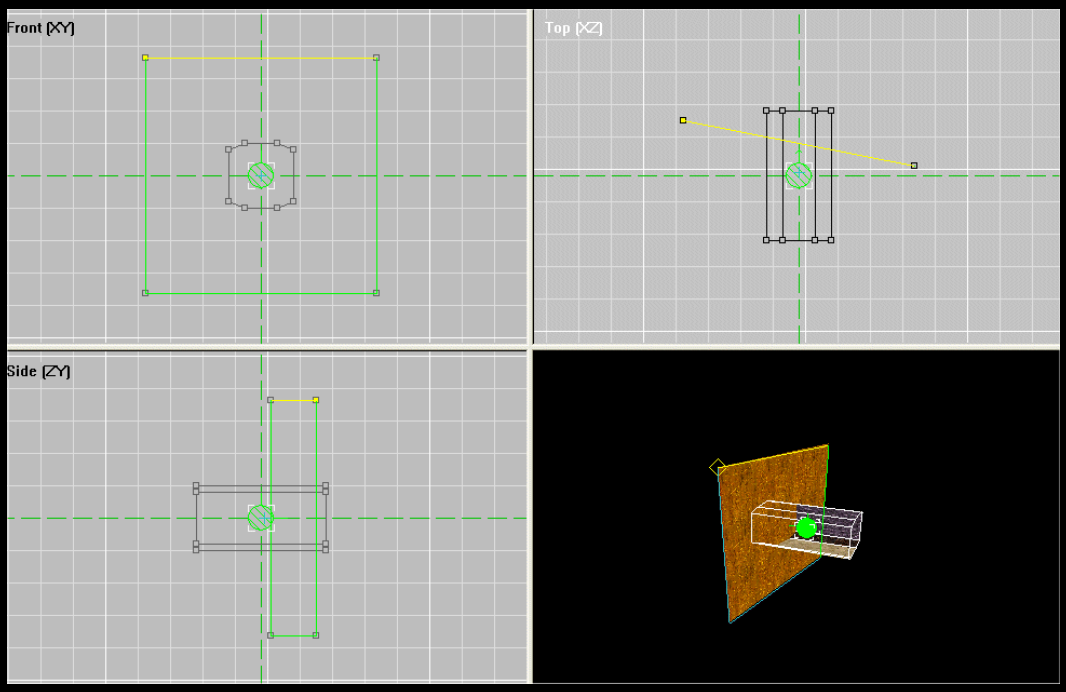
Diese Funktion ist noch nicht zu 100% ausgefeilt und getestet, also vorsichtig an die Sache herangehen.
Bei komplexen Räumen, auch solche mit Innenausbauten sollte man kein Face erstellen lassen, die Ergebnisse sind nicht besonders. Es erfordert viel Nacharbeit, probiert es mal selber aus. Portale sind auch mit Vorsicht zu behandeln.

Ablauf:

In der Room Bar betätigen wir diese Taste:

Damit wird automatisch ein Schneide-Face (mit der Current Textur) passend zum Raum erzeugt.

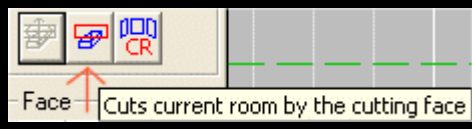


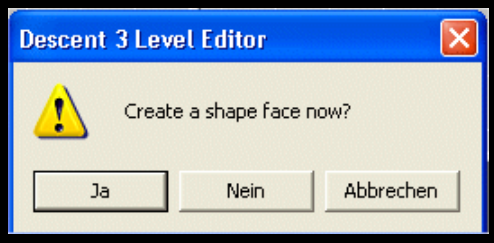


Positionieren des Schneide-Faces:
Dazu benutzen wir bei aktiver Grid die entsprechenden Tastenkombinationen, die **Numpad** Tasten.
(z.B. **2**, **4**, **6**, **8**, **Strg+Alt+1** oder **3**)

Den Raum schneiden:

In der Room Bar betätigen wir diese Taste:

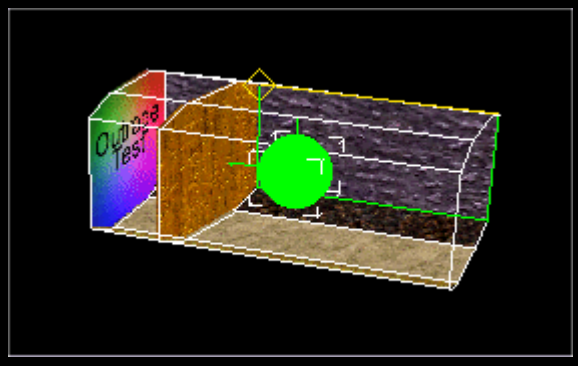


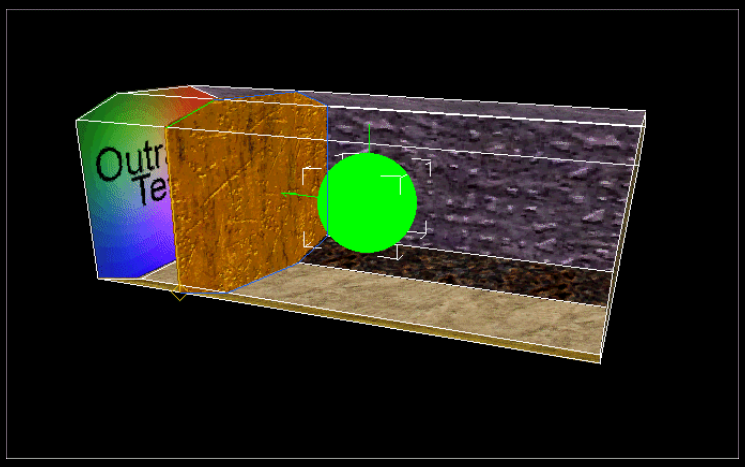


Es erfolgt eine Abfrage ob wir an dem Schnitt ein neues Face einfügen wollen. Wenn nicht erwünscht, dann werden den Schnittpunkten nur die Vertices eingefügt.

Bei komplexen Räumen ist ein Einfügen des Faces nicht sinnvoll, da dann sehr viel Nacharbeit erforderlich wird.

Wir bestätigen mit Ja und erhalten (rechts):

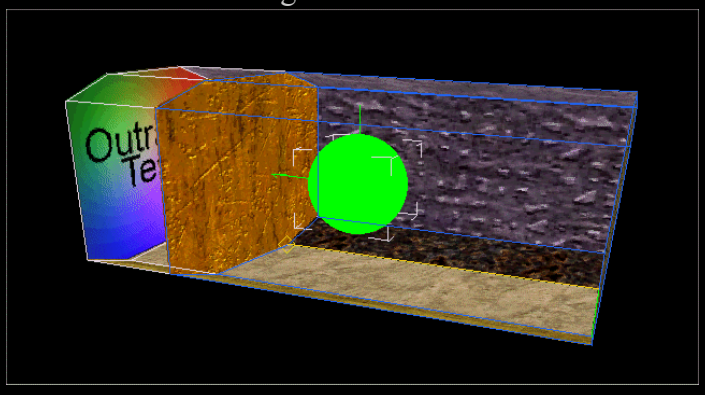




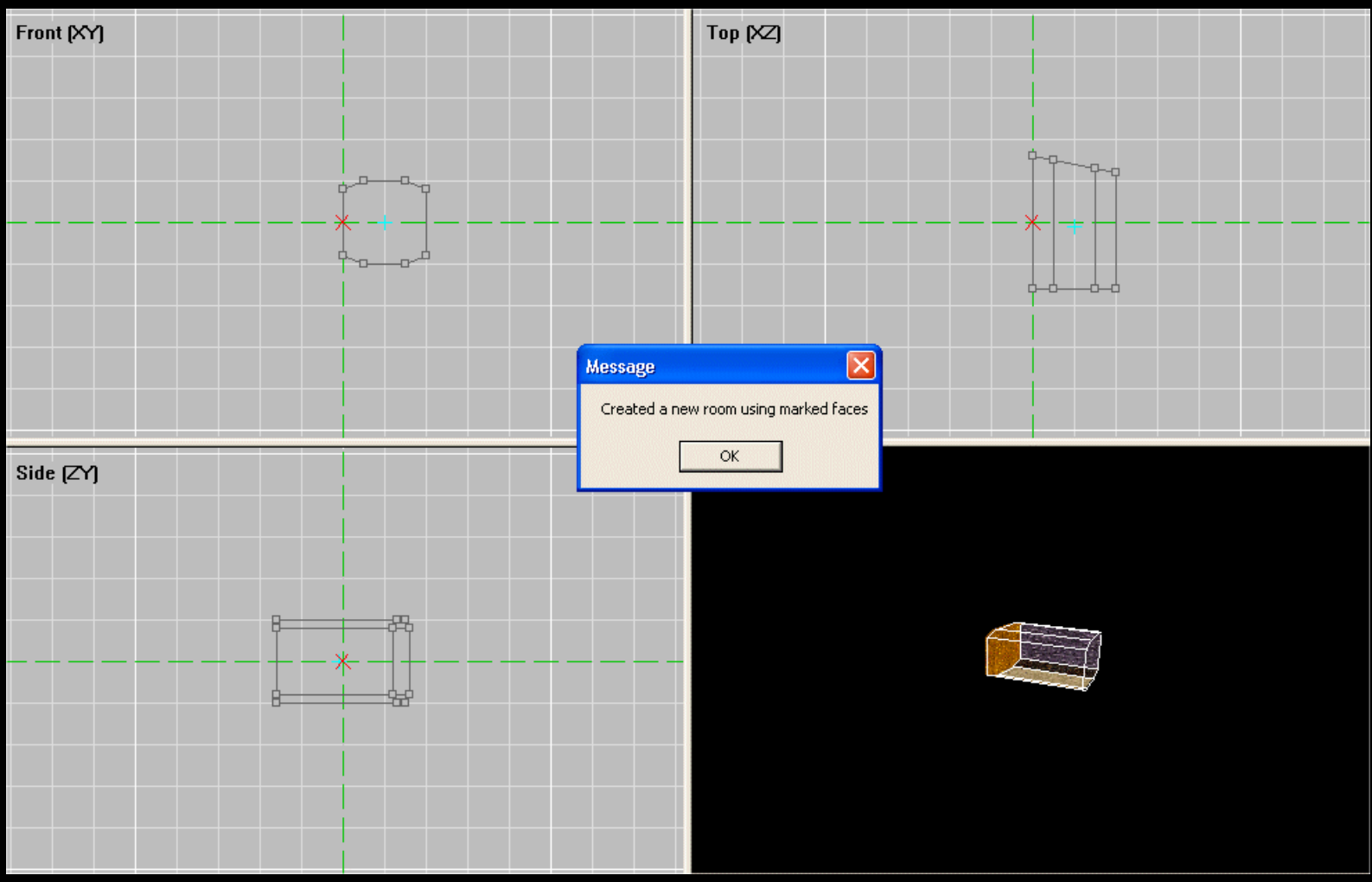
Jetzt könnten wir alle benötigten Faces einzeln anwählen, markieren, kopieren und als neuen Raum wieder einfügen.
Doch das geht auch etwas anders. Dafür markieren wir zuerst das neue Face.

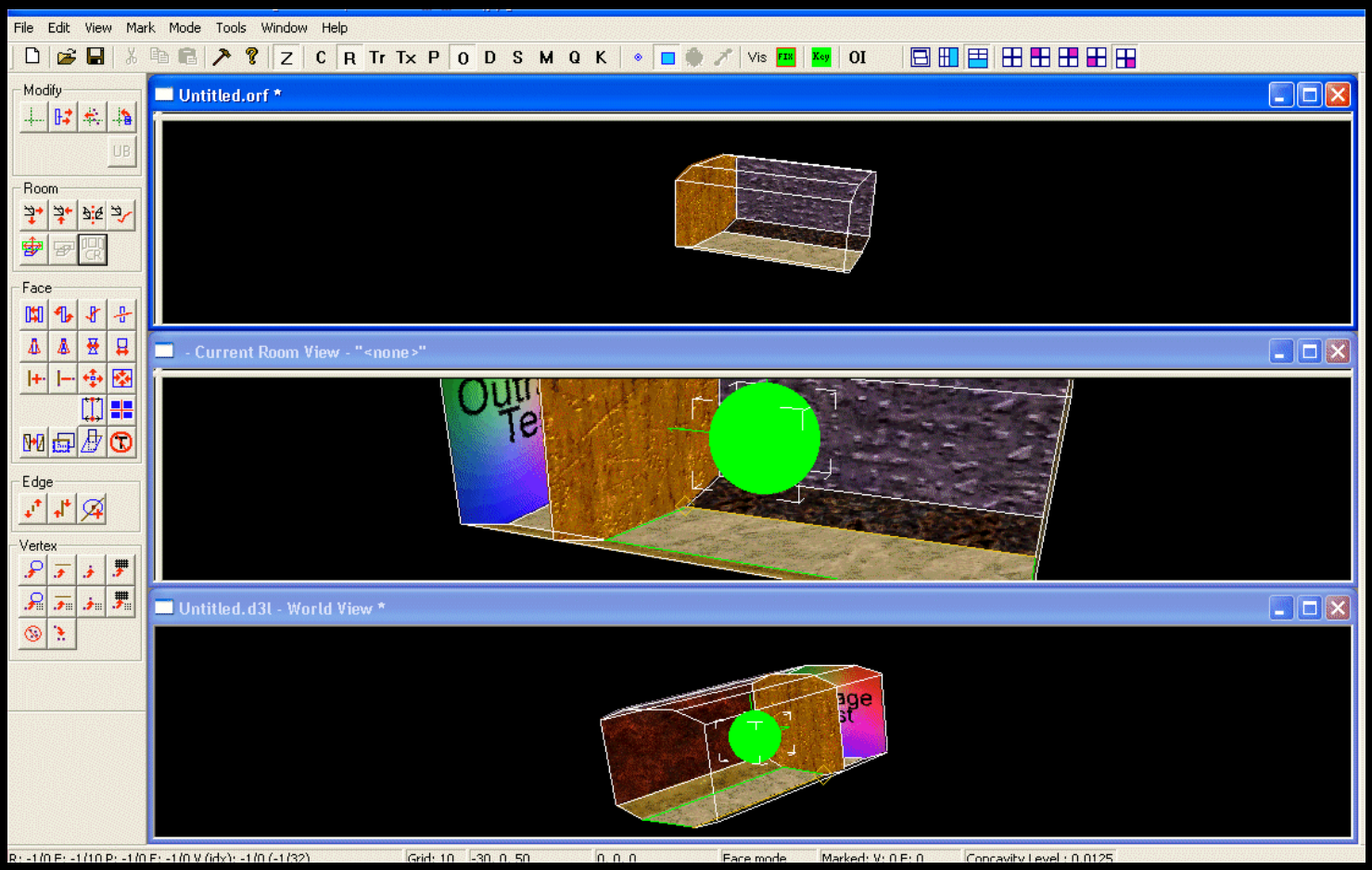
Dann benutzen wir die ‚Split marked faces vertices'-Funktion:

Danach müssen wir ein Current Face innerhalb des gewünschten Bereiches anwählen und Numpad **0** drücken. Dadurch sind auf Schlag alle benötigten Faces markiert. (Auf diese Weise kann man auch sehr gut Teile aus bestehenden Strukturen heraus lösen und z.B. verschieben.)



Nun den CR Button betätigen und wir erhalten unseren neuen Raum sofort in einem eigenen Fenster:



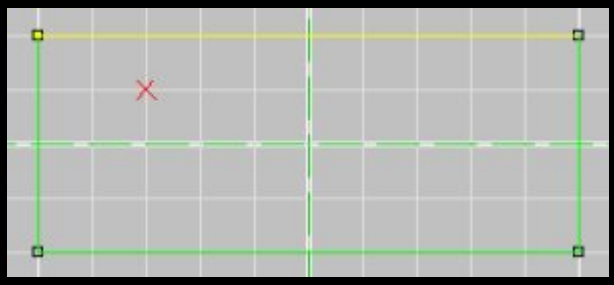


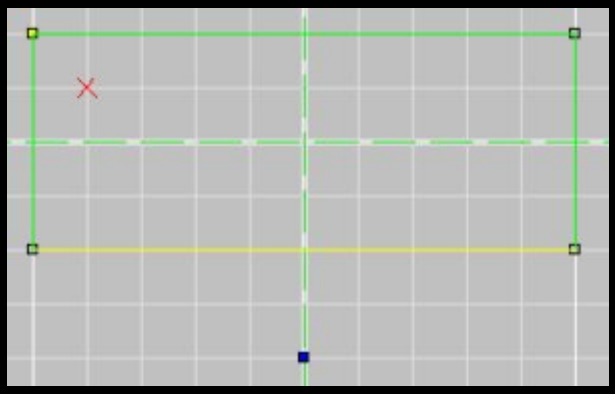
Nach der Trennung kann man den kopierten Teil im Original löschen, oder verschieben.
Nicht vergessen, man muss die Room Shell unbedingt wieder schließen und sollte doppelte Vertices löschen.

031 - Vertexoperationen an Faces

Hydra

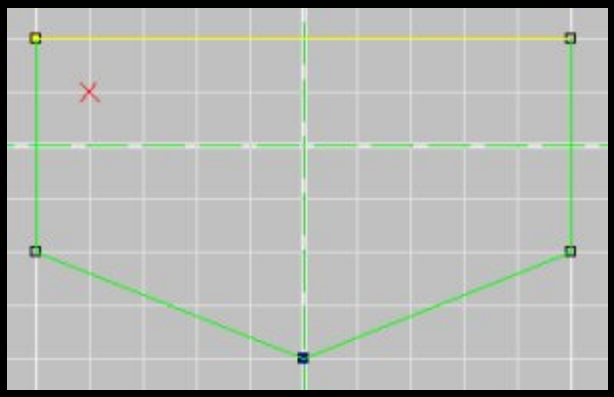
In manchen Fällen in D3Edit wirst Du die Anzahl an Vertices eines Faces ändern müssen. Das kann sein, weil Du einen T-Joint hast oder einfach nur weil Du eine Extra-Kante an diesem Face haben willst, das ist immer eine praktische Funktion wenn Dir nicht danach ist das Face neu zu machen. Zur Rechten ist ein Bild mit einem Face, Ich möchte einen Extravertex zu diesem Face hinzufügen.

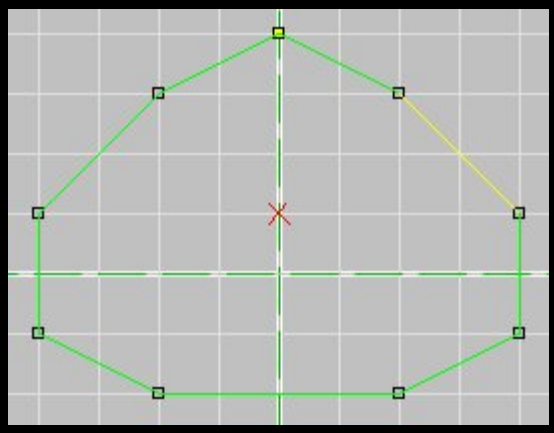




Gehe jetzt in den Vert Mode (Ctrl-R) und füge ein Vertex ein, wenn nötig bewege es **dorthin wo du es am Face haben willst**. Dann wähle das Face aus wo Du den Vertex hinhaben willst und wähle die Kante (die gelbe Linie im Umriß des gewählten Faces, mit E zwischen ihnen wechseln) des Faces, die Du splitten willst. Im Bild links habe ich das Face ausgewählt und die Kante, welche ich zu splitten plane auch. Alles was ich nun tun muß, ist den Button ‚Add Vert' drücken:

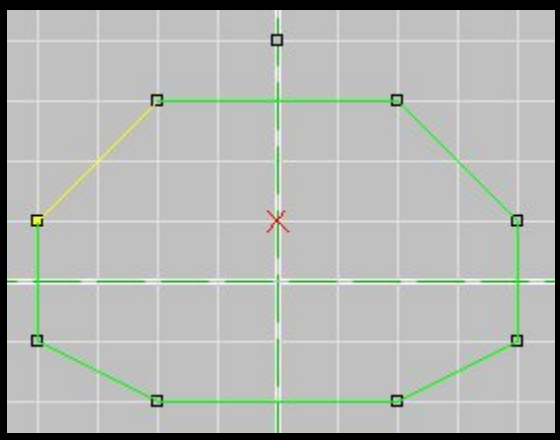
Und Du hast fertig! War das einfach oder was? Vergiß nur nicht die Textur danach wieder auszurichten, oder sie wird spaßig gedehnt sein. Diese Funktion ist sehr praktisch, wenn man versucht diese verteufelten T-Joints aus dem Level zu entfernen.





Ein Vert aus einem Face zu löschen ist beinahe exakt dasselbe. Alles was Du tun mußt ist das Face auszuwählen, dessen Vert weg soll und den Vertex, den Du löschen willst (Gehe durch die Verts mit der Taste V) und drücke den Button ‚Remove Vert':

Es poppt ein Bestätigungsdialog auf, der sicherstellt ob Du das wirklich willst, wenn Du der Meinung bist daß alles paßt klicke ‚Yes' und der Vert wird aus dem Face genommen. Nicht vergessen die Textur wieder auszurichten. Der Vertex selbst bleibt immer unberührt für den Fall daß es von anderen Faces verwendet wird, wenn Du ihn nicht mehr willst, lösche ihn einfach.

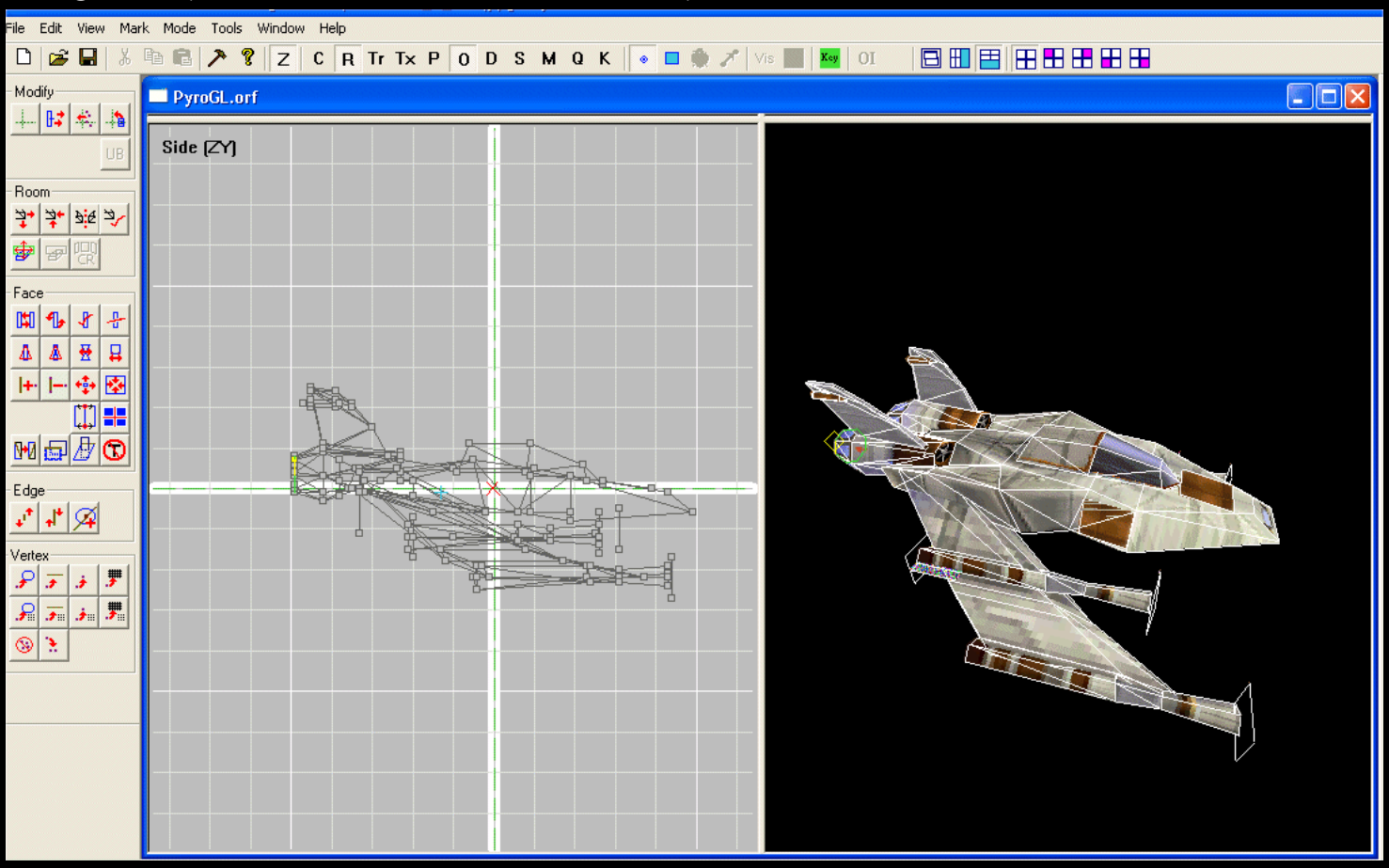


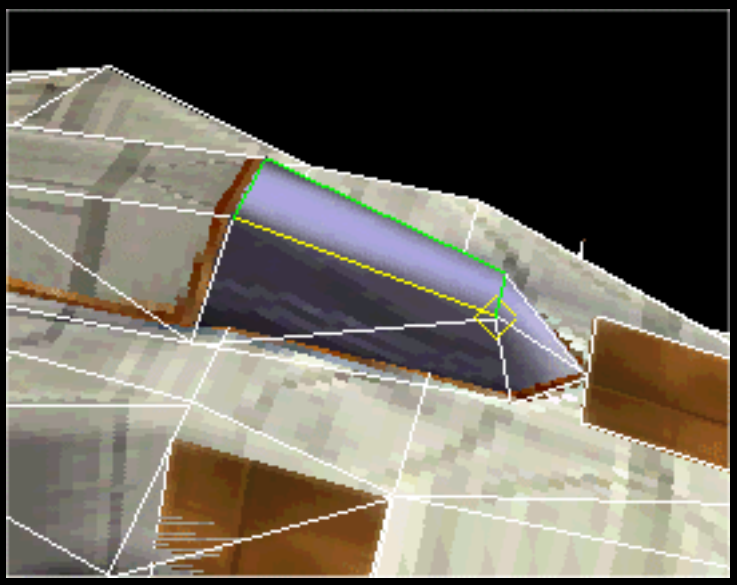
032 - Segmente anwählen

(V1.0)

Atan

Wir haben in unseren Level einen kleinen Raumhafen eingebaut und wollen dort ein paar Pyros parken. Die Idee: Ein Pyro soll mit offenem Cockpit hingestellt werden, für Wartungsarbeiten etc. Mit dem OOF-Editor haben wir bereits den Pyro von OOF in ORF umgewandelt und in D3Edit geladen. (Falls der OOF-Editor nicht läuft, geht es auch mit OOF2ORF).

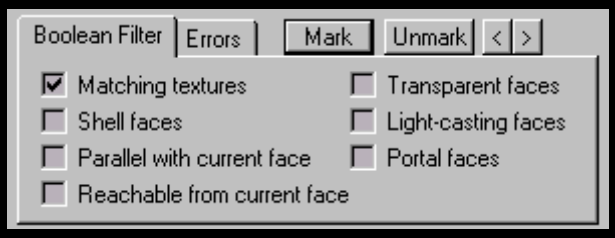


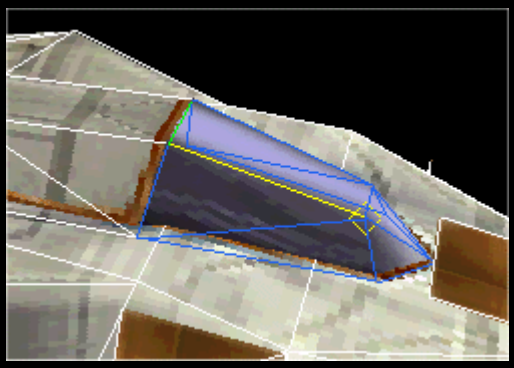


Wir rufen wir das **K**-Tool auf und wählen:
Boolean Filter -> Matching Textures -> Mark:

Mit **Tab** bzw. **Strg-F** oder über die Tool Bar schalten wir auf Face Modus um:

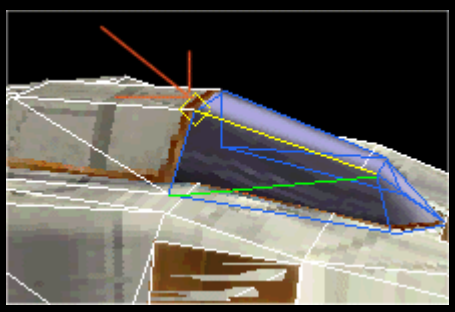
Wir wählen nun ein Cockpit Face an,



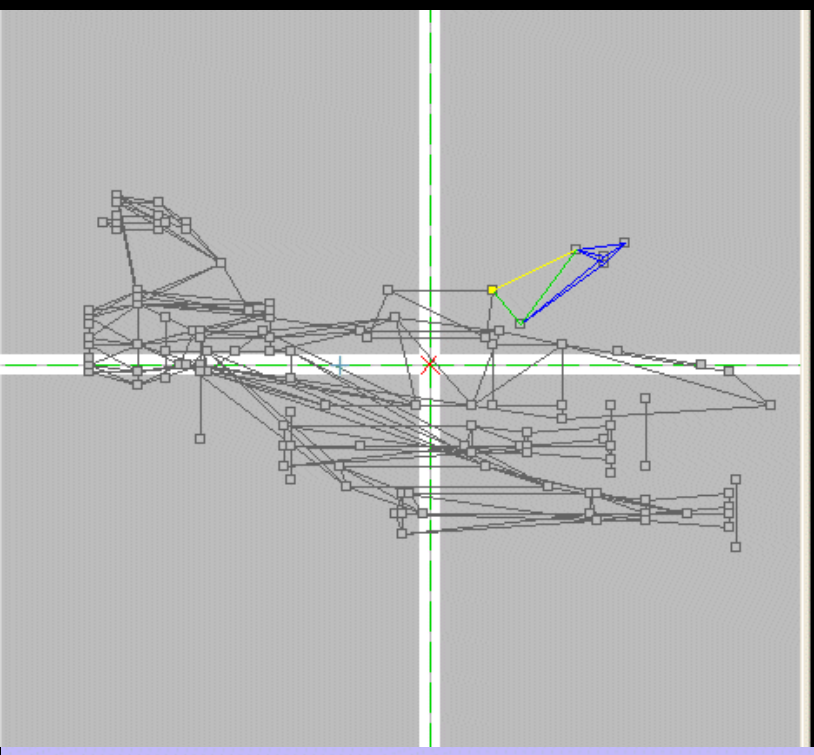


Jetzt bringen wir diese Funktion zum Einsatz:

Die Darstellung verändert sich nicht, aber wir haben alle Vertices der markierten Faces aus der Struktur gelöst. Die Faces sind jetzt von den benachbarten Faces getrennt worden.Nun müssen wir noch das Vertex anwählen um das wir die markierten Faces herum drehen wollen (rechts):



Strg-R betätigen um den Vertex Modus anzuwählen
Mit **V** zum richtigen Vertex gehen
Strg-F betätigen um den Face Modus anzuwählen
Mit der aktivierten Side View und den Tasten **Strg+1** oder **Strg+3** rotieren wir die Faces in die gewünschte Position:



Vielleicht noch ein wenig Innenausbau machen, Leiter ranstellen oder einfach die Löcher schließen.
Die überflüssigen Faces Glow, GP etc. wenn nötig noch doppelte Vertices löschen und unser geparkter Pyro ist fertig.

Sicher können die Faces auch schnell einzeln angewählt und dann markiert werden, das Herauslösen einzelner Bereiche zur Bearbeitung wird aber sicher aufwendiger sein als der oben beschriebene Weg.

033 - Benden – Grundlagen

(LL)Dark

Dieser Text ist zwar noch immer gültig, bezieht sich aber auf v39 und älter.

Hinweis:

- Am besten immer nur im "Draufsicht-Fenster" benden.(also da wo die XZ-Achsen zu sehen sind) ! In den anderen Fenstern gibt es immer Probleme.
- Immer nach links oder rechts biegen - nicht nach oben oder unten!

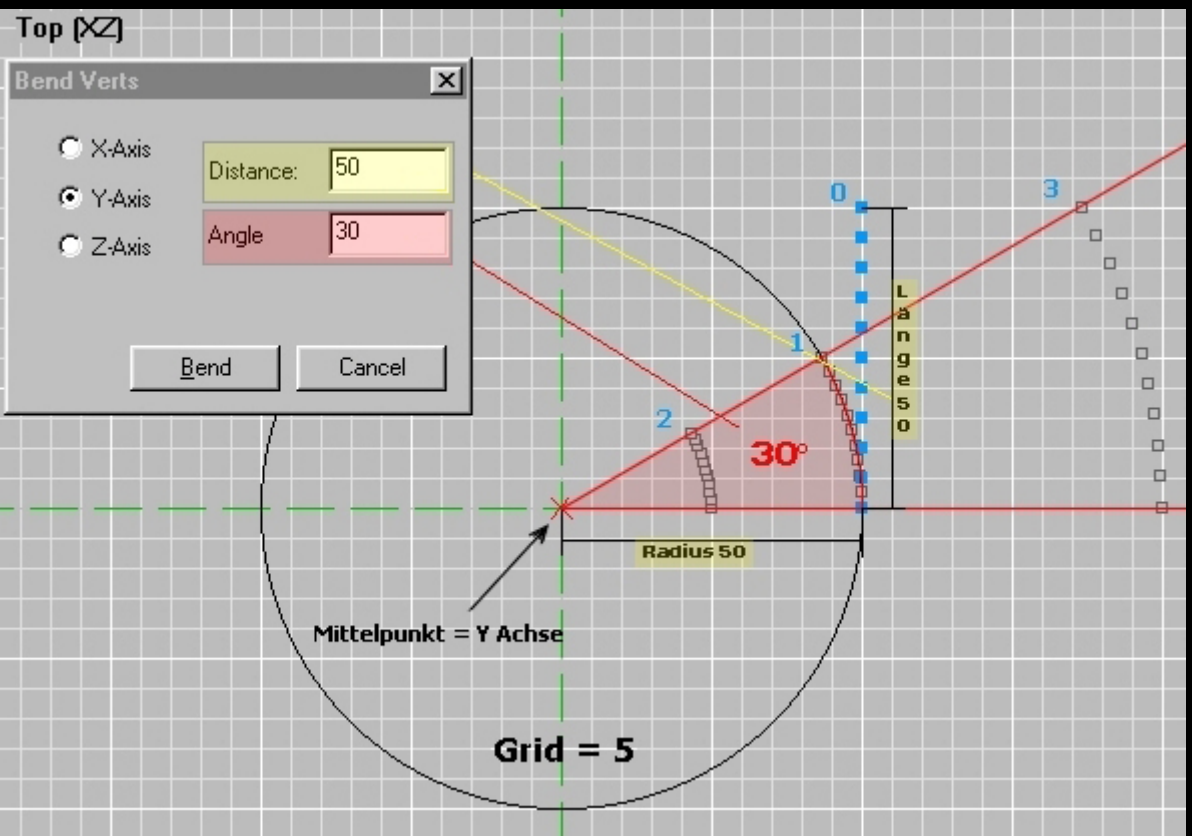
Statt eines Raumes habe ich hier der Einfachheit halber nur eine Linie aus Verts benutzt. (die kleinen **blauen Quadrate**)

Um ein akzeptables und halbwegs voraussagbares Ergebnis zu erzielen, ist es wichtig die Werte für **Distance** und **Angle** richtig anzugeben.

Die **Distance** ist der **Radius** eines gedachten Kreises um die Bendachse (hier **Y**). Sie sollte identisch sein mit der **Länge** des zu bendenden Raumes. Im Bild sind die zueinander gehörenden Werte **gelb** unterlegt.
Angle entspricht dem Kreiswinkel in Grad um den der zu bendende Raum um den **Mittelpunkt** gebogen werden soll. Im Bild sind die zueinander gehörenden Werte **rot** unterlegt.

Das Problem, daß die meisten Leute haben ist daß nach dem Benden die Länge des gebendeten Raumes nicht mit den vorher angenommenen Maßen übereinstimmt. Die Endlänge hängt nämlich von der Entfernung (**Radius**) der Verts zum **Mittelpunkt** ab.

Dies kann man im Bild deutlich erkennen. **0** entspricht den ungebendeten Verts. Alle Verts (**1**, **2** und **3**) wurden mit identischen Werten gebendet (**50**/**30**).
Allerdings jeweils mit unterschiedlichem **Radius** vom **Mittelpunkt** (die Verts von **0** wurden dazu jeweils an die entsprechenden Positionen verschoben). Demzufolge wird die **Länge** mit zunehmender Entfernung auch immer größer obwohl der Winkel (**Angle**) immer **30°** bleibt. Rückt man näher an den **Mittelpunkt** heran, werden die Verts entsprechend gestaucht. Die Krümmung des gebendeten Raumes bleibt jedoch in allen Fällen gleich.(**30°**)



Problematisch ist allerdings die Vorhersage der **Endlänge**. Gemeint ist damit die Strecke die im Bild mit **Länge** bezeichnet ist. (im Grunde genommen also die Höhe des Raum-Endes über der z-Achse - im Bild mit "Länge" bezeichnet)

Gibt man bei **Angle** 90° ein so ist diese **Endlänge** = der Entfernung zum **Mittelpunkt**
Bei anderen Winkeln klappt das aber nicht **!!!**

Noch problematischer wird es wenn man bei **Distance** einen Wert angibt der kleiner oder größer als die **Länge** des zu bendenden Raumes ist. In dem Fall wird das Ergebnis **NICHT!!!** dem erwarteten und eingestellten Winkel (**Angle**) entsprechen.

Könnt es gerne probieren.

Um also vernünftige Ergebnisse zu bekommen, muss man vorher wissen wie groß
die endgültige Krümmung des Endraumes sein soll (= **Angle**)
und man muss wissen wie lang der (ungebogene) Ausgangsraum ist (= **Distance**).

Um dem Endraum nun die gewünschte **Länge** zu geben, (weil man z.B. den Anschluß an einen bereits vorhandenen Nachbarraum herstellen will) muss man den Ausgangsraum "lediglich" in die richtige Entfernung zur Bendachse (**Mittelpunkt**) bringen.

Diese Entfernung kann man nach folgender Formel berechnen

c=a/sin(alpha)

... weil Entfernung zum Mittelpunkt = c = Hypotenuse des rechtwinkligen Dreiecks, welche gleichzeitig den Radius um Punkt A (hier der Ursprung der Drehung) durch Punkt B darstellt.
(die kürzere Kathete ist immer a, Punkt A und Winkel Alpha liegen immer der jeweiligen Seite a gegenüber)

- **c** ist dabei die gesuchte Entfernung vom >**Mittelpunkt**<
- **a** ist die gewünschte Endlänge des Raumes (Höhe des gebendeten Raum-Endes über der z-Achse – im Bild mit "Länge" bezeichnet)
- **alpha** ist der Winkel (im Bild der rote Kreisausschnitt)

im Bild für Beispiel **1** also: Entfernung zum Mittelpunkt = 25/sin(30°) = 50

Die gewünschte Endlänge war 25. Die Entfernung zum **Mittelpunkt** muss also 50 betragen. Schaut euch das Bild an - die Werte stimmen.
Vielen Dank an Floyd der diese Formel für mich herausgesucht hat.

Um für eine bestimmte Entfernung vom **Mittelpunkt** die Endlänge des Raumes zu berechnen, braucht man die Formel nur ein wenig umzustellen.

In diesem Fall würde sie lauten: c * sin(alpha) = a

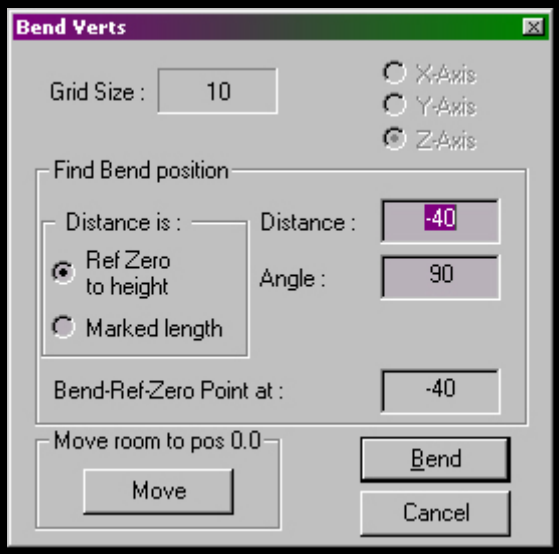
Wem das alles zu viel Rechnerei und Formelei war, der achte eben halt wenigstens darauf Bei Angle und Distance die RICHTIGEN Werte einzugeben.
Dann klappt's auch mit dem Biegen ... ☺

Viel Spaß beim Benden wünscht euch (LL)Dark

034 - New Style Bend, v39

Ragil Ral

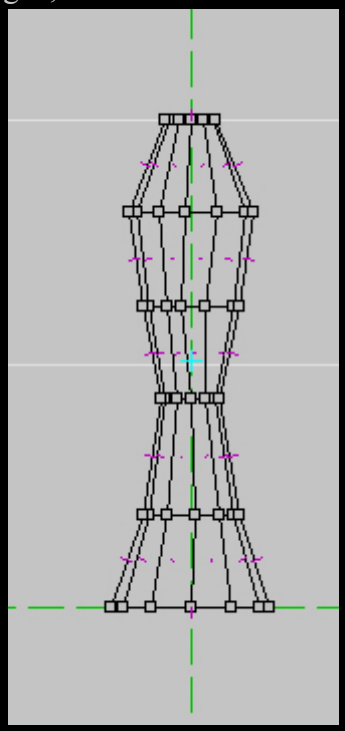
Für Bend mit der v40 lest Euch bitte {nr} - {bendv40} durch, da die Bend-Funktion komplett überarbeitet wurde.
Seit der v38 gibt es auch die Möglichkeit, mit dem 'New Style'-Dialog zu benden. Um den New Style anzuschalten geht nach Settings->Bend->dort den Haken bei New Style setzen. Der neue Bend-Dialog sieht so aus:



Aber was macht das Ding jetzt eigentlich anders?

Es nimmt einem die Berechnungen ab, die Dark vorhin ausgeführt hat, zumindest für bestimmte Situationen.

Zum Einen wird die Länge ermittelt, welche die markierten Verts haben, zum Anderen wird auf die 'Breite' (hängt ja immer von der verwendeten 2d-Ansicht ab) gemittelt eine (gedachte) Mittellinie eingezogen, welche für die Länge hergenommen wird.

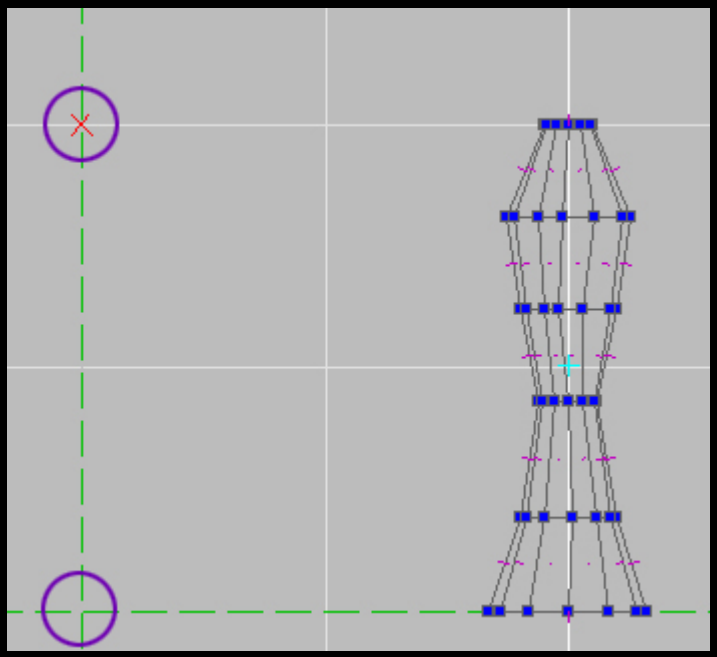


Gegeben sei das Konstrukt rechts:

Es ist genau 200 Einheiten hoch und hat einen Durchmesser zwischen ca. 25 und 66 Einheiten, ist aber zentrisch.

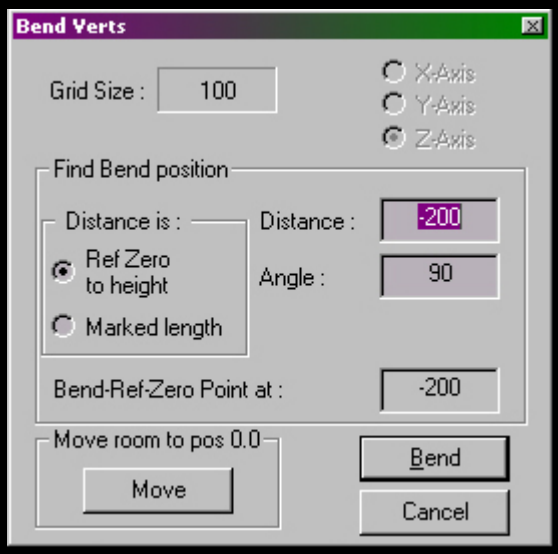
Die Achse geht genau durch die Mitte (das Bezugszentrum steht noch auf 0,0,0), ich hab ein paar Verts verschoben sodaß man diese sehen kann. Bend wird sich auf diese Achse beziehen, wenn es um 'Länge' geht.

Um Bend zu verwenden, muß eine der 2d-Ansichten aktiviert sein; das Beispiel spielt sich in der Front-View ab. Desweiteren müssen Vertices markiert sein (nona).



Gebogen wird immer gegen den Uhrzeigersinn. Will man in die andere Richtung benden, wechselt man bei Distance das Vorzeichen.

Ruft man nun den Bend-Dialog auf, werden der Bezugspunkt als auch der Cursor auf jene Punkte/Werte versetzt, die der Bend-Dialog zeigt (violett eingekreist):
Die Figur ist 200 hoch, daher wird das Bezugszentrum um 200 verschoben und der Cursor wird auf das Ziel gesetzt, wo die Biegung landen wird. Verändert man Distance, wandert der Cursor zum entsprechenden Zielpunkt.



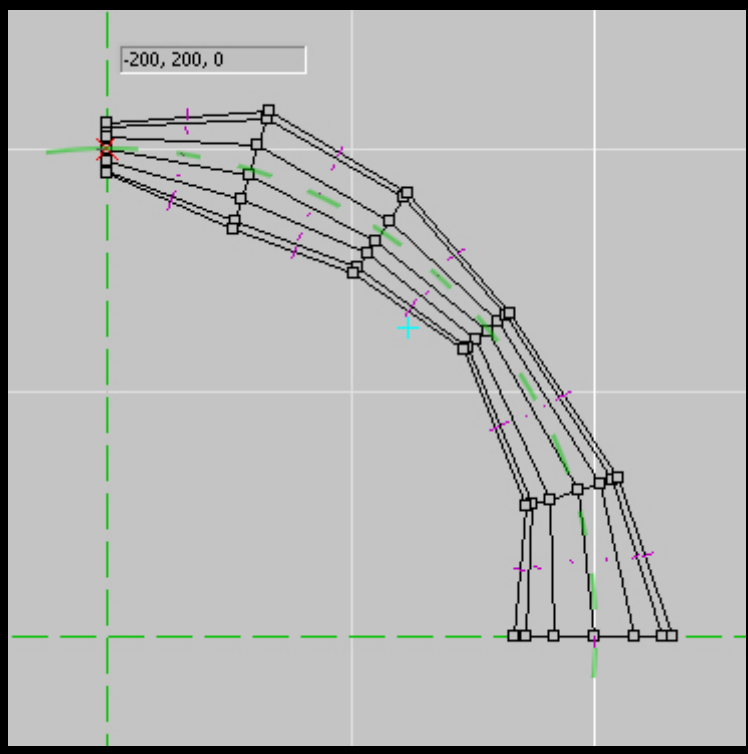
Der Bend-Dialog öffnet sich standarmäßig mit der Einstellung, daß um 90° nach links gebogen wird und die Endlänge... dazu kommen wir noch.
Der Dialog Links gehört zu der Figur von vorhin:

Wie Ihr seht, können die Achsen nun nicht mehr angewählt werden, sie werden durch die Ansicht bestimmt, welche aktiv war als Ihr Bend aufgerufen habt.

Distance und Angle: Haben weiterhin jene Funktion die sie im klassischen Bend auch hatten. Nur,
Distance is: Hier bestimmt Ihr die Endlänge Eures Ergebnisses, wobei:

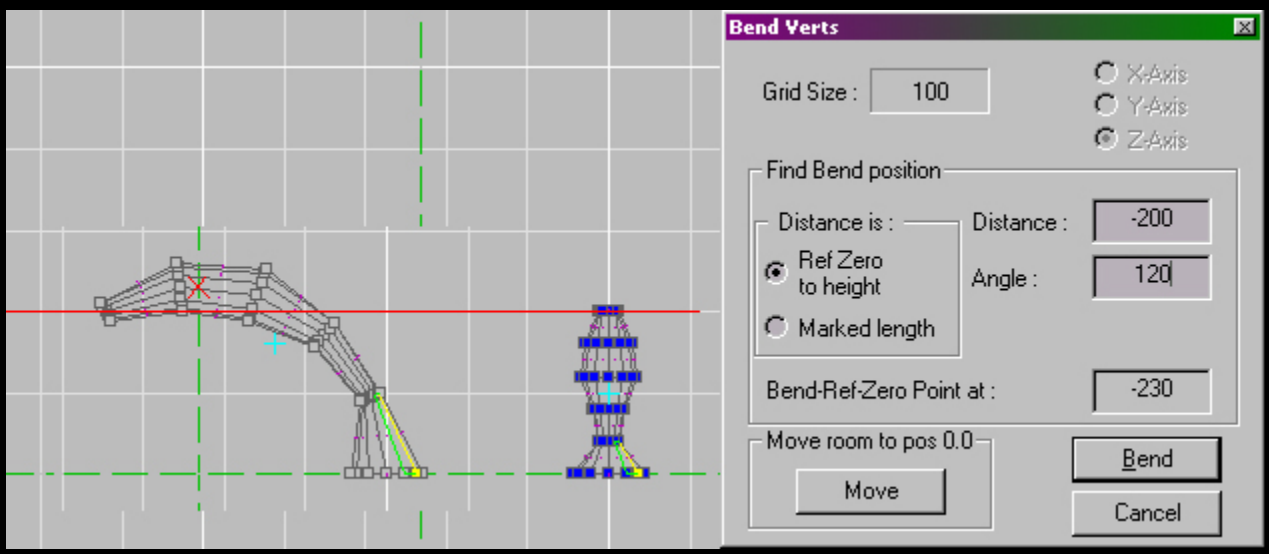
Ref Zero to Height bewirkt, daß der Bezugspunkt bzw. die Bezugsachse fürs Benden derart gelegt werden, daß das Endprodukt diejene Länge (s.o.) bekommt, welche die Markierung aufweist. Nimmt man als Winkel 180° und mehr, wird die Länge kurzerhand in die andere Richtung ausgerichtet (nur in der Front-View).

Das Ergebnis sieht so aus:
Ich habe die alte Achse als grün strichlierten Teilkreis eingezeichnet und die Position des Cursors reingesetzt.
Wenn man als Winkel 62.185° angibt, wird das Stück genau diese Krümmung haben und die (gedachte) Mittelachse die Länge der ursprünglichen Markierung.

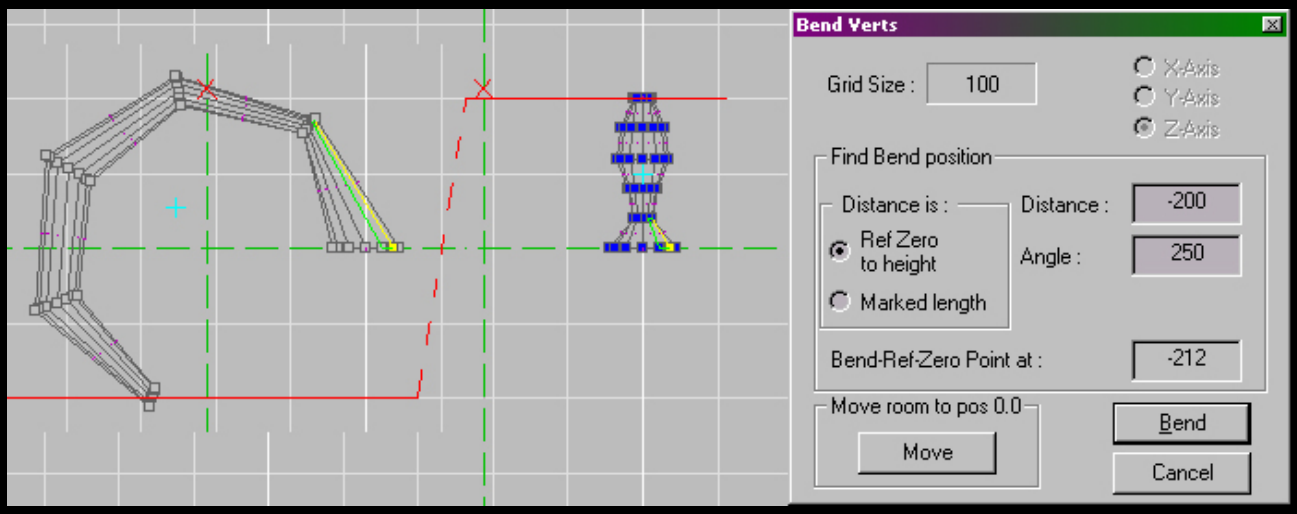


Soll aber die Außen- oder Innenseite der Gebendeten Verts eine bestimmte Länge erreichen, muß man noch die andere Abmessung (Breite bzw. Durchmesser) beachten und diese bei Distance einrechnen; hier müßt Ihr Euch wieder an Darks Ausführungen halten, und die Werte im Dialogfeld entsprechend anpassen.

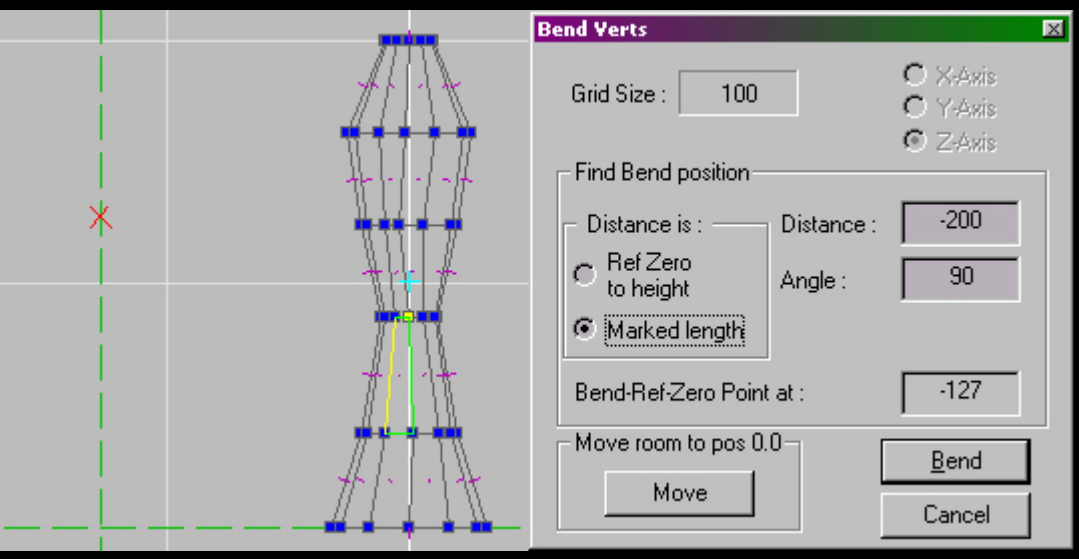
Winkel größer als 90°: Die rote Linie repräsentiert Distance.



Winkel größer als 180°: Die rote Linie repräsentiert wieder Distance*, das 'Umklappen' ist durch die strichlierte Linie angedeutet.*

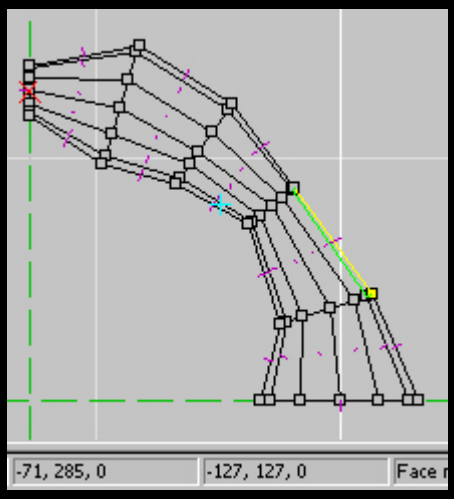


Marked length bewirkt, daß sich die Länge des gebendeten Objektes selbst nicht ändert, sondern daß es nur um den angegebenen Winkel verbogen wird (Links): Das heißt natürlich auch, je größer der Winkel, desto kleiner wird die Konstruktion zusammenschnurren.



Man beachte Bend-Ref-Zero Point at: er stimmt nun nicht mehr mit Distance überein.

Das Ergebnis:



Auch zu sehen, rechts unten im Bild die Position des Cursors.

Wenn man's nachrechnet, bekommt man als effektive Länge der (gedachten) Mittelachse 200,27 Units, das liegt daran daß New Style Bend mit ganzzahligen Werten arbeitet.

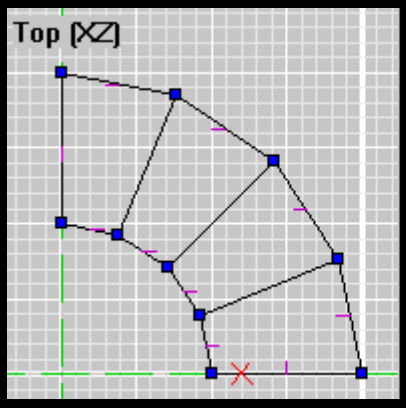
Weiters ist der 'Offset' des Objektes vom Koordinatenursprung von Relevanz, in diesem Beispiel war die Mittellinie des Objektes deckungsgleich mit der Ursprungsachse. Befindet es sich nicht genau darauf, wird der Offset zweimal auf Distance aufgeschlagen, sofern dieser positiv ist.

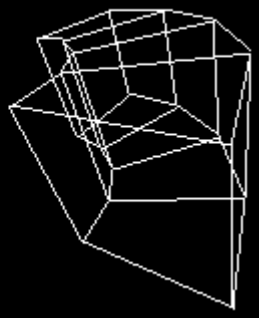
Nimmt man andere Werte für Distance, dann entspricht der erreichte Winkel nicht mehr so genau dem was man erwarten würde, hier experimentier' ich selbst noch... doch nehmt die neue Editorversion, die v40, haltet Euch an die Sektion {nrxxx} - {runderneuertes Bend} und Ihr spart Euch sehr viel Herumexperimentiererei.

035 - Die Bend-Funktion

Schplurg

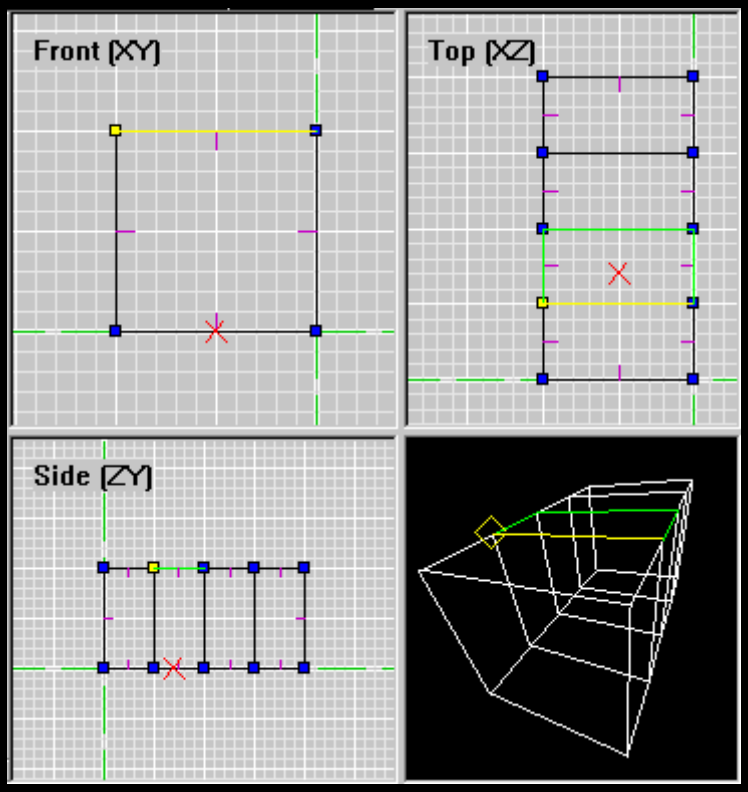
Perplex gebendet? Siehe...



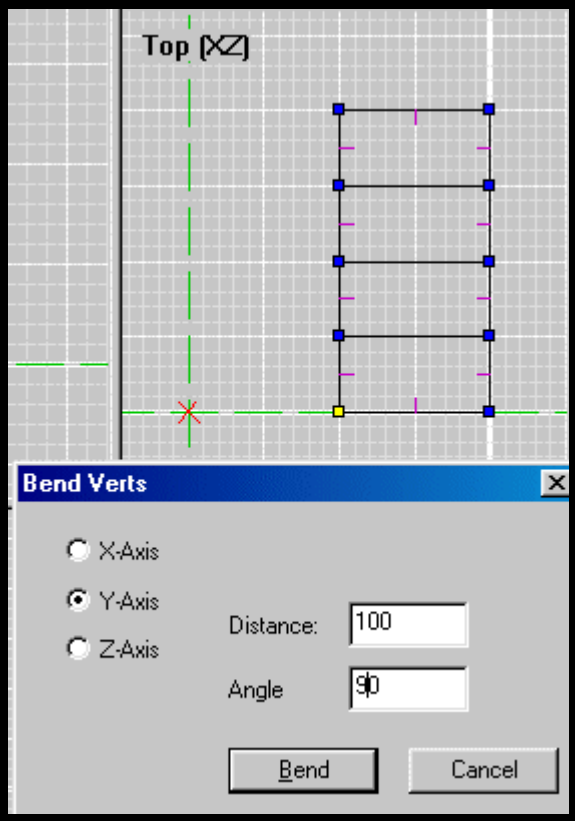


Ich empfehle diesen Schriten zu folgen und die exakten Spezifikationen, die in diesem Tutorial angegeben werden beim Bau Deines Tunnels zu verwenden. Bist Du erst mal vertraut mit der Art wie das funktioniert, wird späteres Experimentieren einige sehr interessante Resultate erbringen.

Unser fertiger Tunnel wird um 90 Grad nach links gebogen, wenn man von oben draufschaut wie im Bild Links.



Fange damit an, einen Tunnel zu bauen der 100 Einheiten lang ist und jeweils 50 Einheiten breit wie hoch. Dazu beginnst Du mit einem 50x50-Face und extrudierst den Tunnel entlang der Z-Achse in 4 Segmenten, jedes 25 Einheiten lang (rechts). Die Segmente sind nötig da sich bei diesen Verbindungen der Tunnel biegen wird.



Gehe in den **Vert-Mode** und markiere alle Verts. Wir werden unsere Aufmerksamkeit nur auf die **Top-View** (XZ) konzentrieren, da wir dort unseren Bend am besten sehen. Stelle den Referenzpunkt 50 Einheiten links und bündig mit der Vorderfläche. Verwende **Ctrl-Klick** um den Reference Frame (Der Punkt wo sich die zwei grünen Linien schneiden) zu setzen. Dies ist der Zentralpunkt, um den sich der Tunnel biegen wird.
Du solltest den Raum zu diesem Zeitpunkt speichern, es gibt zwar ein Un-Bend , aber sicher ist sicher. Öffne den Bend-Dialog über den Button in der Geometrie-Bar: Hier ist ein Hinweis auf die Begründung den Referenzpunkt zu setzen, wie wir es gemacht haben (die grünen Linien).

Die Werte im Bend Verts-Dialog können recht knifflig werden. Ich beschreibe sie mal kurz; am Ende dieser Übung bespreche ich diese Funktionen tiefergehender.

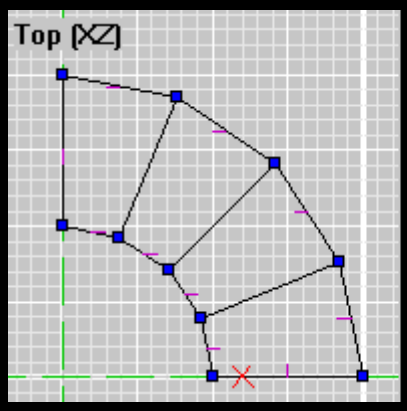
Axis ist der Achsen-Pol um den sich die Biegung windet. im vorigen Bild ist X horizontal und Z vertikal. Du betrachtest Den Tunnel daher entlang der Y-Achse genau von oben. Dies ist der Pol um den sich unser Tunnel biegen wird - die Y-Achse.

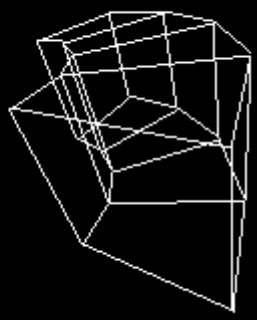
Distance sollte auf die Länge der Sektion gesetzt werden, die gebogen werden soll. Unser Tunnel ist 100 Einheiten lang... gib hier 100 ein. Es können für andere Resultate auch negative Werte eingegeben werden.

Angle ist der Betrag um den Du den Tunnel biegen willst. Für jetzt verwende die sicheren und geraden 90 Grad. :o)

Ok jetzt, achtung... hol tief Luft und... (gespeichert hast Du ja?) klicke Bend!

Du solltest jetzt soetwas sehen. Wenn es nicht funktioniert hat, entspann Dich, streck Dich, und dann mach die Übung nochmal.





Wie das funktioniert:

Es gibt einige wichtige Schlüssel beim Benden:

Der Reference Frame ist die Achse für den Bend, wie die Nabe eines Rades.

Distance ist eine große Variable. Versuche die obige Übung, aber setze den Reference Frame zur **rechten Seite** des Tunnels anstatt der Linken. Gib die selben Werte in den Bend-Dialog ein und guck was passiert... er wird nach rechts gebogen, aber auch umgestülpt!!! Nun mach dasselbe, aber setze diesesmal die Distance auf **–200**... hey, es hat hingehauen!

Angle ergibt ebenso unterschiedliche Resultate, wenn Du den Wert änderst.

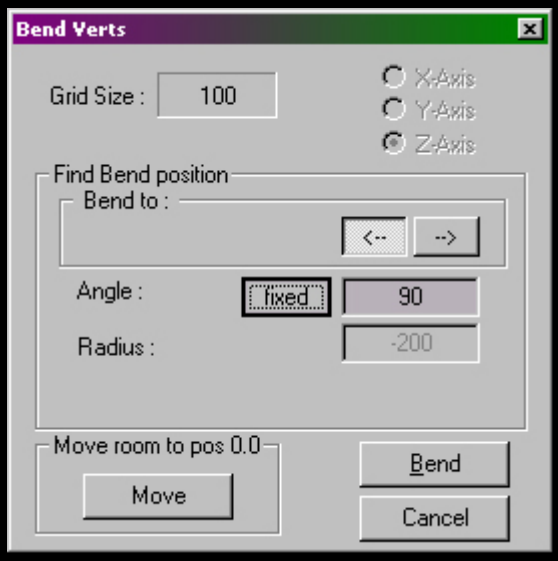
Spiele mit den Einstellungen unter Verwendung dieses einfachen Tunnels und du hast Dir einen scheinbar komplizierten Aspekt von D3Edit erobert.

Anmerkung: im folgenden Tut gibt es wichtige Informationen zur Bend-Funktion und Texturausrichtung auf gestreckten Oberflächen. Es ist wärmstens empfohlen, daß Du sie durcharbeitest!

036 - Runderneuerte Bend-Funktion in v40

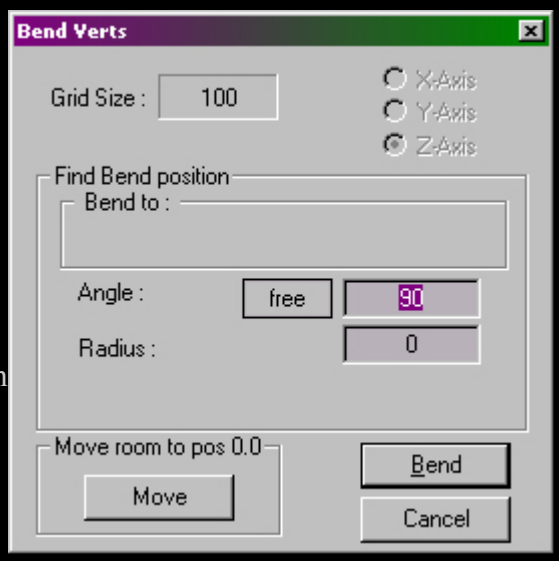
Ragil Ral

In der v40 wurde das Bend vollkommen überarbeitet. Es gibt nun nicht mehr die Möglichkeit mit 'old' oder 'new' Style zu benden, sondern nur noch ein einziges Dialogfeld. Dafür ist Benden nun einfacher und eingängiger geworden. Der Dialog hat nun zwei Gesichter:



Die Achsenwahl erfolgt wie beim new Style Bend durch die Ansicht, in der Ihr Euch befandet als die Funktion aufgerufen wurde.

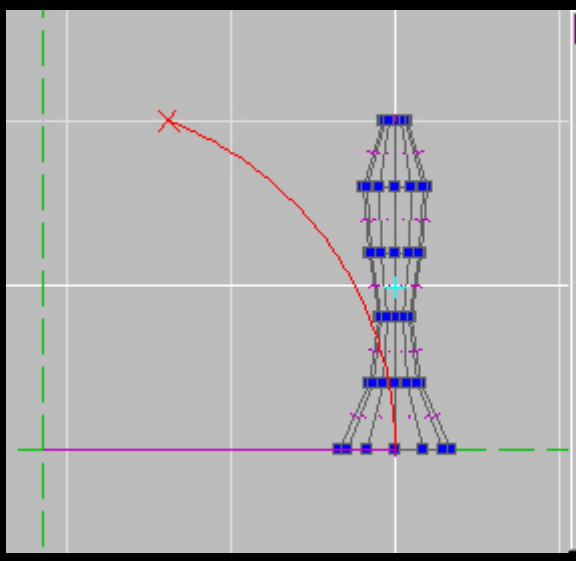
Beachtet den Button links von Angle.
Damit bestimmt Ihr den Bend-Modus.



Bend mit fixed

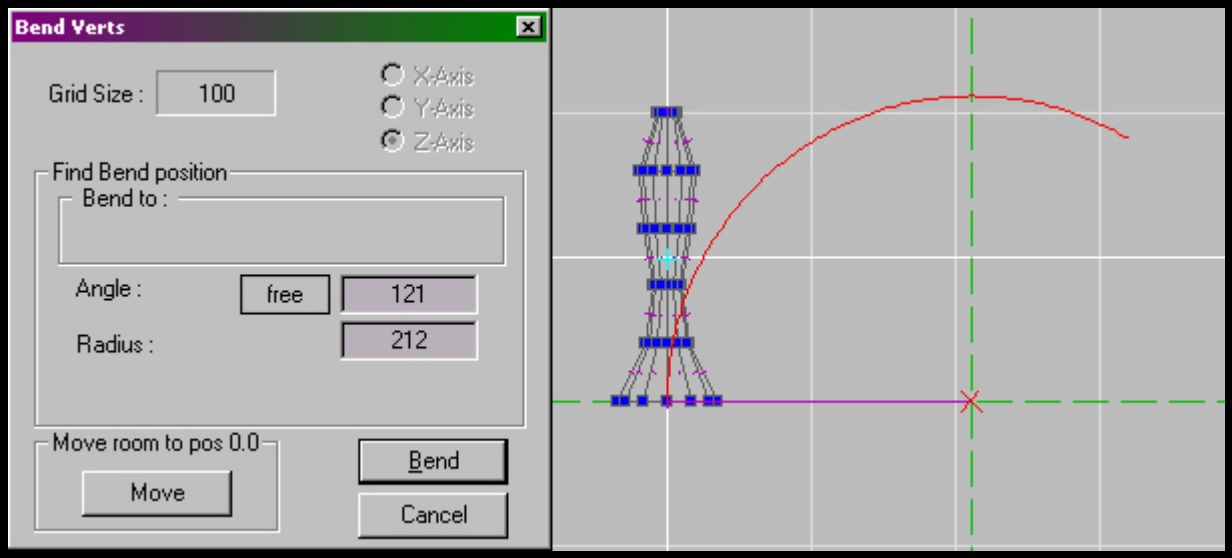
Diese Einstellung verhält sich weitgehend wie new Style Bend. Neu sind jedoch die Buttons mit den Pfeilen, wo Ihr die Biegerichtung angeben könnt. Als besonderes neues Schmankerl hat Atan aber etwas eingeführt, was Ihr sehr bald zu schätzen wissen werdet, nämlich eine Voranzeige wo der Bend landen wird!

Die rote Linie zeigt Euch das an.
mit der Einstellung **fixed** wird der Radius errechnet, Winkel können nur bis 90° angegeben werden. Die Länge des Bendats (ahem) ist immer die markierte Länge.



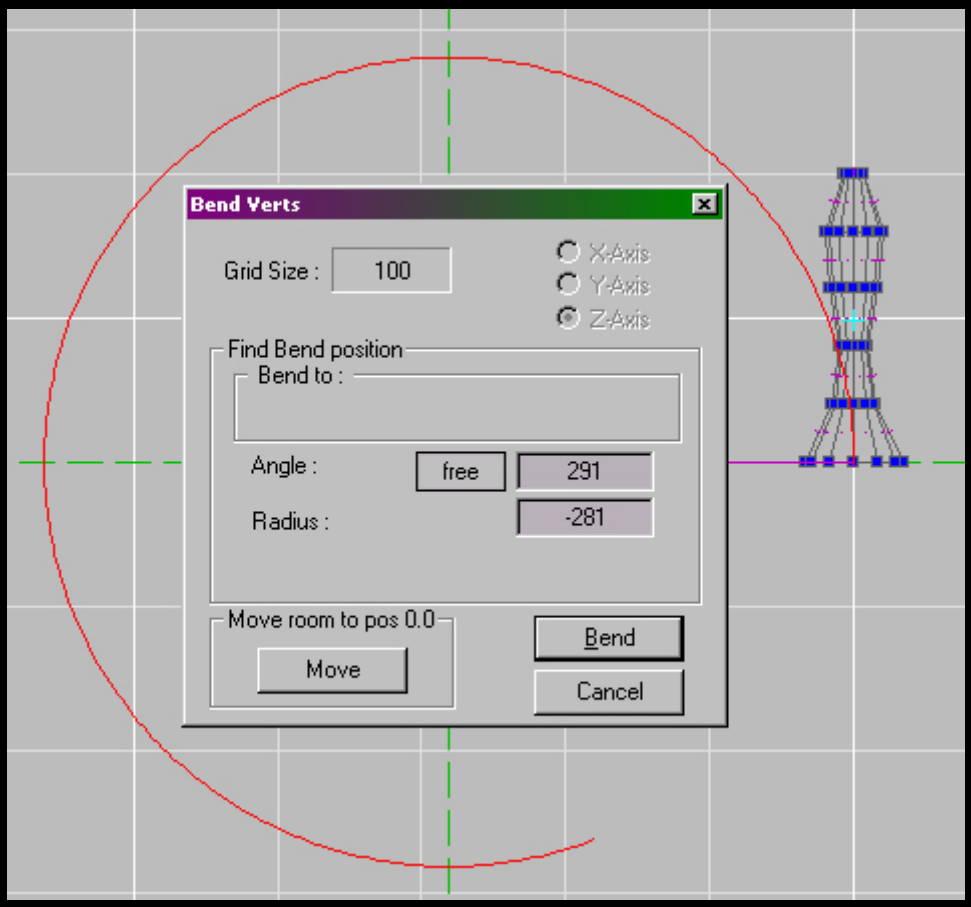
Bend mit free

Nun ist es aber vielleicht so, daß Ihr andere Winkel und/oder einen anderen Radius benötigt. In diesem Falle klickt auf den Button neben Angle, sodaß **free** drinsteht.



Nun könnt Ihr nach Herzenslust und -laune Werte setzen: Winkel zwischen 0 und 360, Radius beliebig. Beachtet aber, daß sich die Länge dann entsprechend verändert, hier haltet Ihr Euch entweder an Dark's und Papacats Ausführungen zur Mathematik hinter Bend um den gewünschten Wert zu errechnen, oder aber Ihr spielt mit den Einstellungen solange 'rum, bis der Voranzeiger dort landet wo Ihr euch das vorstellt.

Grundsätzlich wird rechtsherum gebogen, die Richtung ändert Ihr hier wieder über das Vorzeichen bei Radius.



Move room to pos 0.0

Für beide Bendvarianten, **free** und **fixed**, bewirkt dieser Button daß die markierten Verts auf die Position 0,0 verschoben werden (in der entsprechenden 2d-Ansicht, versteht sich), wobei das 'Unten' der markierten Verts genau auf der horizontalen Achse landet sowie der Winkel auf Standard (Angle 90) gesetzt wird. Und wenns unwahrscheinlicherweise doch nicht so geworden ist wie erwartet, gibt es immer noch den UnBend-Button 😁

Na, dann fröhliches Biegen!

037 - Bend: Mathematik

Papacat

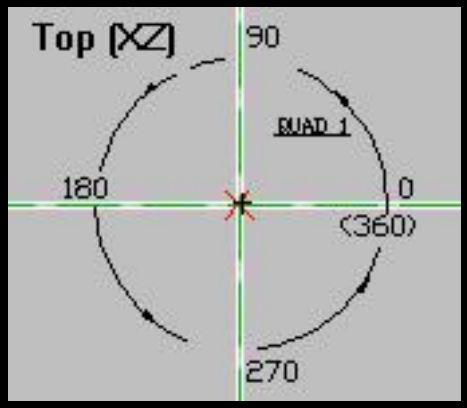
Erstens, die zwei Regeln mit denen Du in D3Edit arbeiten mußt:

1. Objekte (Räume, Säulen, Befestigungen...) Rotieren mit einem Inkrement von 11.25 Grad (360 Grad geteilt durch 32) wenn Du die 1 oder 3 am Numpad betätigst.
   (in neueren Editorversionen ist dieses Inkrement von der Gittergröße abhängig. Das hier angeführte Inkrement gilt bei Gittergröße 10)
2. Das Texture-Alignment-Tool dreht Texturen um 2.8125 Grad-Inkremente... das bedeutet daß 32 Klicks die Textur um 90 Grad drehen.
   (Auch diese Inkremente hängen von der gesetzten Gittergröße ab, können aber im Align-Dialog per Hand eingegeben werden.)

Es ist einfach zu merken: Objekte rotieren - beginne mit 90... geteilt durch 2 = 45... geteilt durch 2 = 22.5... dividiert durch 2 ist 11.25. Verwende diese Inkremente oder eine aus ihnen kombinierte Summe um Objekte auszurichten oder sie im Verhältnis im Raum zu platzieren (Siehe auch die inkrementeliste im Abschnitt Reisberg.).

Für Texturen gehe zwei Schritte weiter... 11.25 geteilt durch 2 = 5.625 dividiert durch 2 = 2.8125. verwende diese Inkremente oder eine aus ihnen kombinierte Summe wenn Du entwas biegst oder rotierst um sicherzustellen daß du die Texturen dann ausrichten kannst (speziell Böden und Decken in Tunnels).

Nun erstmal ein Auffrischungskurs in Mathematik der 7. Stufe. Danach gehen wir durch die Schritte des Biegens eines Tunnels.

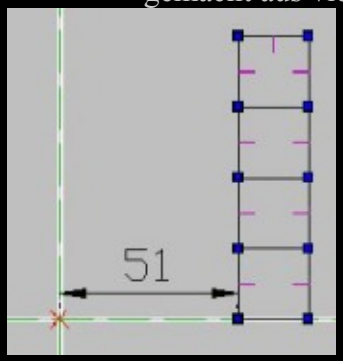


Erstmal das Offensichtliche. 360 Grad in einem Kreis. Hier ist eine Top-Ansicht aus D3Edit.
Wenn Du bendest, beginnst Du im ersten Quadranten (QUAD 1). Das Objekt wird sich gegen den Uhrzeigersinn um die Achse biegen/bewegen. Dies wird Dir helfen herauszufinden, welchen Wert Du bei Angle im Bend-Dialog setzen mußt.
Zweitens, drei Formeln um herauszufinden wie weit man den Tunnel von der Achse setzen muß:

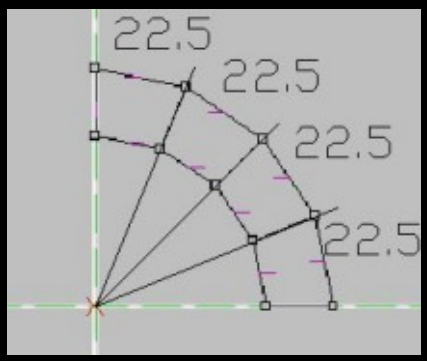
Umfang=Durchmesser x Pi
Radius=Durchmesser/2
Das Verhältnis der Tunnellänge zum Umfang ist gleich dem Verhältnis des Biegewinkels zu 360 Grad.
Tunnellänge/Umfang = Biegewinkel/360

Das dritte hilft einen gewissen Teil des Tunnels ‚fast' in der selbenGröße zu erhalten, bevor Du ihn gebogen hast. Du kannst eigentlich von jeder Distanz zur Achse Benden, mit jedem Winkel und D3Edit wird die Segmente quetschen oder strecken, sodaß sie in den Winkel passen.

Laß uns einen Tunnel machen und sehen was passiert. Wir werden einen 80-Einheiten-Tunnel, gemacht aus vier 20x20x20-Segmenten, um 90° biegen.

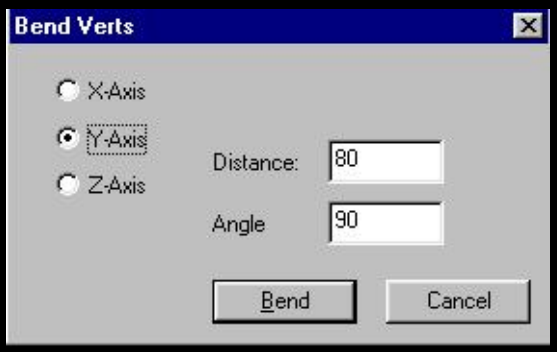


Ich will wissen ob die Segment-Decken es erlauben Texturen auszurichten, bevor ich mit dem Bau Zeit verschwende. 90/4=22.5... das ist gut für Texturen. Es ist auch gut zum Rotieren, so daß ich den Tunnel an seinen Platz drehen kann wenn nötig (rechts).



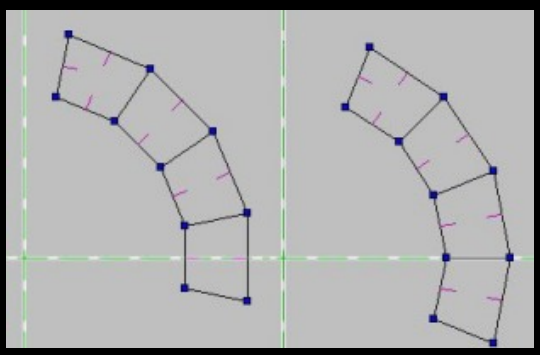
Nun... ich möchte daß die Innenwand ‚fast' die selbe Größe behält, also wie weit weg von der Achse setze ich sie?

Finde heraus welchen Teil des Umfangs der Tunnel brauchen wird. (360°/90=4)
Wenn der Tunnel 80 Einheiten lang ist, was ist der Umfang? 80 Einheiten x 4=320 Einheiten
Nun den Radius finden. 320 Einheiten / Pi = 101.9 Durchmesser, 101.9 / 2 = 50.95 Einheiten Radius.



Da die kleinste Einheit in D3 1 ist, setzen wir die innere Kante 51 Einheiten von der Achse. Öffne den Bend-Dialog und setze die Variable wie im Bild. Klicke **Bend**.

Da alle Texturen entweder gestreckt oder gestaucht wurden als wir den Tunnel gebended haben, mußt du **Align** verwenden und alle Texturen auf ‚Default UVs' setzen. Aber da wir sichergestellt haben, daß die Segmentwinkel zu treffen sind haben wir kein Problem die Decken- und Bodentexturen zu drehen, bis sie mit den Wänden passen.



Nun beobachte was passiert wenn Du den Tunnel rotierst. Markiere alle Verts und drücke dann die 3 am Numpad einmal. Der Tunnel dreht sich um 11.25° im Uhrzeigersinn und das Segment sitzt nun quer über der X-Achse. Rotiere nocheinmal, und die Naht zwischen den Segmenten liegt genau auf der X-Achse. Rotiere weiter, und schau wie es sich verhält. Dies ist großartig, wenn Du Objekte wie Halterungen oder Säulen immer in den selben Segmentpunkt setzen willst, alle gleich ausgerichtet... etwa alle zum Zentrum blickend, oder der Biegung des Tunnels entlang. Platziere das Objekt einfach dort wo Du es haben willst, dann kopiere es. markiere alles, rotiere, und verwende Edit->Paste on Top. Weiterrotieren, Paste on Top, Weiterrotieren,...solange bis Du fertig hast.

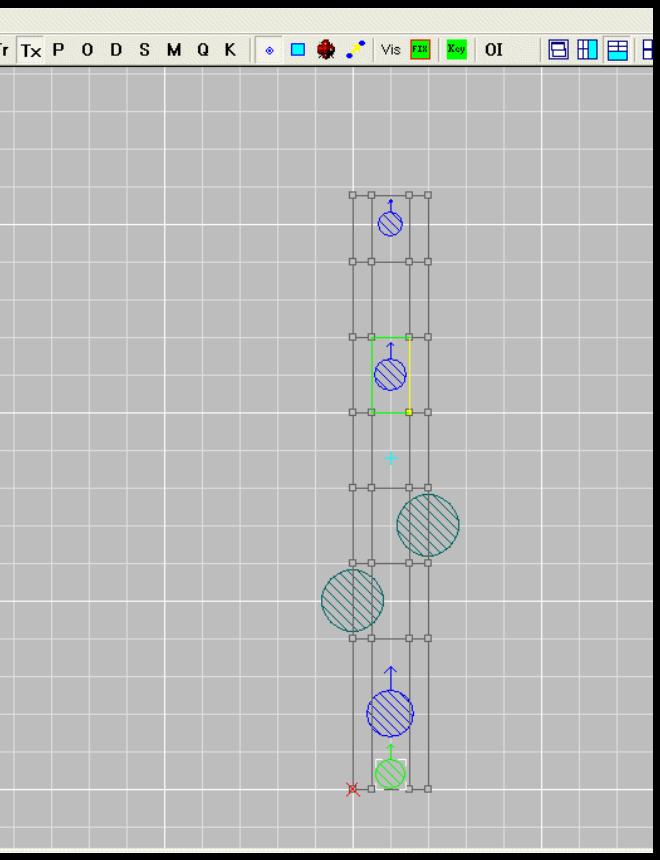
Ein letzter Punkt. Erinnerst Du dich was ich darüber gesagt habe, daß die Texturen bei 2.8125° sind? Das bedeutet mit was immer Du auch ausrichten willst. Im Falle unseres Tunnels sind es die Innen- und Außenwände. Wenn die Nähte bei 22.5° mit unserem Tunnel sind, sind die Außenwände bei 11.25°... einfach halbieren (Bild rechts).

Nimm Dir fünf Minuten um das zu lernen und eine um es anzuwenden, und Du wirst Meister der Bend- und Lathe-Tools.

038 - Benden, Objekte inklusive

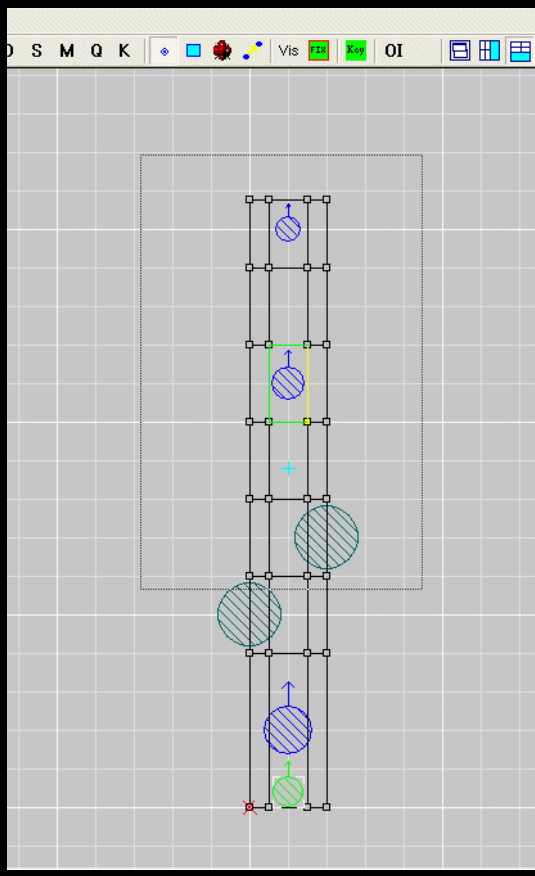
Atan

(V1.0)

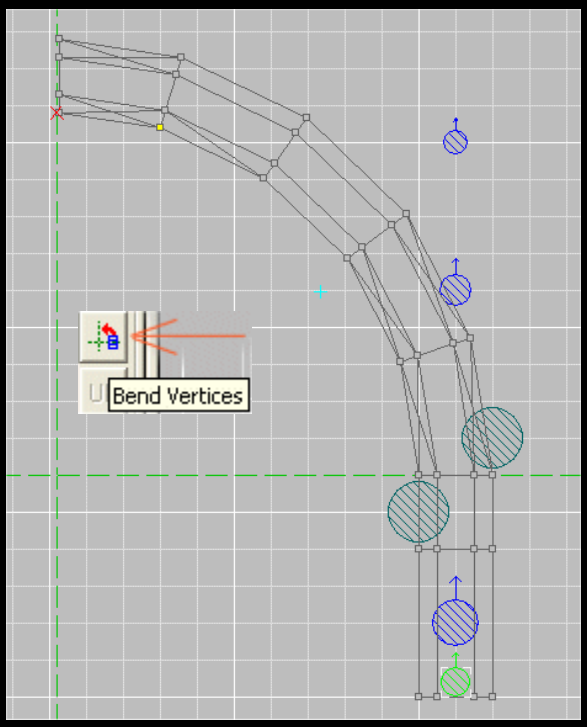


Benden ist eine Sache für sich. Entweder man hat den Dreh raus, oder es bleibt einem für immer ein Rätsel. Aber egal ob Profi oder nicht, vielen ist noch unbekannt, dass man mit der neuen Bend Version Objekte sofort mit bewegen kann. Nicht perfekt, aber schon mit ein wenig Nacharbeit bekommt man bereits ein brauchbares Ergebnis.

Zuerst müssen wir die Vertices markieren die gebogen werden sollen, dazu ziehen wir mit der Maus einen Rahmen um den gewünschten Bereich (rechts)

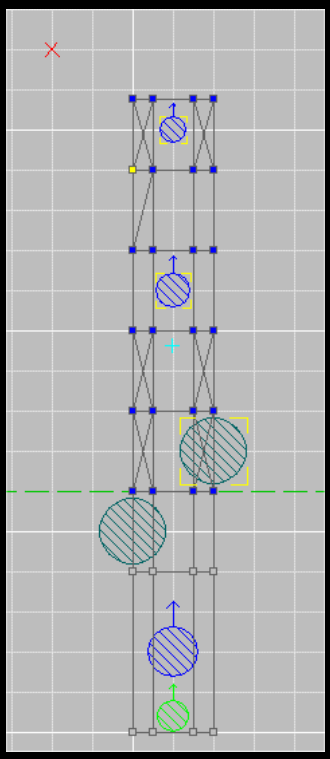


Jetzt sind die Vertices markiert und wir betätigen die Bend-Taste und benden mit den gegebenen Voreinstellungen.Leider sieht unser Ergebnis nicht wie erwartet aus, die Objekte bleiben alle auf ihren Positionen (unten links).

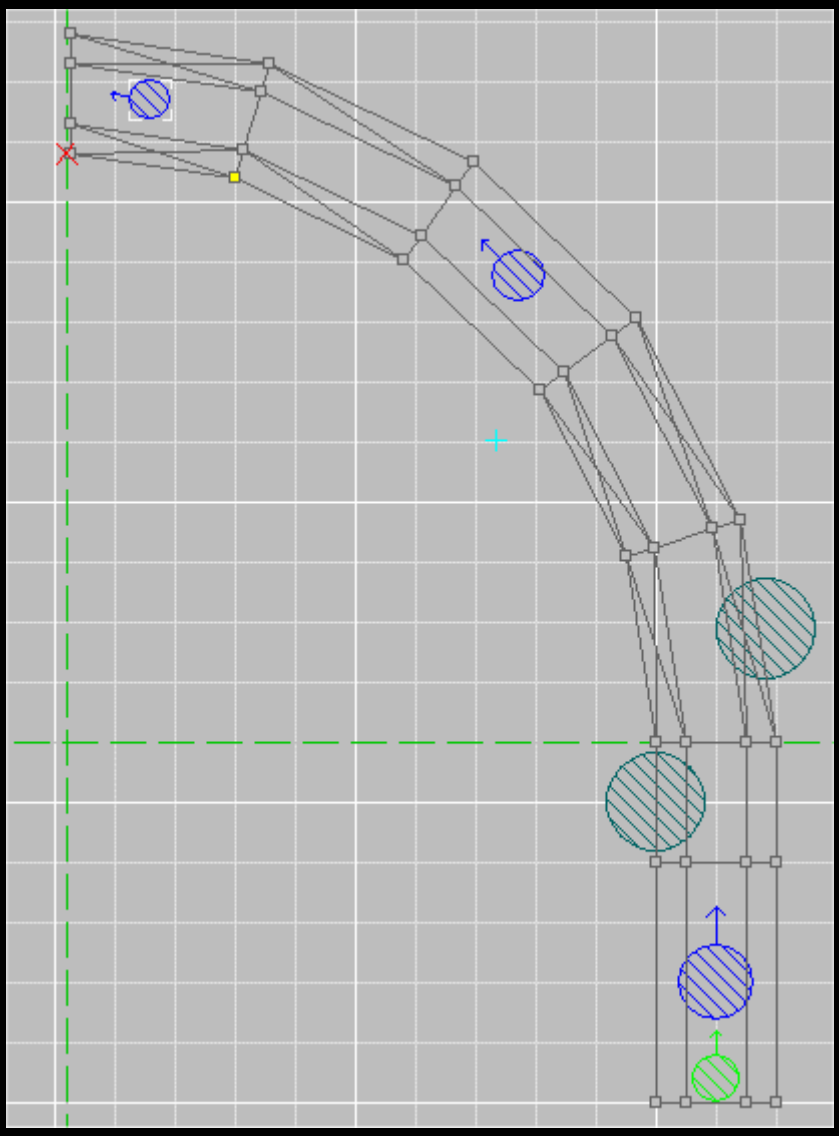


Umfangreiche Nacharbeiten sind nun nötig geworden um die Objekte zu positionieren.
Mit der UNDO Bend Taste bringen wir den Raum wieder in den Urzustand und markieren vor dem Benden auch die Objekte.

Dazu ziehen wir den Mausrahmen um die gewünschten Objekte, vorher mit **Ctrl-G** in den Object Mode gehen.



Die gelben Rahmen identifizieren die markierten Objekte.
Wir rufen wieder den Bend Dialog auf und benden unseren Raum mit den gegebenen Voreinstellungen.
Das Ergebnis sieht so aus:



Noch ein wenig nacharbeiten und die Sache ist erledigt.

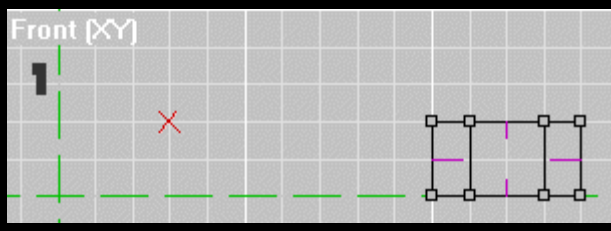
File: 039-ohne_bend.zip

039 - Biegung ohne Bend

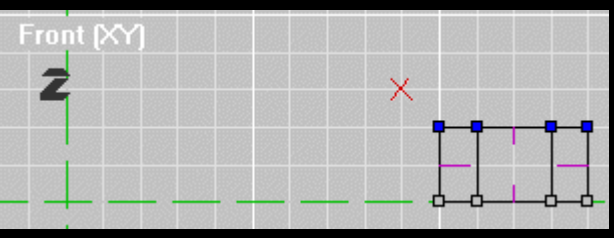
Fischlein

(Ich hab mir erlaubt, diesen Artikel zu überarbeiten.)

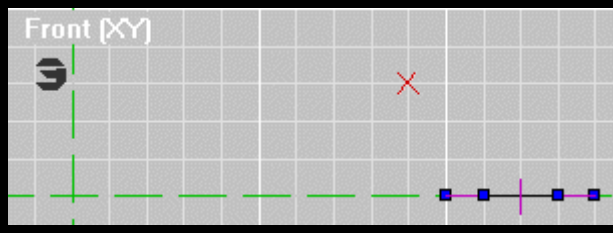
Für alle die sich den Raum/Biegung ansehen wollen, können sich der File bedienen. Die Gittergröße ist hier auf 10 eingestellt.

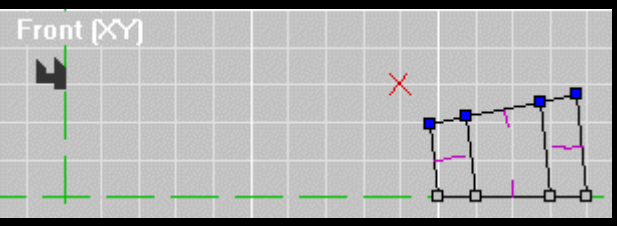


Zuerst habe ich ein Face erstellt, mit sechs Eckpunkten (Verts) und dieses extrudiert. Wie weit ist eigentlich egal, nur nicht zuweit. Hinweis: Lösche die oberen und unteren Faces, dann hast Du am Ende weniger Arbeit!

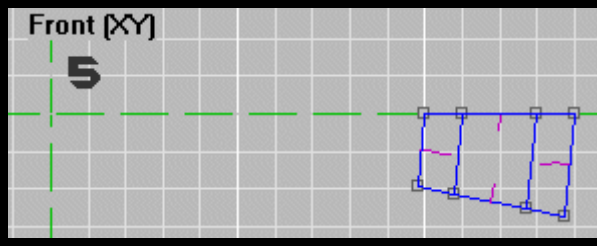


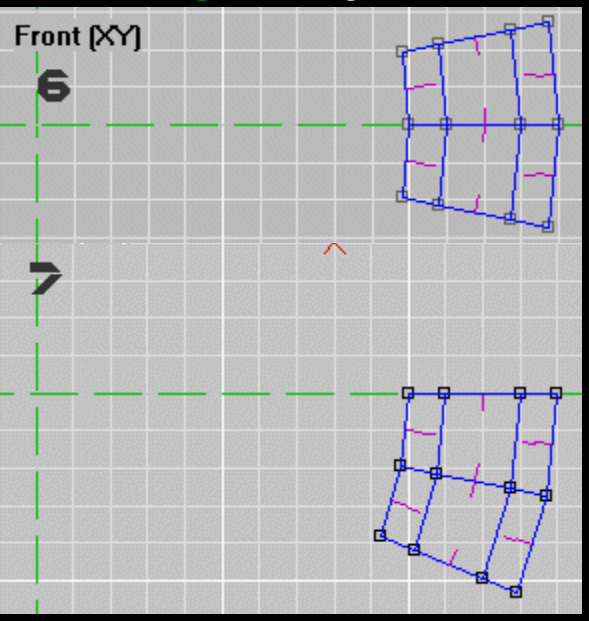
Wie Du im Bild 2 sehen kannst, habe ich die oberen Verts markiert, dann habe ich die **2** auf dem NumBlock 2x betätigt (was die Vertices ineinanderschiebt, Bild 3). Dann habe ich **1** betätigt und wie Du im Bild 4 siehst beginnt der Bogen.



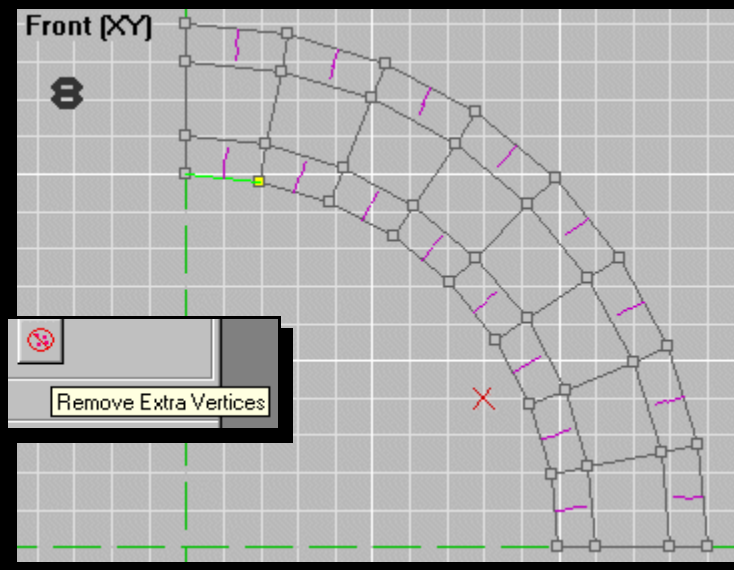


Ich habe dann mit der Taste **M** alle Verts Markiert und bin mit **Strg+F** in den Face Modus gewechselt und habe dann nochmals die Taste **M** betätigt, kurz gesagt ich habe alle Faces und Verts markiert, um alles mit **Strg+C** zu kopieren. Jetzt markiere ich alle Faces und drücke die Taste **3** auf dem NumBlock. Wie Du siehst hat sich das Tunnelstück nun nach unten gedreht (Bild 5). Dann habe ich die Tasten **Strg+Shift+V** gedrückt, was das in der Zwischenablage wieder an dieselbe Position einfügt (= Edit->Paste on Top, Bild 6). Dann drücke ich wieder die Taste **3**.





Das wiederhole ich so lange, bis ein Tunnel mit einer 90° Biegung entstanden ist. Das Ergebnis sieht dann so aus wie in Bild 8: Aber der Tunnel ist noch nicht fertig! Wir müssen nun noch die Extra Verts entfernen, ein Klick auf den Button im Bild rechts und fertig.



Wenn Du den Raum in Deine Level einfügen willst, musst Du nun wieder die Faces an den beiden Enden einfügen. Dazu markierst Du zuerst an einem Ende die Verts, wechselst in den Face Modus und drückst die Taste **Einfg**, dasselbe noch mal an dem anderen Ende, Fertig. Der große Vorteil dieser Methode ist, daß es immer paßgenau sein wird und auch die Texturen ausgerichtet sind.

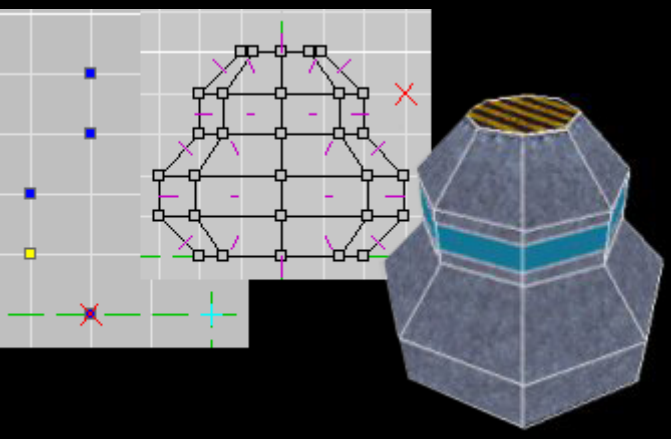
Achtung: Für andere 2d-Ansichten gelten andere Tasten; Macht man das alles z.B. in der Top-View, sind **1** und **3** vertauscht. Herumprobieren ist angesagt!

040 - Lathe – Vertiefung

Hydra

(wurde überarbeitet)

Überblick



Lathe ist eine nette Funktion die man da zur Verfügung hat. Obwohl ich sie nicht sehr oft in meinen Leveln verwende, kann sie wirklich nette Effekte machen. Diese Funktion ist perfekt, wenn Du ein wohlgerundetes Objekt oder Raum in Deinem Level brauchst.

Die Lathe-Funktion ist schnell erklärt. Nachdem Du ein Set an Vertices entlang einer Achse platziert hast, gehst Du in den Lathe-Dialog und gibst ein paar Optionen ein. Nachdem Du diese bestätigt hast, werden die Verts um eine eingestellte Achste rotiert/kopiert. Ihre Anwendung funktioniert gut, wenn Du einen hübschen runden Korridor oder nur ein paar große, gleichmäßig aussehende Pfeiler brauchst.

Das größte Problem bei dieser Funktion ist, daß sie wirklich viele Faces erzeugen kann. Wenn Du ein Objekt zu detailliert machst, hast Du vielleicht mit nur einem Lathe-Vorgang 200 Faces erzeugt. Du mußt damit vorsichtig sein, wenn Du Räume machst. Nochmal, das Beste an dieser Funktion ist daß sie keine Bad Shells oder T-Joints in Deinem Raum erzeugt, was Dir Zeit spart.

Der Lathe-Dialog

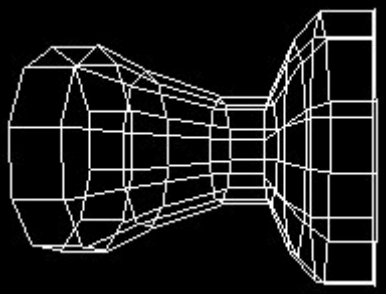
Rotate around: [X, Y, Z]
Rotiert/kopiert die markierten Verts um die [X, Y, Z]-Achse.

New Face Normals should:
Point inward – Alle Normals der neuen Faces schauen in den Raum hinein. Das wird verwendet um Raum-Shells zu erzeugen.
Point outward – Alle normals gucken nach außen, oder voneinander weg. Dies wird meist für Pfeiler oder Röhren verwendet.

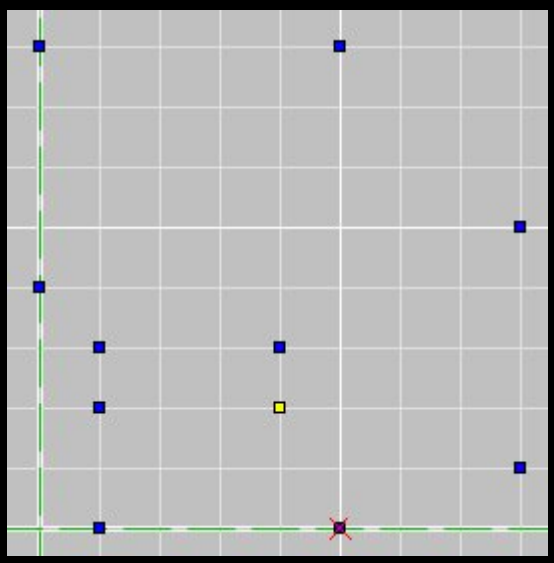
Sides: Wie oft die Verts Rotier-kopiert werden. Wenn Du zum Beispiel 4 eingibst, wird ein quatratisches Objekt entstehen, wenn Du 8 eingibst ein achteckiges.
Create End Caps: Erzeugt zusätzlich oben und unten Faces auf Deinem Objekt.

Randnotiz: Ich mache diese Tutorials sehr frei, hauptsächlich weil Du hier nicht nur Fertigkeiten im Räumebauen erwirbst, sondern es Dir erlaubt in einer von Dir erzeugten Umgebung herumzuexperimentieren.

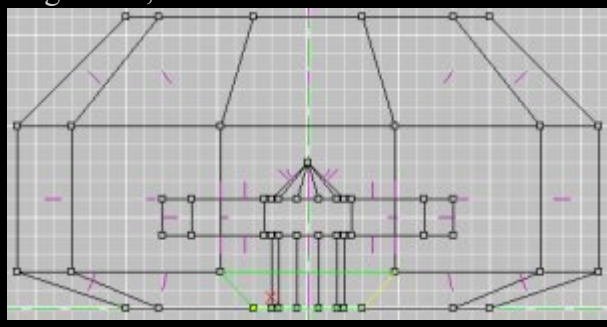
Anwendung 1: Extern



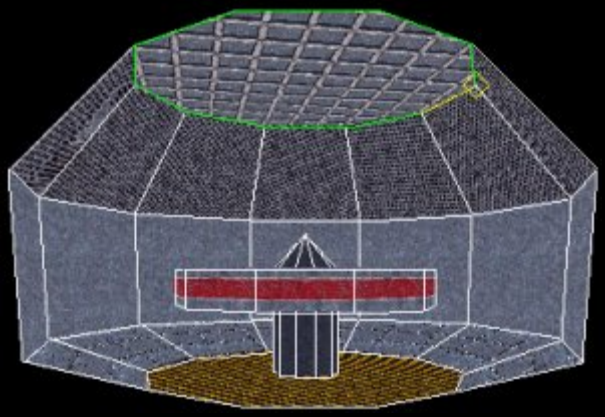
Ganz wie beim Extrudieren ist beim Lathen von großen Bereichen Vorausplanung das Wichtigste. Du solltest wissen, welche Texturen du verwenden wirst, wo Du sie verwenden wirst, wo andere Räume angehängt werden und vielleicht sogar wo Pfeiler oder Anderes hinkommt. Dies ist eine ganze Menge zu bedenken, weswegen ich für gewöhnlich versuche alle meine Ideen auf Papier zu bringen bevor ich anfange.



Es gibt grunsätzlich zwei Typen von Räumen, die Du mit Lathe machen kannst. Du kannst einen großen rund geformten Raum machen, oder Du kannst einen donut-förmigen Korridor machen. Ich werde einen großen Raum machen; hauptsächlich weil es nicht empfohlen ist Korridore mit Lathe zu machen, da der gesamte Korridor ein Raum ist, was ein wenig Extra-Lag erzeugen als auch die FPS mindern kann. Ich habe meinen Raum angefangen indem ich einfach die Verts dorthingestellt habe wo ich sie wollte; bemerke, die Form welche sie umreißen muß nicht konvex sein.

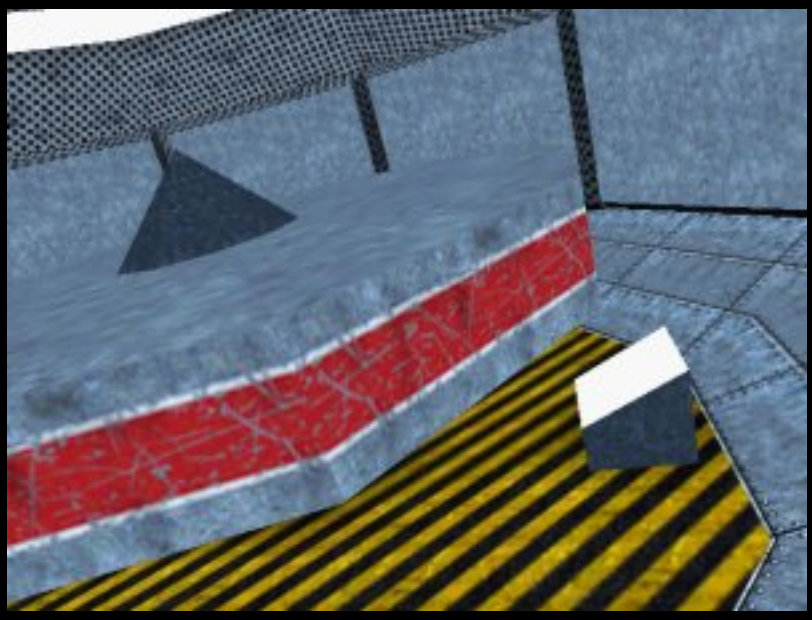


Ich habe diese Verts um die Achse gelathet, und sofort hatte ich ein paar Bad Normals. Ich mußte durch jedes Face durch und die ‚bösen' rauspicken und löschen, was nicht zu lange gedauert hat. Ich habe begutachtet was mir überblieb und stellte sicher daß es korrekt gelathet wurde und daß die Normals in die Richtung schauten die ich wollte. (kommt anschließend)

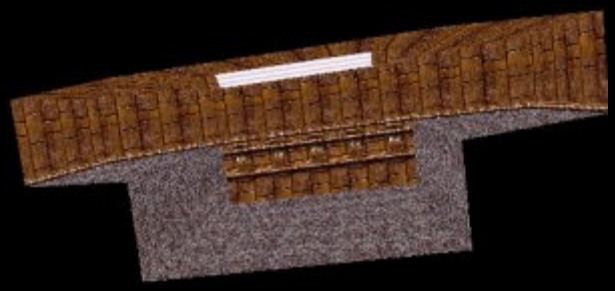


Danach habe ich Texturen hinzugefügt und ein paar Tweaks an einigen Verts vorgenommen, welche durch das Lathen eine oder zwei Einheiten deplatziert worden waren.

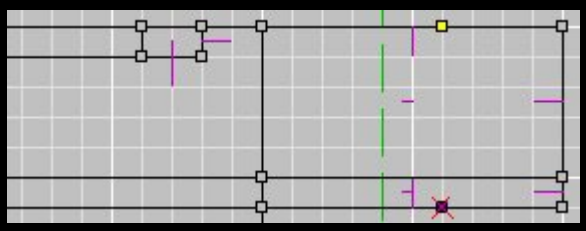
Ich habe etwas Beleuchtung hinzugefügt (eigentlich ein bischen zu viel) und ein paar andere Sachen. Dieses Level wäre viel zu klein um einen vernünftigen Dogfight zu haben, aber es wäre ein guter Verbindungsraum zwischen zwei anderen Räumen.



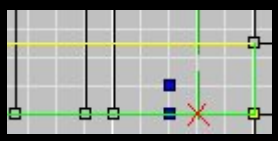
Anwendung 2: Intern



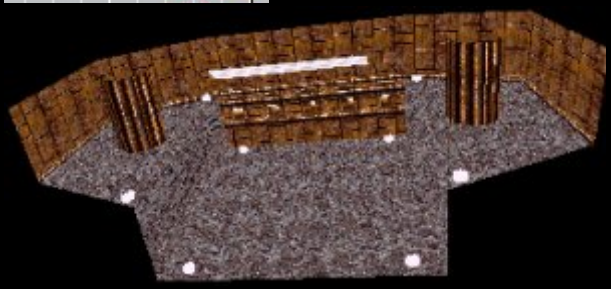
In dieser Lektion gebe ich Dir ein paar Beispiele, wie man die Lathe-Funktion verwendet um damit Indoor-Objekte zu erzeugen, etwa Pfeiler. Ich werde den Raum verwenden, den ich für meine Extrude-Tutorials (kommt anschließend!) gemacht habe, weil ich seinen Aufbau mag.

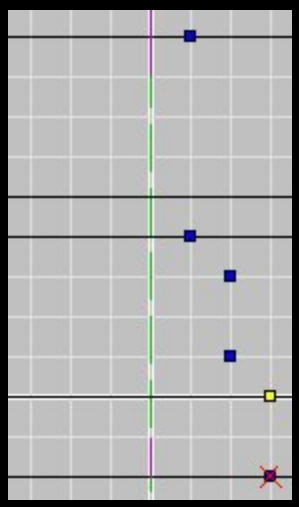


Als erstes möchte ich ein paar Pfeiler entlang der Kanten machen, was den Spielern erlaubt Feuer auszuweichen und Homing-Waffen leichter zu entkommen. Du willst Deine Räume spielbar und gut aussehend. Beachte, daß ich den Referenzpunkt bewegt habe (die grünen Linien welche die Achsen definieren). Vergiß nicht daß Lathen immer UM die Achse herumgeht, also mußt du die Achse dort positionieren, wo du das Zentrum deines Lathes haben willst. Du kannst den Referenzpunkt durch Ctrl-Klick in eine 2d-Ansicht neu positionieren.



Nachdem ich einen Pfeiler gelathet habe, habe ich dasselbe auf der anderen Seite gemacht. Danach habe ich einige Lichter im Raum verteilt, indem ich wie bei den Pfeilern gelathet habe, nur viel kleiner. Die 2d-Ansicht zeigt die Lichter vor, die 3d-Ansicht darunter nachdem ich sie alle im Raum gelathet habe.





Danach bin ich draufgekommen, daß dieser Level ein reaktor-ähnliches Objekt brauchte. Die sehen wirklich kool in einem Level aus und sie sind recht einfach zu machen. Ich habe dasselbe wie bei den Pfeilern gemacht, nur daß ich die Verts mehr versetzt angeordnet habe, also nicht in einer Linie. Danach habe ich sie gelathet und das Ergebnis texturiert.

Und da ist das Resultat in D3. Wie Du sehen kannst, sogar ein paar einfache Lathings können einen Raum ausgezeichnet aussehen lassen. Obwohl, wie beim Extrudieren, mußt Du sicherstellen daß der Level selber einfach zu navigieren ist, sodaß die Spieler während eines Feuergefechtes nicht in den Wänden hängenbleiben. (das kennen wir doch alle...?)

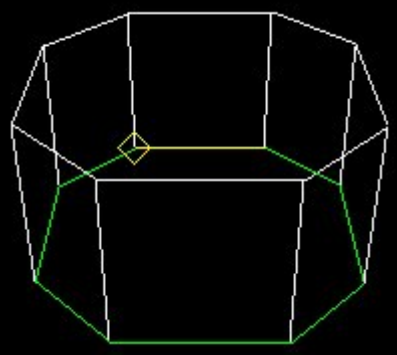
041 - Extrude – Vertiefung

Hydra

(wurde überarbeitet)

Übersicht

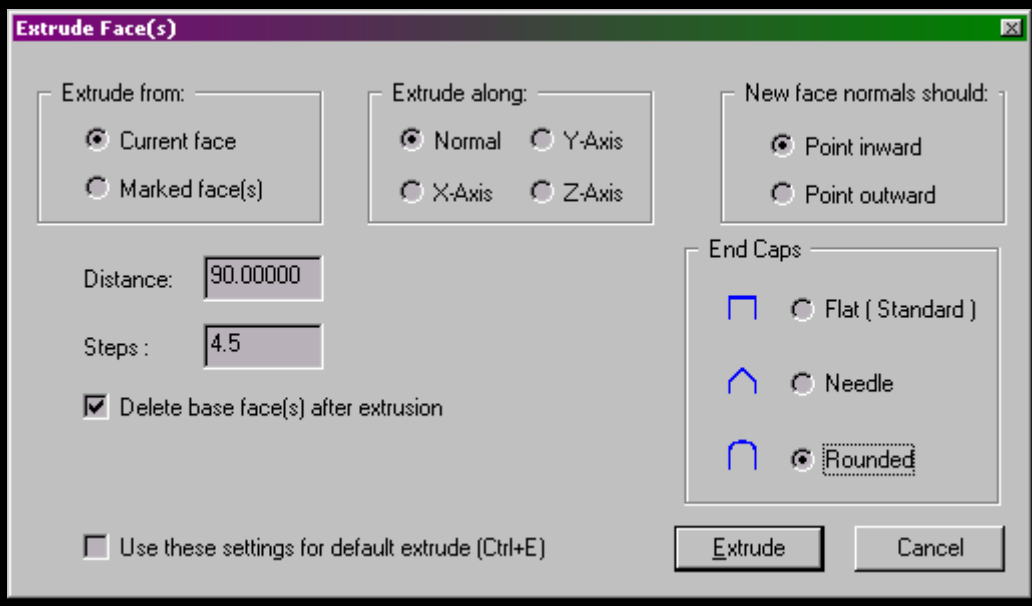
Extrude ist nicht nur die grundlegendste aller drei geometrischen Aktionen, sondern es ist auch die welche ich am meisten benutze. Diese Funktion ist nicht nur perfekt um Säulen und Pfeiler schnell zu erstellen, sondern sie kreiert auch effizient eine ‚versiegelte' Raum-Shell mit ein paar einfachen Klicks.



Die Extrude-Funktion ist schnell erklärt. Alles was sie macht ist, daß sie ein gewähltes Face kopiert und ein zweites eine bestimmte Distanz entfernt einfügt, danach erzeugt sie Faces seitlich ‚rundherum', und versiegelt sie damit. Das Bild zeigt ein extrudiertes Achteck. Du siehst, daß sie eine exakte Kopie des gewählten Faces macht und dann die ‚Seiten' hinzufügt, was es zu einem Raum macht.

Hast Du erstmal heraußen wie diese Funktion tut, wirst Du nicht nur die Shells von einfachen Räumen machen können, sondern auch Shells von komplexen Räumen wie den da rechts. Und das Beste ist, die Funktion wird Dir keine T-Joints machen oder Bad Shells in Deinen Räumen.

Der Extrude-Dialog (v39)

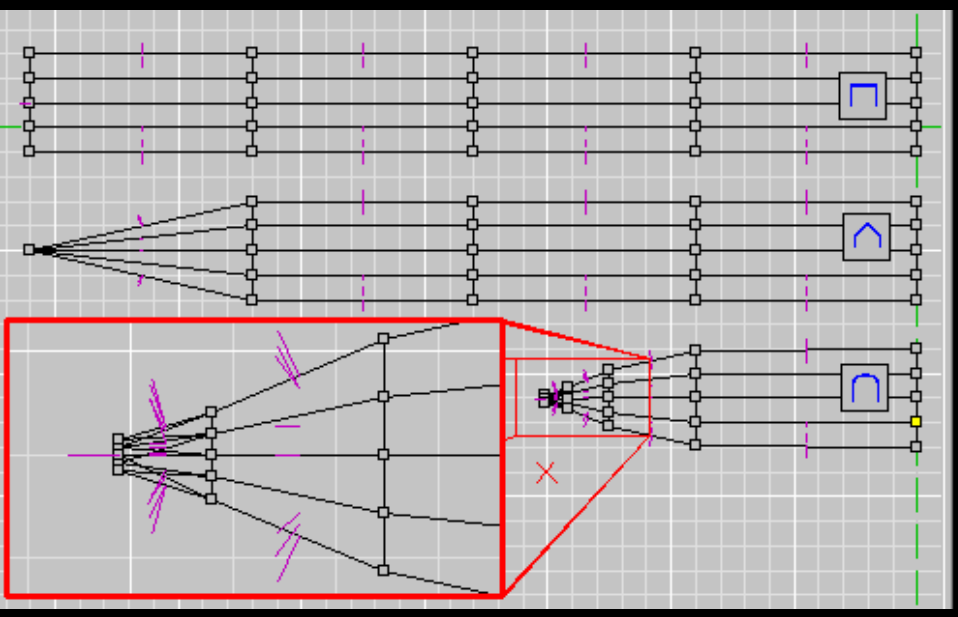


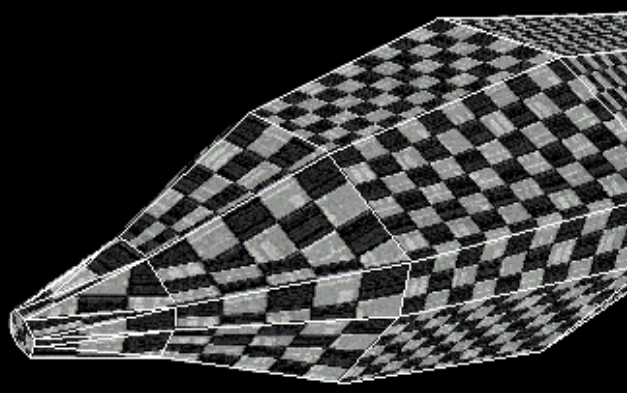
Extrude from:
Current Face – Extrudiert vom ausgewählten (grün/gelben) Face aus.
Marked face(s) – Extrudiert von allen markierten (blau) Faces heraus.
Extrude along:
Normal – Dies ist die Einstellung die ich wirklich verwende, sie extrudiert in die Richtung, wo die Normal des Faces hinzeigt.
[X, Y, Z]-Axis – Zwingt D3Edit dazu das Face in die [X, Y, Z]-Achse zu extrudieren, hilfreich um von einem schrägstehenden Face gerade wegzuextrudieren.

New Face Normals should:
Point inward – Alle Normals der neuen Faces schauen in den Raum hinein. Das wird verwendet um Raum-Shells zu erzeugen.
Point outward – Alle normals gucken nach außen, oder voneinander weg. Dies wird meist für Pfeiler oder Röhren verwendet.
Distance – Erklärt D3Edit, wieweit weg das zweite Face positioniert werden soll.
Steps – wie oft die Extrusion durchgeführt werden soll. Jeder Step geht über die volle Distance.
Delete base face(s) after extrusion – Dies wird die Faces von denen Du wegextrudiert hast löschen (oder in anderen Worten, die gegenwärtig markierten/ausgewählten Faces). Das ist hilfreich, wenn Du multiple Extrusionen vornimmst oder das Basis-Face, mit dem Du beginnst nicht sichtbar ist.

Use these settings for default extrude – Dies setzt Deine gegenwärtigen Einstellungen als eine Art ‚Standard'. Dann mußt Du nur **Ctrl-E** drücken und es wird, ohne den Dialog überhaupt nur anzuzeigen, losextrudiert.
End Caps:
Flat – In etwa dasselbe wie die End Caps im Lathe-Dialog. Das Extrusionsergebnis wird mit einem Face abgeschlossen.
Needle – Im letzten Extrusionsschritt wird nicht auf ein Face, sondern auf ein Vertex hinextrudiert.
Rounded – Hm... interessant, seht selbst.

Die verwendeten Einstellungen entsprechen dem oben abgebildeten Dialog.

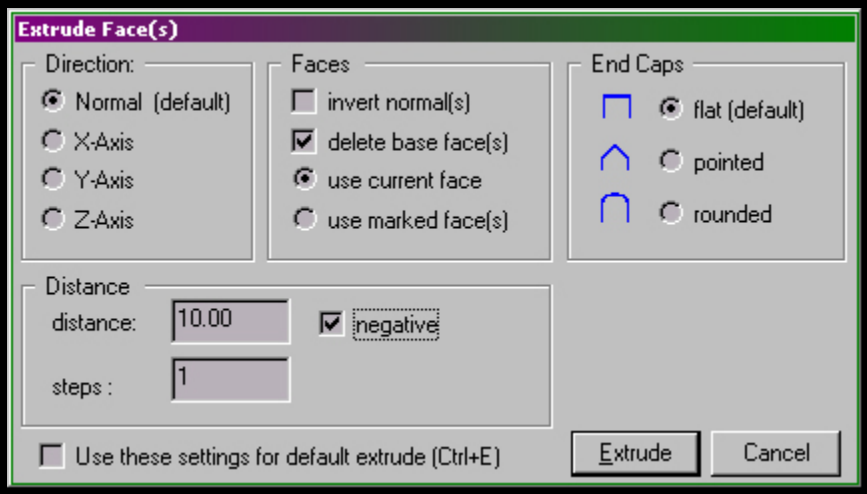




Randnotiz: Ich mache diese Tutorials sehr frei, hauptsächlich weil Du hier nicht nur Fertigkeiten im Räumebauen erwirbst, sondern es Dir erlaubt in einer von Dir erzeugten Umgebung herumzuexperimentieren.

Update für v40:

Die Extrusion wurde überarbeitet. Der neue Dialog sieht so aus:



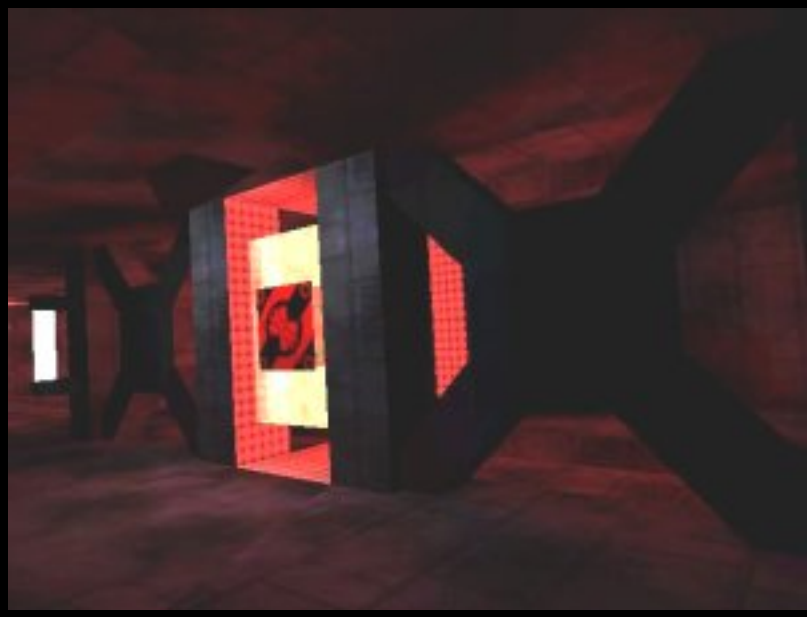
Direction und End Caps bleibt sich gleich, geändert wurden Distance und Faces.

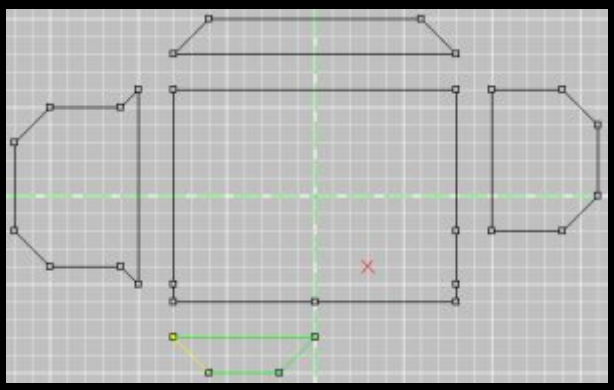
Die Normals von Neuen Faces schauen nun per Standard immer in die selbe Richtung wie diejenige(n) der Basisface(s).

Bei Distance ist jetzt negative anhakbar; diese Einstellung bewirkt Extrusion entgegen der Face-Normal(s) des/der Basisface(s).
Unter Faces gibts noch die Einstellung invert normals(s), dies dreht die Normals in Relation zum Basisface um.
Die neue Extrusion fühlt sich auch ein wenig anders an, ist aber intuitiver; spielt einfach mal ein wenig herum und denkt es durch.

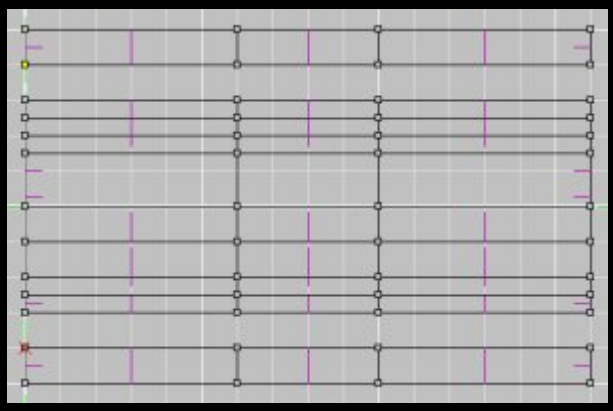
Anwendung 1: Extern

Eines der wichtigsten Dinge beim erschaffen eines Raumes ist es vorauszuplanen. Das ist für mich sehr schwer, da ich lieber improvisiere während ich baue als mich hinzusetzen und alles im Kopf, oder besser auf einem Blatt Papier, aufzuzeichnen. Ich sage nicht, daß Du Deinen gesamten Level komplett ausgeplant haben mußt, mit allen Röhren und Pfeilern, sondern daß Du eine grobe Idee haben mußt wie du texturieren wirst und wo angehängte Räume sein werden. Ich zeige das Bild rechts als ein Beispiel schlechter Planung. Ich habe dafür nicht vorgeplant, und es hat mich fast drei Tage gekostet nur im diesen einen simplen Raum korrekt zu funktionieren zu kriegen. Daher schätze ich, daß dieses Bild nur mir etwas bedeutet ☺

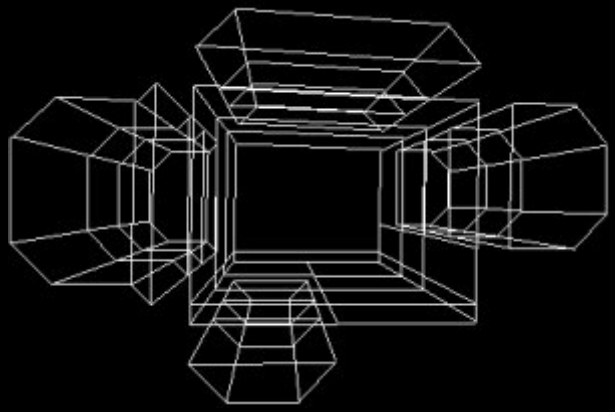


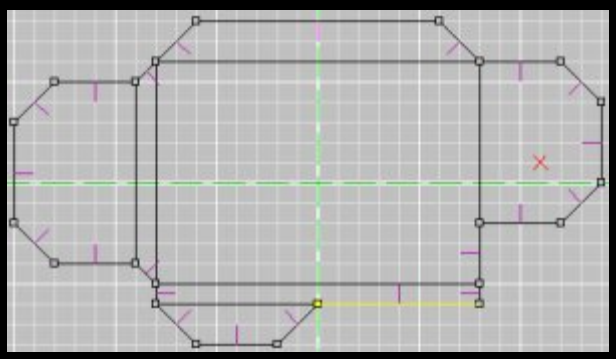


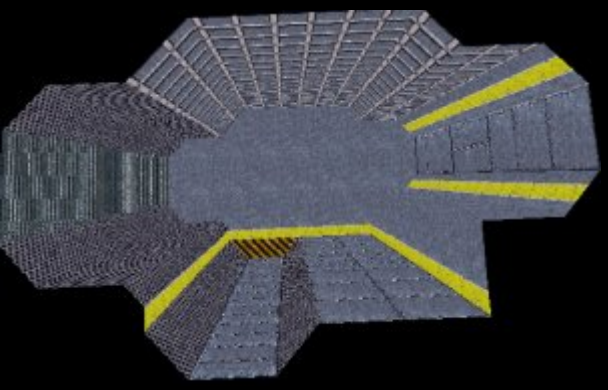
Hast Du erstmal eine Idee, wie Dein Raum aussehen wird, kannst Du beginnen Teile davon zu extrudieren. Ich möchte, daß mein Raum ein großer, komplex aussehender Korridor ist. Ich habe den Raum bereits in meinem Kopf zusammengesetzt (und auf einem Stück Papier, ich vergaß ☺ wie er aussehen soll, also beginne ich damit die ‚Teile' des Raumes zu machen. Das linke Bild ist die Frontansicht (XY) von dem was ich bis jetzt habe.



Danach habe ich alle Faces markiert und sie einmal 60, dann 40 und dann nochmal 60 Einheiten extrudiert. Beachte, daß ich einige Faces mit Vertices versehen habe, die ich nicht wirklich brauche. Diese Extra-Faces die dadurch entstehen werden eigentlich für Extra-Texturen verwendet, das zu machen hilft Deinen Raum ein wenig realistischer zu machen. Dies ist ein Punkt wo das Vorausplanen ins Spiel kommt, weil Du nicht nur wissen mußt wo die Portale der anderen Räume sein werden, sondern wie groß diese Portale sein werden. Dies vorzeitig zu machen kann Dir einen Haufen Ärger mit T-Joints und Bad Shells ersparen. Das Bild rechts ist die Top- (XZ) und die 3d-Wireframe-Ansicht davon.





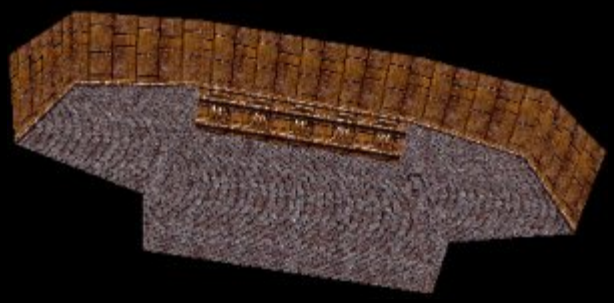


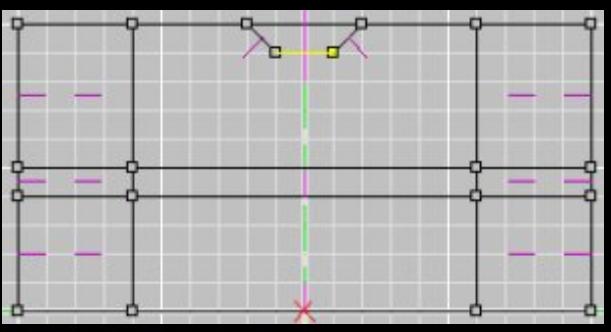
Nachdem ich das gemacht habe, nahm ich all die getrennten Sektionen (Ich halte sie getrennt, da es mir es erlaubt sie leicher zu überschauen) und schob sie wieder zusammen. Dann löschte ich die Faces und habe alle Extra-Verts entfernt, also das Level ‚versiegelt'. Dann habe ich ein wenig kreative Texturierung hinzugefügt, was ich wirklich mag. Das obere Bild (links) ist die wieder die Front-Ansicht, wo alle Segmente zu einem Ganzen vereinigt sind (Beachte daß die Normals, die üblicherweise blockierende Wände zu anderen Segmenten sind, gelöscht wurden), das untere ist eine texturierte Ansicht des Raumes wie er bis jetzt ist.

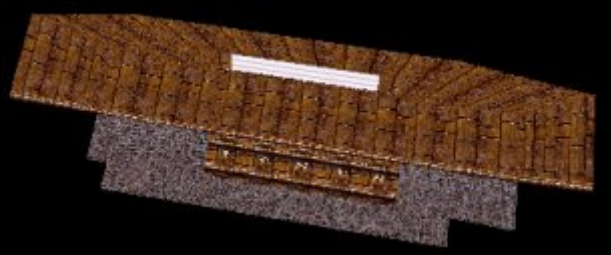
Dann bist Du mit der Shell, oder ‚Rahmen', des Levels fertig. Als nächstes werde ich Dir zeigen, wie man Pfeiler und andere Hindernisse hinzufügt, um Deinem Level einen realistischen Anstrich zu geben. Ich habe den Raum mit ein paar Pfeilern und Lichtern fertiggemacht und einen Screenie in D3 gemacht, ich denke es sieht ganz gut aus.

Anwendung 2: Intern

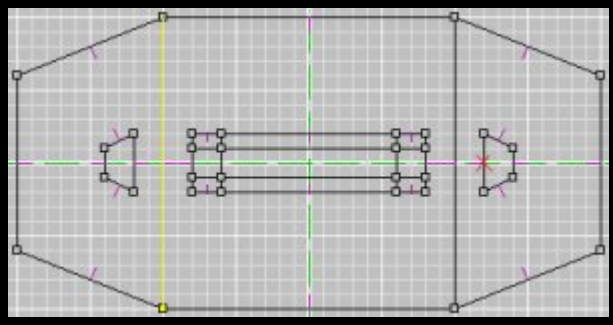
Pfeiler und verschiedene ‚Effekt'-Objekte wie Röhren zu extrudieren ist eine meiner Lieblingssachen beim Erschaffen eines Levels. Weil ich weiß daß ich bereits eine ‚versiegelte' Shell habe und ich mich nicht mehr damit herumärgern muß, kann ich jetzt einfach kreativ sein und die Räume selbst nett aussehen lassen. In dieser Übung werden wir ein paar einfache Objekte machen um den Raum zur Rechten aufzupeppen.

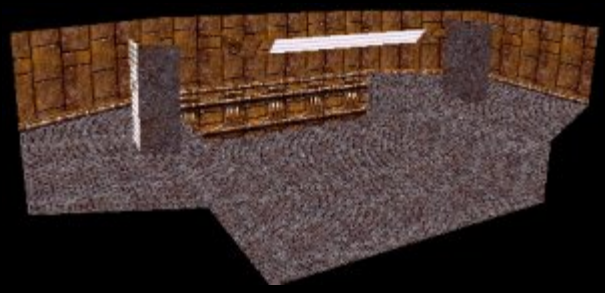




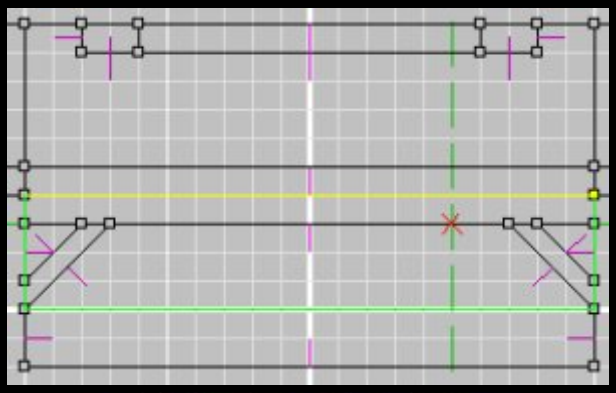


Als erstes habe ich entschieden meine Beleuchtung zur Decke hinzuzufügen. Ich erzeugte eine kleine Form wie ich wollte daß das Licht aussieht und extrudierte dieses Face dreimal durch den Raum, einmal 10 Einheiten, dann 60, dann nochmal um 10 Einheiten. Ich machte das, weil ich dann das Licht so aussehen lassen konnte, als würde es durch einen Teil der Struktur getragen werden. Links sind zwei Bilder, das erste ist die Form die ich verwendete (in der Side-View, ZY) und darunter die 3d-Ansicht.





Um die Beleuchtung fertigzustellen, habe ich noch zwei Lichter mehr an jeder Seite hinzugefügt, diese sollten nicht nur Teile des Raums, die vom großen Mittel-Licht nicht erreicht werden können ausleuchten, sondern auch die Portale zu den nächsten Räumen. Das war nur ein einfaches Extrude und Texturieren. Diese Pfeiler sind auch groß genug daß ein Spieler dahinter verstecken und vor Waffonfeuer Deckung nehmen kann (und Homing-Waffen ausweichen). Das obere Bild zeigt die Top-Ansicht (XZ) und darunter wieder die 3d-Ansicht.



Aus einer Laune heraus entschied ich, eine Brücke über den Zwischenraum hinzuzufügen. Es wäre kool, um zu dodgen unter der Brücke durchfliegen zu können, aber die derzeitige Tiefe des Zwischenraums ist nicht groß genug; ich extrudierte weitere 10 Einheiten vom Boden weg, was mir genug Platz für ein paar Arbeiten gab. Ich begann damit, die Brücke selbst zu machen, die mehr oder weniger ein Würfel ist. Danach habe ich ein paar Abstützungen hinzugefügt, indem ich sie in der Front-Ansicht erzeugt und seitlich extrudiert habe. Das linke Bild zeigt die Frontansicht (Die Raumseiten sind abgeschnitten) und die 3d-Ansicht.

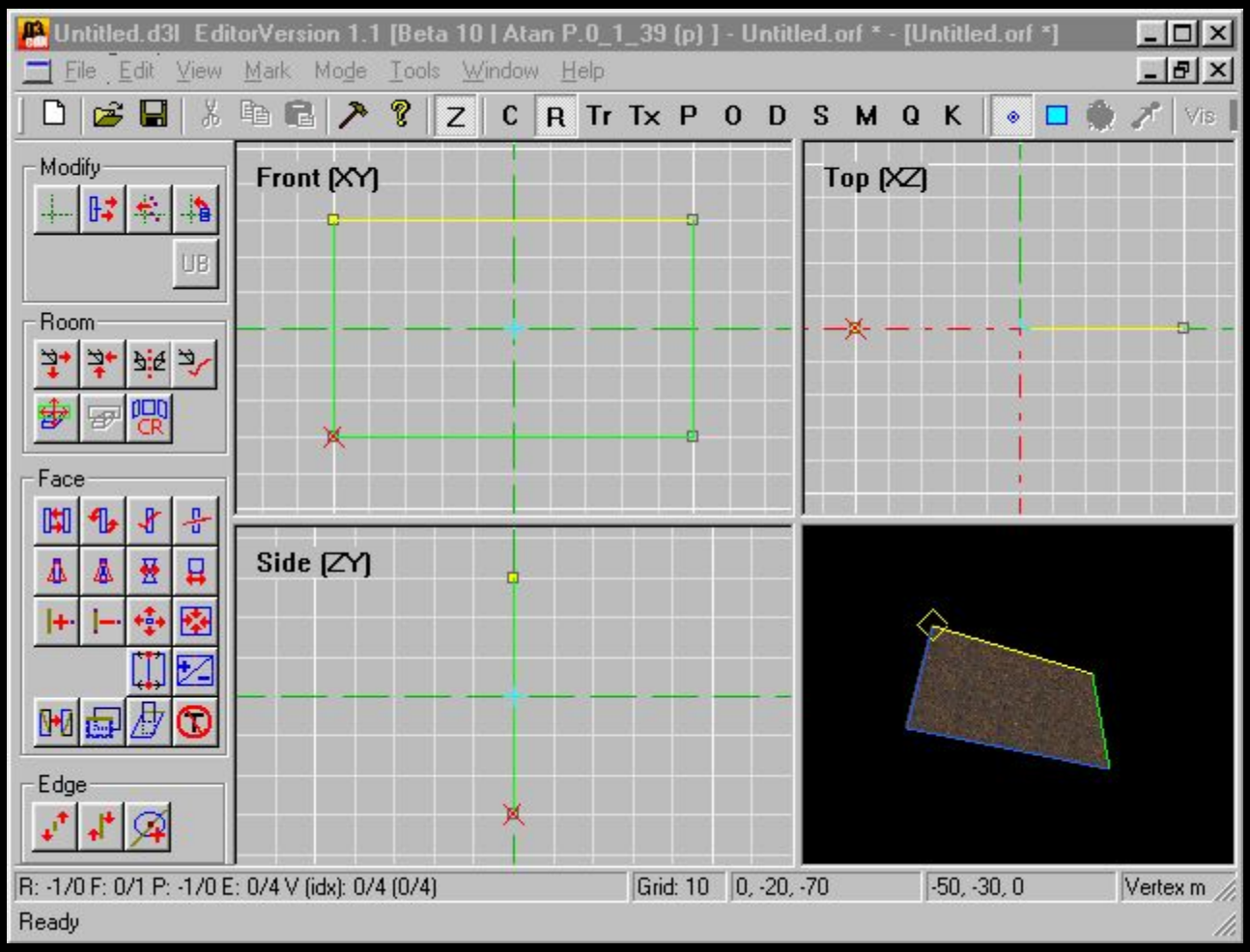
Und wenn wir's uns in D3 ansehen, sieht es recht gut aus. Obwohl es ein bischen finster ist, das ist der Effekt den ich in meinen Levels mag. Ich mag Orte, wo du Dich tief in den Schatten verstecken kannst. Wenn Du allerdings Extrusionseffekte wie diesen machst, mußt Du daran denken den Spielern einen weitläufigen Raum zu geben um zu manövrieren und nahendes Waffenfeuer zu dodgen. Obwohl Extrudieren wie hier Dein Level realistischer aussehen lassen kann, willst Du auch daß die Spieler im Spiel Spaß haben.

042 - Extrude: Zero

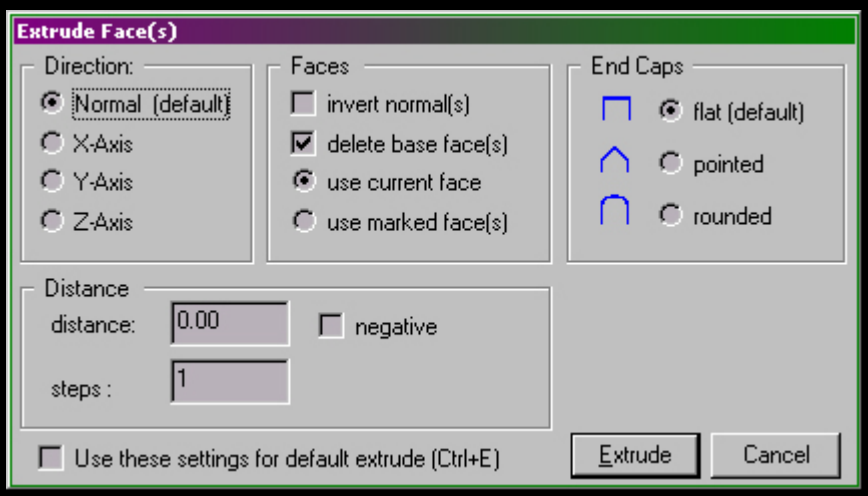
WillyP

(Aus der Descentpedia, http://www.prepare4descent.net/descentiapedia/DescentiaPedia)

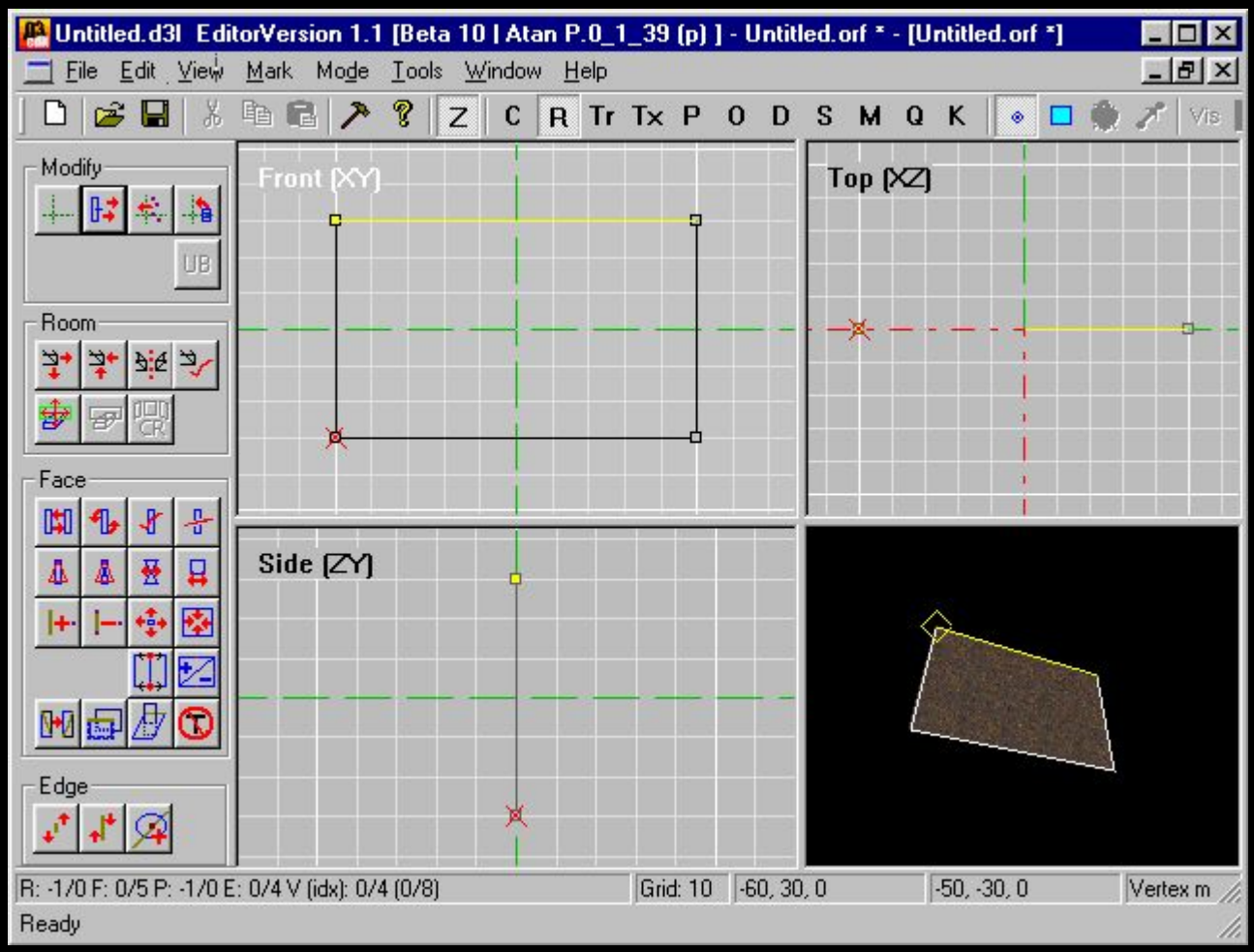
Während des Verwendens von D3Edit, aktuell v39, bemerkte ich eine kleine Macke welche beachtet werden sollte. Das betrifft auch v38/2, aber nicht v37. Ist keine große Sache, da es einen einfachen Workaround gibt... da ein Bild mehr als 1000 Worte sagt, beschreibe ich es bildlich:



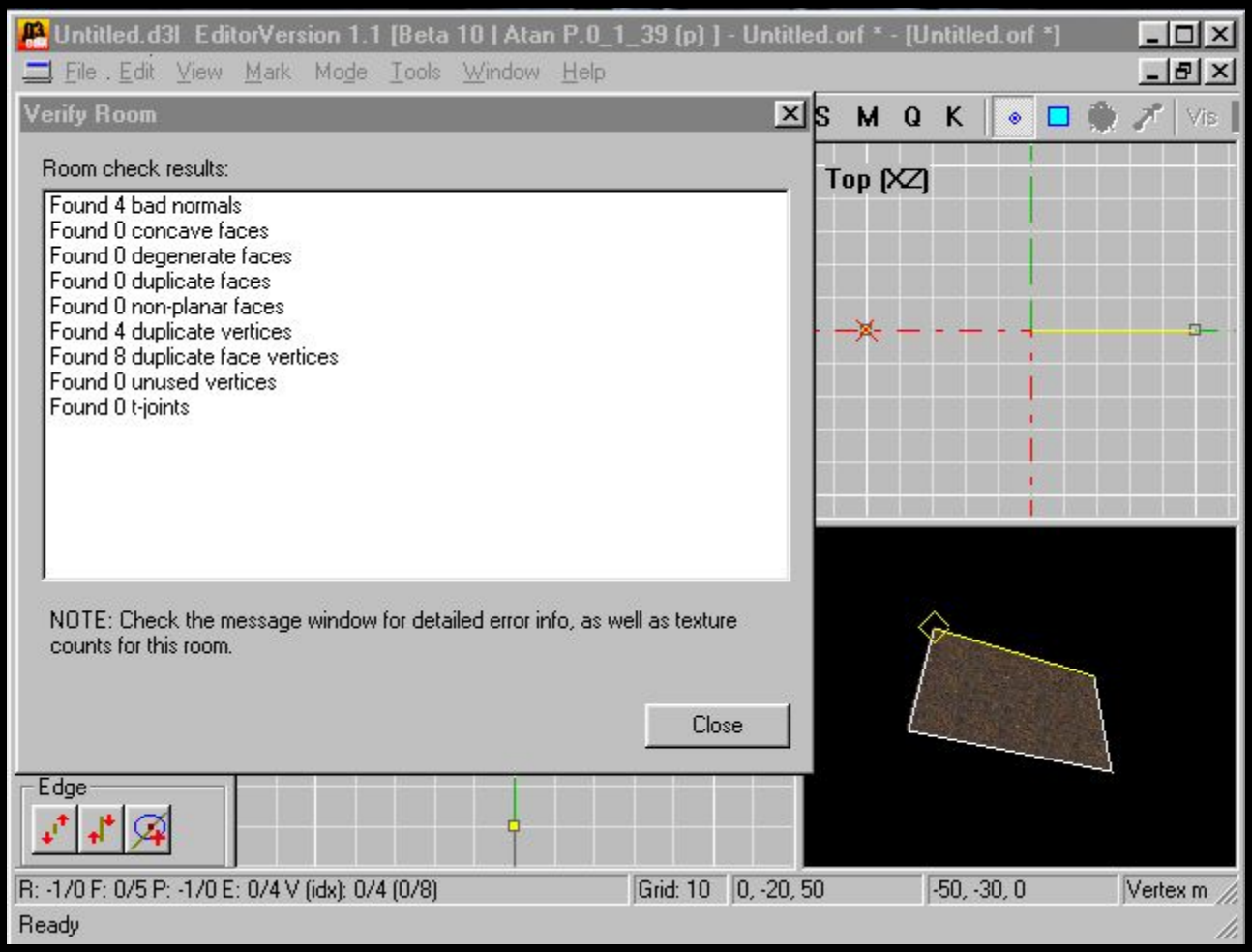
Ein Minimum von einem Face. Könnte Teil eines Raumes oder Level sein.



Öffne den Extrude-Dialog und setze zu 0 Einheiten, 1 Step, delete base, flat.

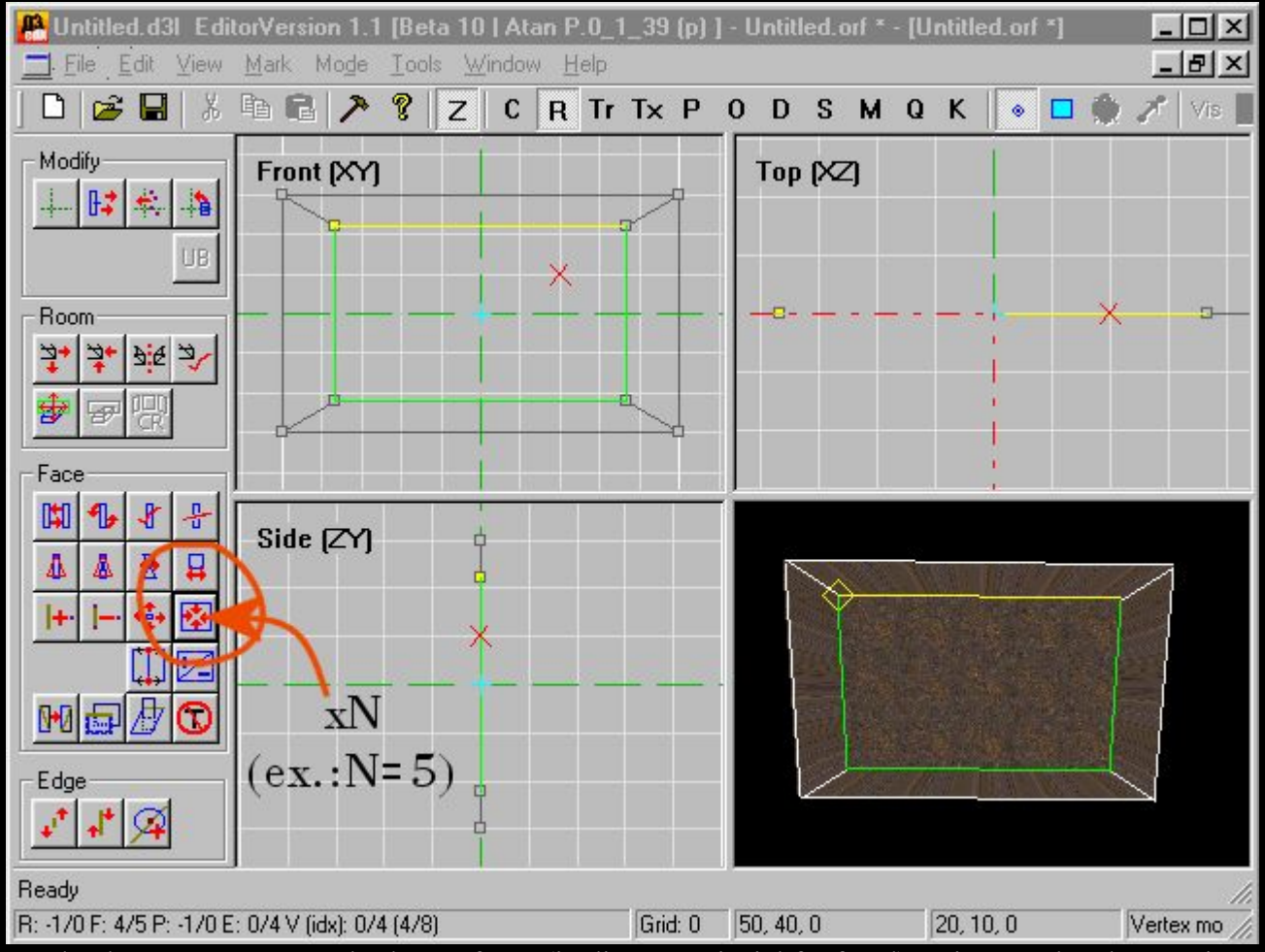


Das Face scheint sich nicht verändert zu haben, aber wir haben nun fünf Faces und acht Verts.

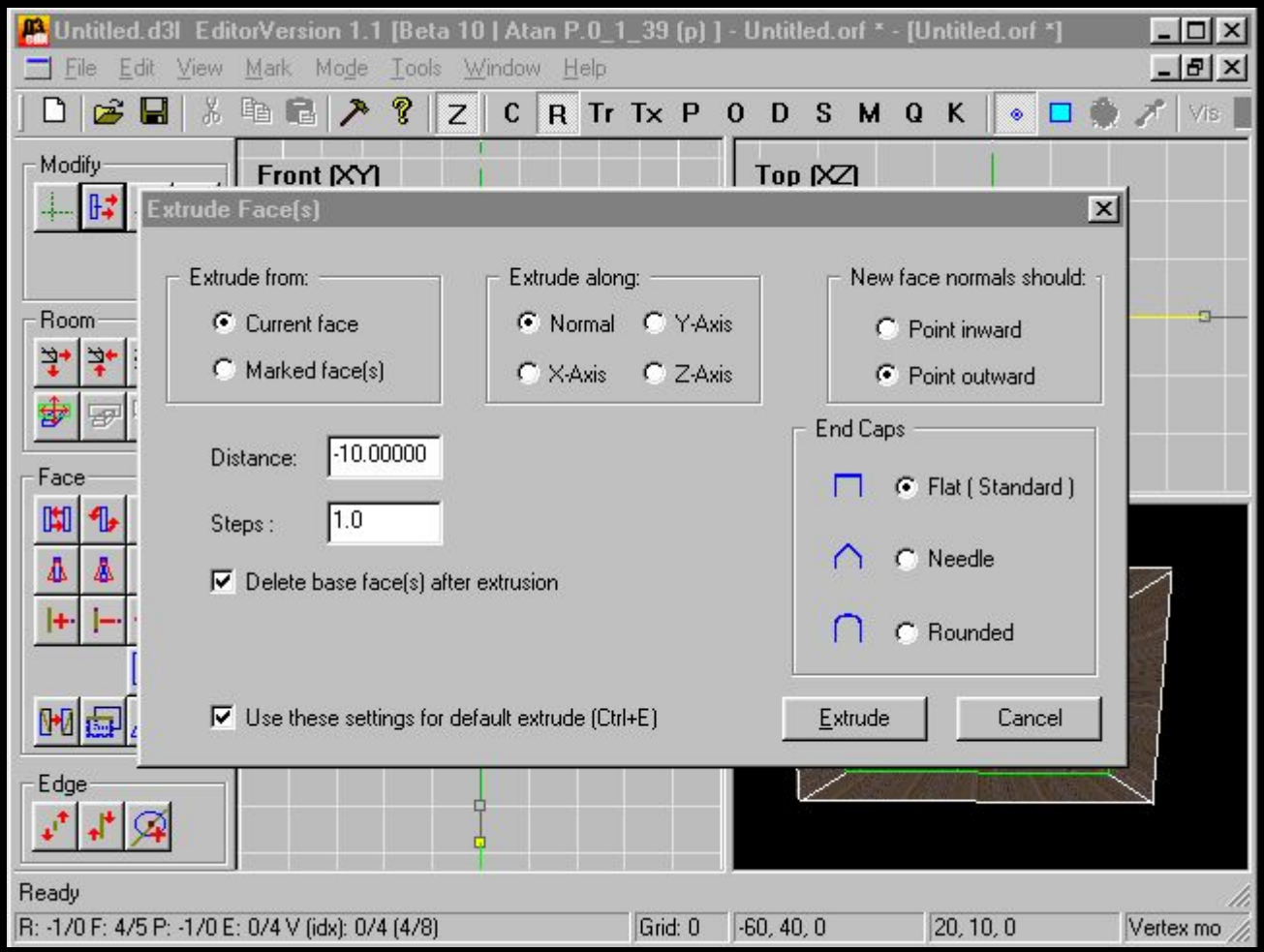


Ignoriere die Fehler fürs Erste...

Remove Extra Verts NICHT ausführen



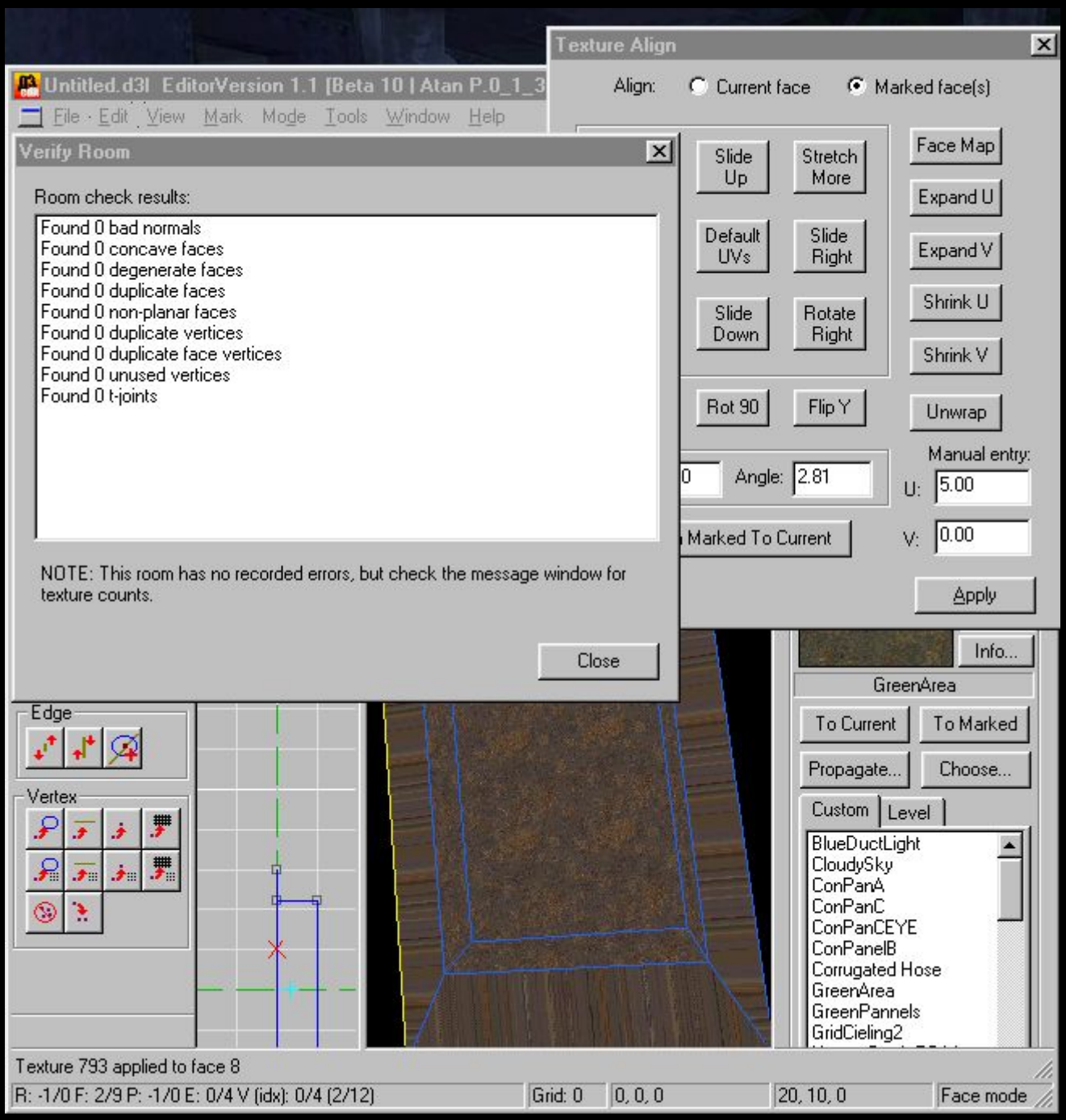
Mache das Face current und schrumpfe es. (in diesem Beispiel fünf mal). Wie Du sehen kannst, ist die Textur an vier Faces extrem gedehnt (wie sie es sollten, wenn man bedenkt daß sie von 0 gestreckt wurden) In v37 und früher wären drei der Faces gesplittet worden.



Nur um zu zeigen *warum* ich sowas wollen würde, habe ich das zentrale Face extrudiert...

...etwa so.

In v37 und früher würde es gut funktionieren, die Faces zu markieren und sie auf **Default UVs** zu setzen. In v38/2 und v39, (nicht mehr in der v40) warumauchimmer, ändern sie sich nicht. Die Texturen-U/V *hat* sich geändert, aber nicht so wie wir wollen würden. ich habe jeden denkbaren Weg versucht um die Faches zu den Defaults zurückzusetzen, und nur einen gefunden...



...Verify auszuführen.

Nun berichtet Verify keine Fehler mehr, und wichtiger noch...

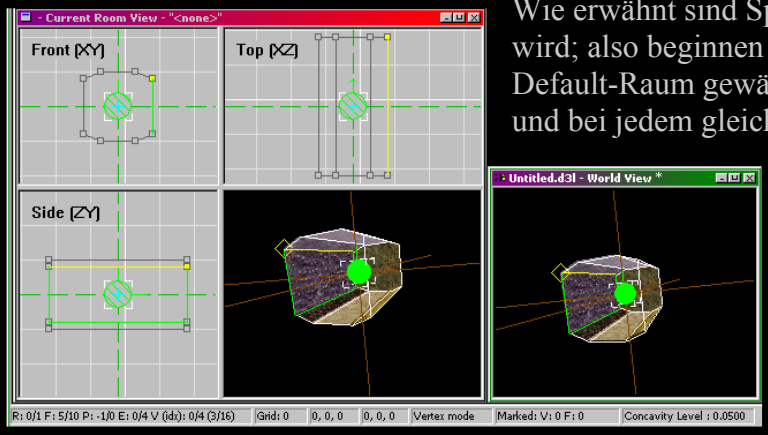
Ich kann nun die U/V auf Default setzen.

Beispiel im Spiel:

043 - Spieglein, Spieglein...

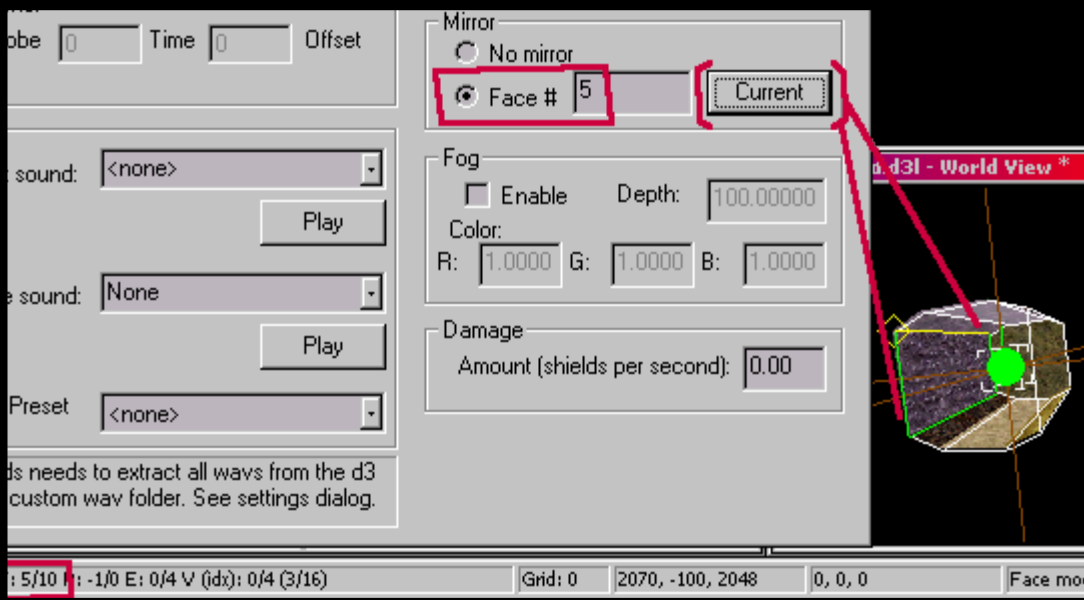
Ragil Ral

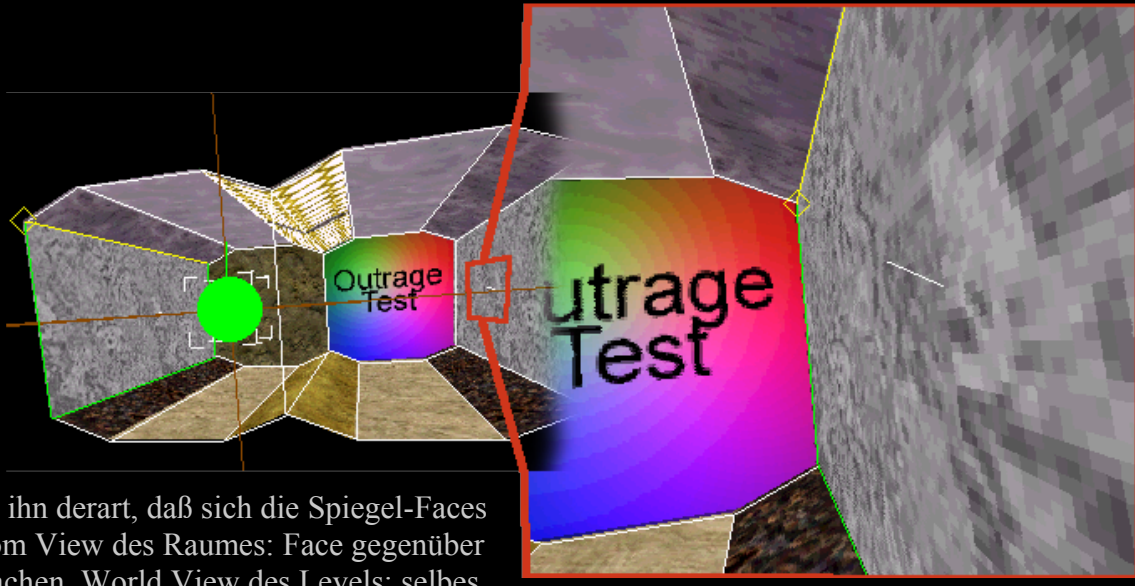
Eine der Neuerungen in D3 gegenüber seinen Vorgängern sind die Spiegel. Vorwiegend in Singleplayermissionen als gestalterisches Element verwendet, haben sie aber auch in Multiplayer so ihre Reize. Schon mal versehentlich in einen Spiegel geschossen anstatt...? ☺
Allerdings haben Spiegel gravierende Nachteile. Zum einen müssen ja die Reflexionen berechnet werden was auf Kosten der Leistung geht; aus diesen Gründen sind Spiegel in heutigen (2007/8) Multiplayer-Leveln verpöhnt. Zum anderen unterliegen Spiegel auch gewissen Beschränkungen, so darf z.B. nur ein Face pro Raum ein Spiegel sein. Leider, leider, leider sind auch keine mehrfachen Reflexionen zu realisieren, D3 berechnet nur die Erst-Reflexion und stellt ab dann die Spiegeltextur als Normal-Textur dar; Ihr werdet gleich sehen. Ist zwar unendlich schade, aber ich denke mir die D3-Engine würde sonst ausflippen, wenn man sie zwei Spiegel berechnen läßt die einander gegenüberstehen... aber hätte man das nicht auf 10 oder 20 begrenzen können? Naja kannma nix machen ☺ Eine weitere Einschränkung ist, daß maximal 31 gleichzeitig sichtbare Faces eine Spiegeltextur haben dürfen, ansonsten legt D3 einen standesgemäßen Crash hin. In der Außenwelt funktionieren Spiegel auch nicht richtig; teilweise spiegelt's, überwiegend wird das Face durchsichtig.



Wie erwähnt sind Spiegel eine Sache, die einem Raum zugeordnet wird; also beginnen wir mal mit einem Raum. Ich habe hier den Default-Raum gewählt, da er zur Demonstration voll ausreichend ist, und bei jedem gleich sein dürfte, der dieses Tut durchgeht und außerdem gehts schneller. Also File->New->New Level - Default Room und schon steht ein Level mit Spielerstartpunkt und allem. Gehe in die Room View, dann gegebenenfalls in den Face Mode wechseln (**Strg-F** oder **Tab**).

Hier greift eine weitere Einschränkung; denn Spiegel sind nur möglich, wenn die verwendete Textur das Flag 'ALPHA' hat, also als durchscheinend gekennzeichnet ist. Der Wert von Alpha bestimmt, wie stark die Textur spiegeln wird. Also gebt einem Face so eine Textur, ich hab hier die 'Dirty Glass' (unter Mine Textures) verwendet, die hat das Alpha-Flag. Da Spiegel eine Raumeigenschaft sind, gehe nach Window->Room Properties. Dort hast Du jetzt die Möglichkeit ein Face als Spiegel zu definieren. Ein Druck auf den Button **Current** trägt die # des current Faces ein, aber Du kannst natürlich auch ein anderes bestimmen. Es geht immer um die Facenummer *bezüglich des Raumes*, siehe Bild unten rechts. Definiert also eins als Spiegel und speichert den Level erstmal. Der Editor zeigt Euch allerdings nichts an, das Face wird einfach mit der Textur dargestellt, seit v40 bekommt man aber die Spiegeleigenschaft angezeigt, siehe weiter hinten. Oh, und Lichter nicht vergessen weils sonst finster ist ☺ Berechnet die also, packt den Level, hüpft in euren Pyro und schmeißt Euch vor den Spiegel.
Interessant finde ich, daß der Pyro in der Rückansicht nach vorn schaut wenn man einen Spiegel hinter sich hat.





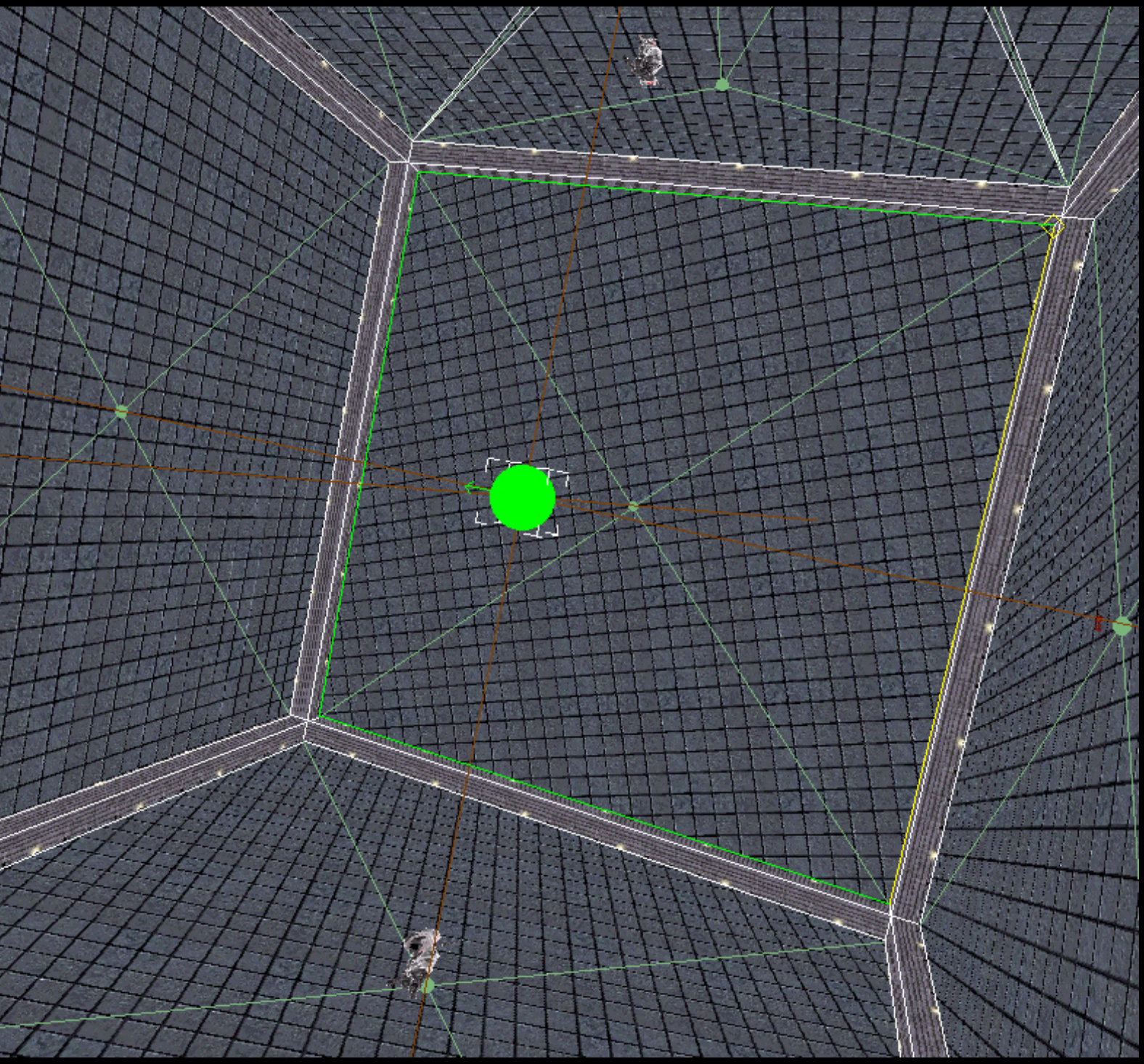
Nun der Beweis: Wir werden den Raum an diesen Raum anhängen, quasi spiegeln. Dazu speichert erstmal den Raum als separate Datei (in der Current Room View). Öffnet den Raum und platziert ihn derart, daß sich die Spiegel-Faces Gegenüber stehen (Room View des Raumes: Face gegenüber dem Spiegel current machen, World View des Levels: selbes Face current machen, Room->Place Room At Current Room, Room->Snap placed Room->Attach Room, wißt Ihr doch ☺ Dann müßt Ihr aber noch in den **Room Properties** des angefügten Raumes den Spiegel definieren, denn der wird nicht mit dem Raum mit abgespeichert.

Lichter berechnen, den Level `.mn3`-isieren, teschten, staunen. Der Screenie zeigt, daß beide Flächen spiegeln, aber eben nur einmal. Die Reflexion zeigt das hinter dem Pyro liegende Face, die 'Dirty Glass'-Textur, welche durchsichtig (alpha-Flag!) ist und uns deshalb ausm Level gucken läßt, wohingegen in der Rückansicht dieses Face normal spiegelt und das Face welches in der Hauptansicht spiegelt ebenfalls die Dirty Glass Textur bekommt. Man sieht also in der Vorder- und Rückansicht genau das Gleiche. Gespiegelt gewissermaßen.

Wenn man will daß eine größere Fläche spiegelt welche aus mehreren Faces besteht, geht folgendes: Zuerst sicherstellen daß alle planparallel sind. Dann für alle dieselbe Textur vergeben, und einem der Faces die Spiegel-Eigenschaft zuweisen. Es werden alle spiegeln, aber nur die Spiegelfläche wird opak sein; wenn Ihr also dahinter etwas baut wird das durch die anderen - ebenfalls spiegelnden - Faces sichtbar sein.
Sind die Polygone zur Spiegelfläche ein wenig geneigt, spiegelt's zwar trotzdem, aber es sieht etwas, naja, merkwürdig aus... versucht selbst.

Update für die v40

Mittlerweile bekommt man Spiegel angezeigt:

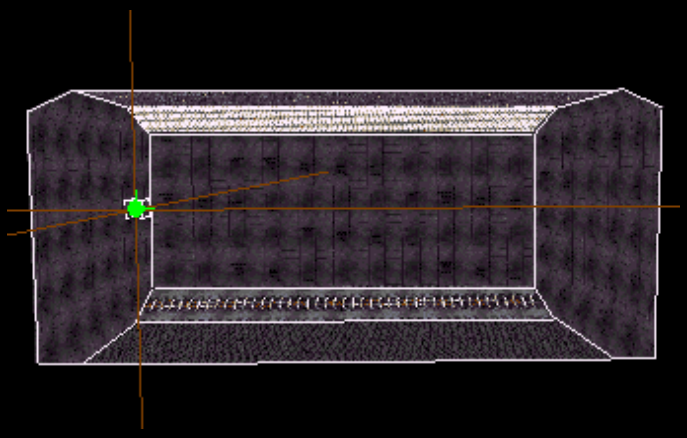


Ähnlich wie Trigger, nur mit der Farbe fahlgrün.

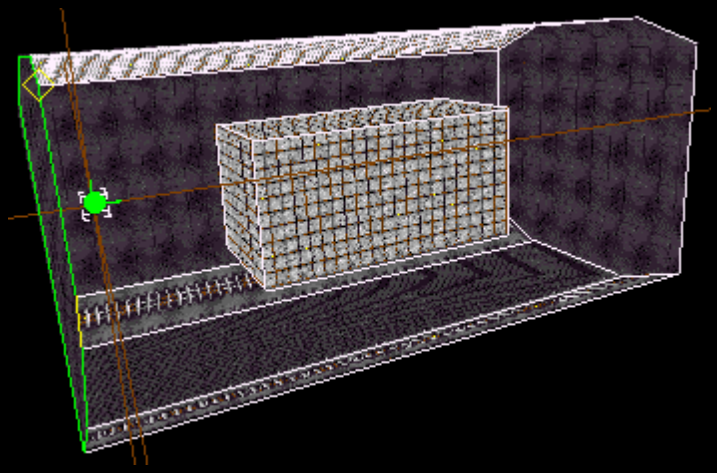
044 - Raum im Raum

Fischlein

Öffnet D3Edit und wählt File/New... "New Level - Default Room. Vergrößert den Raum mit Hilfe der **Expand Room** Funktion, ca. 35 mal auf die Schaltfläche. Der Raum hat nun eine Größe von etwa 100x100 Einheiten. Texturiert ihn nun nach Belieben.
Drückt nun die Tasten **Strg+G** um in den Objekt Modus zu wechseln, klickt den Grünen Player Startpunkt an und verschiebt (NumBlock: **2**, **4**, **6** und **8**) diesen an eine der Seiten im Raum.

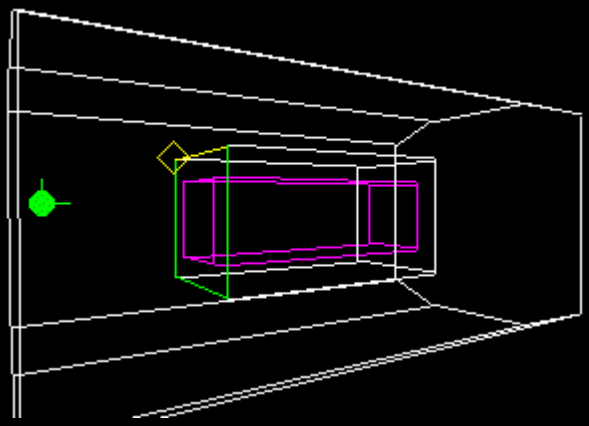


Nun wechselt in die Room View. Ich erstelle einen einfachen Raum mit acht Ecken in einer Grösse von L=100, B=50 und H=50, jeweils Einheiten. (erstellt ihn neben dem eigentlichen Raum) Wenn Ihr das gemacht habt markiert den Raum, in dem Ihr ihn mit der Maus auswählt (klicken und Ziehen, vorher in den Face Mode gehen). Jetzt verschiebt ihn in die Mitte des eigentlichen Raumes (NumBlock: **2**, **4**, **6** und **8**)



Ok, nun wählt im Menü File/New... New Room. Wir erstellen Ihn in einer Grösse von L=100, B=30 und H=30. Die Faces sollen diesmal nach innen zeigen. Speichert den Raum unter `temp.orf` und schließt ihn. Öffnet ihn nun wieder. Ich weiss nicht warum aber man muss ihn erst schliessen und dann wieder Öffnen damit es Funktioniert.

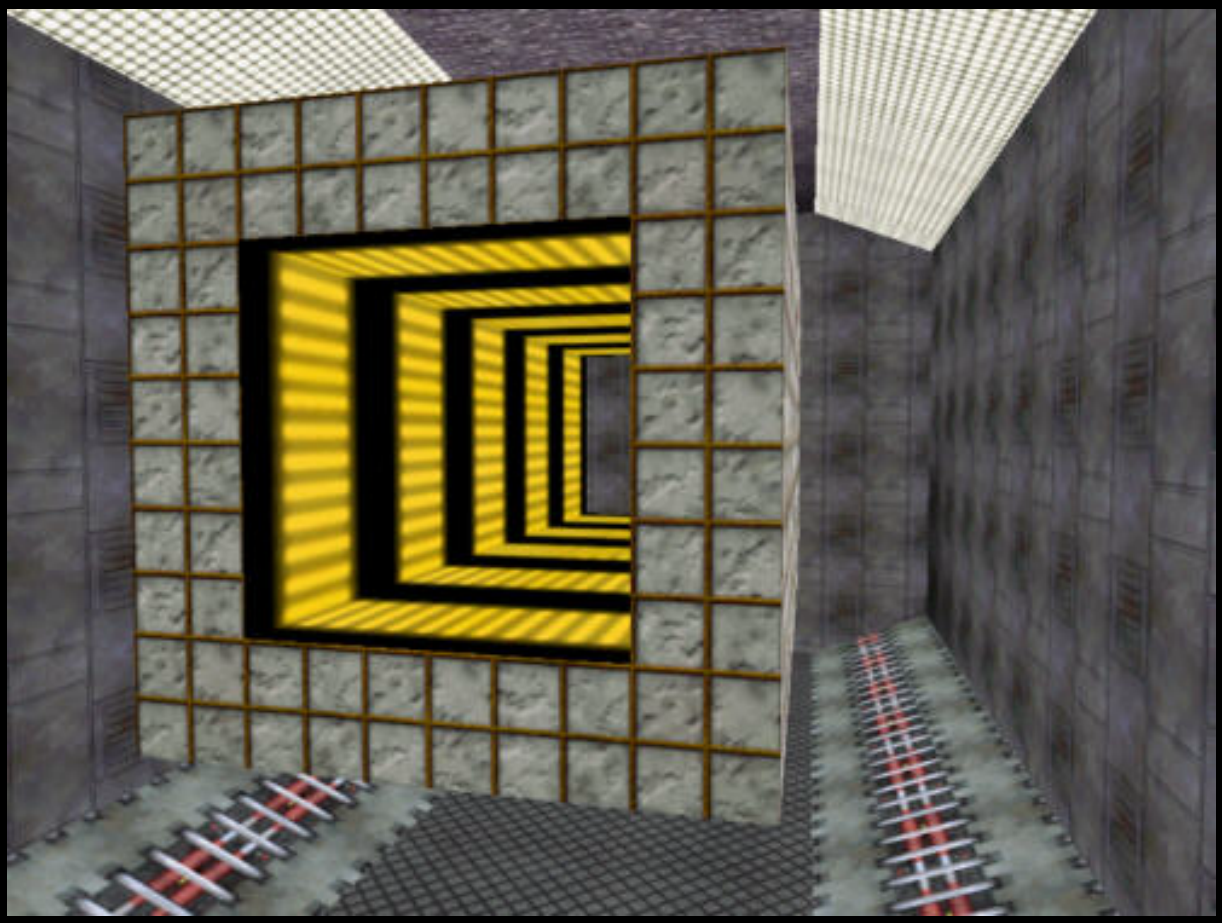
Wechselt jetzt in die World View und macht eins der Seiten-Faces zum Current Face. (im inneren Room) Öffnet das Room Menü und wählt Place Room at Current Room (Bild links). Nun wieder ins Room Menü und wählt Attach Room.



Klickt nun das gegenüberliegende Face im inneren Raum an und dann die Taste **M** um den Raum zu markieren. Jetzt klickt das gegenüberliegende Face des äusseren Raumes an, geht wieder in das Room Menü und wählt Join Room. In dem nachfolgenden Dialogfeld wird man gefragt ob man in die Mitte des Faces den Raum Zentrieren will, wir klicken auf Nein.

Klickt den inneren Raum an und wechselt in das Menü Windows, Room Properties und setzt ein Häkchen bei Refueling.

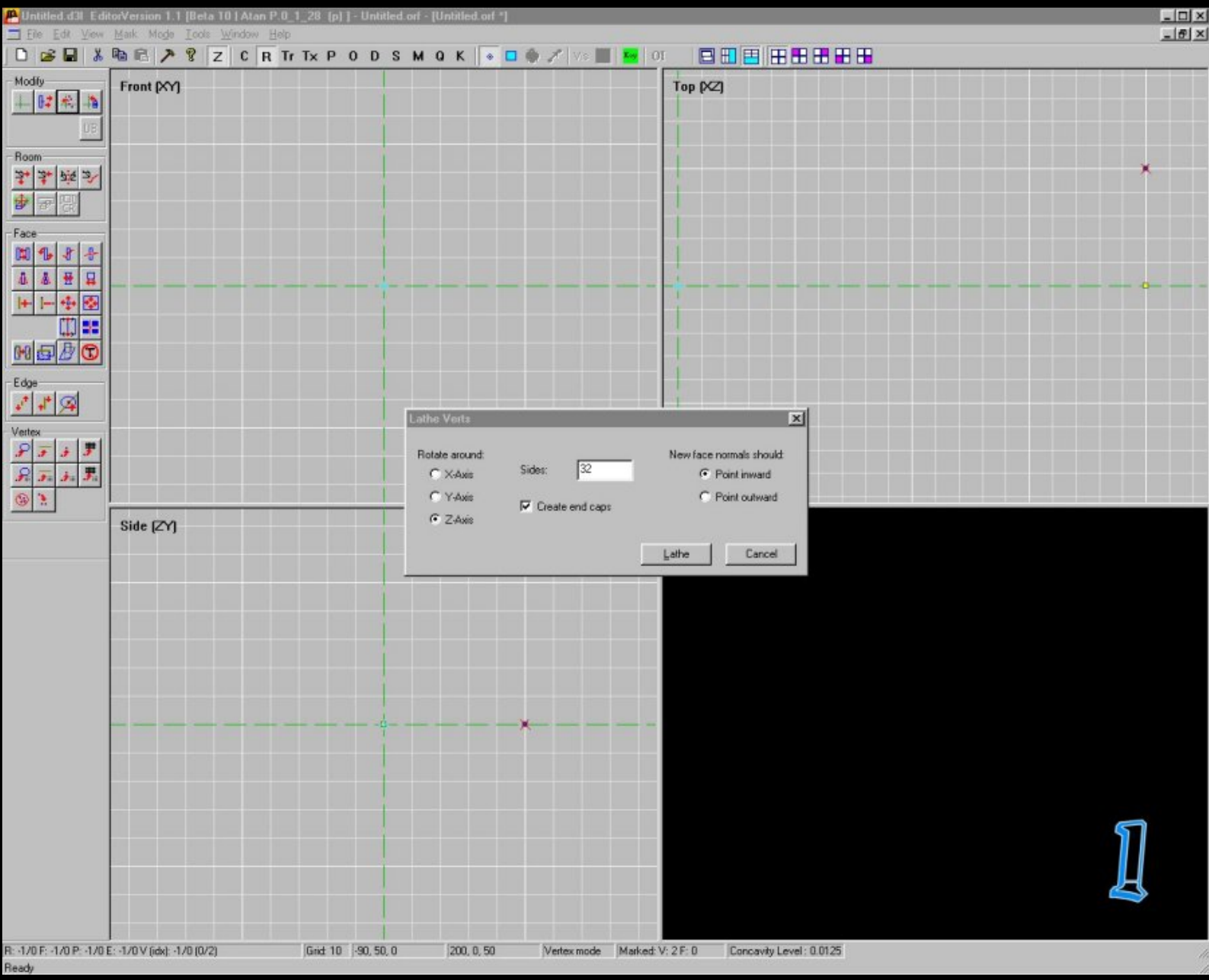
Jetzt habt Ihr in dem Raum einen Energietanker.



045 - Eine Sphäre machen

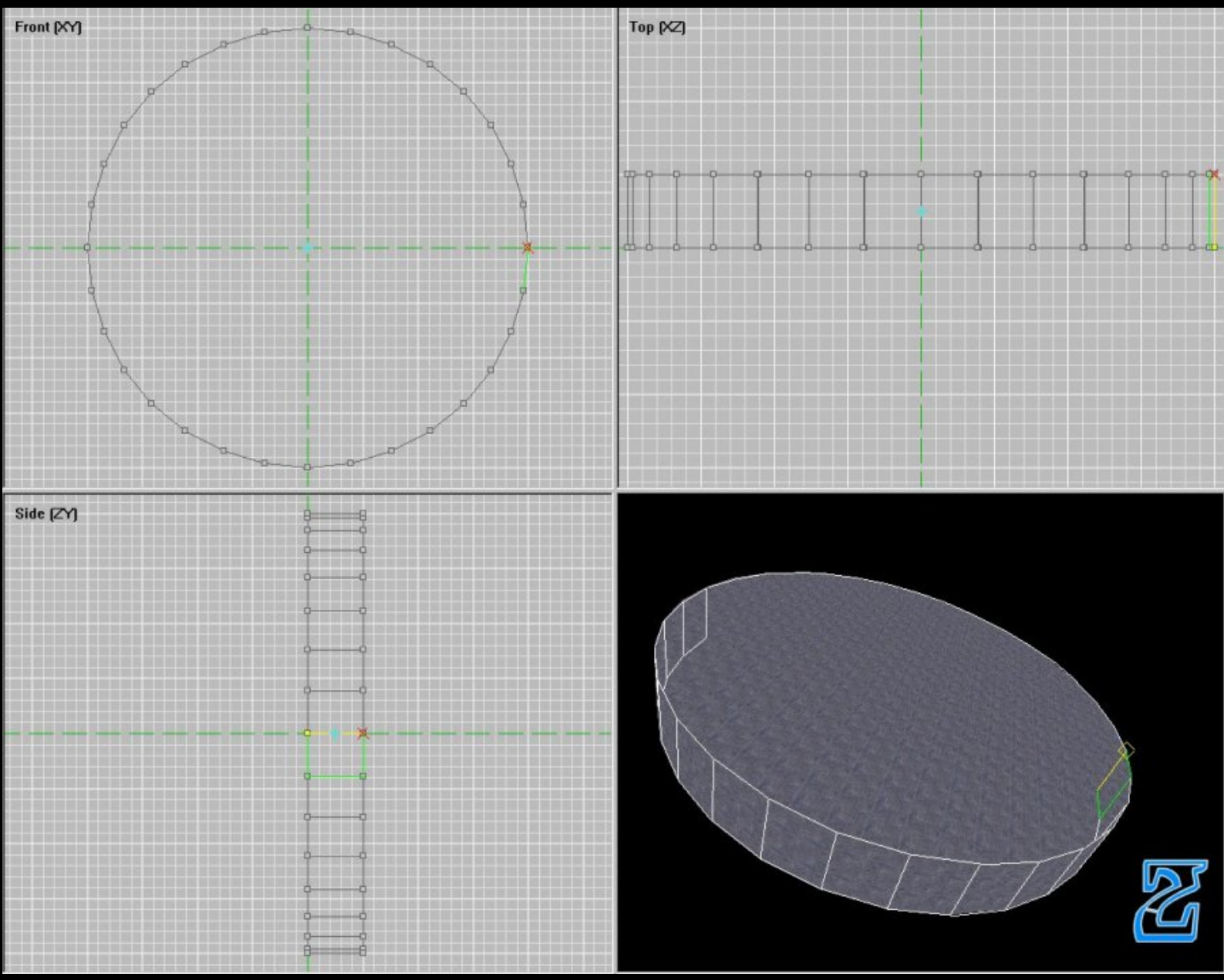
WillyP

Vertices platzieren

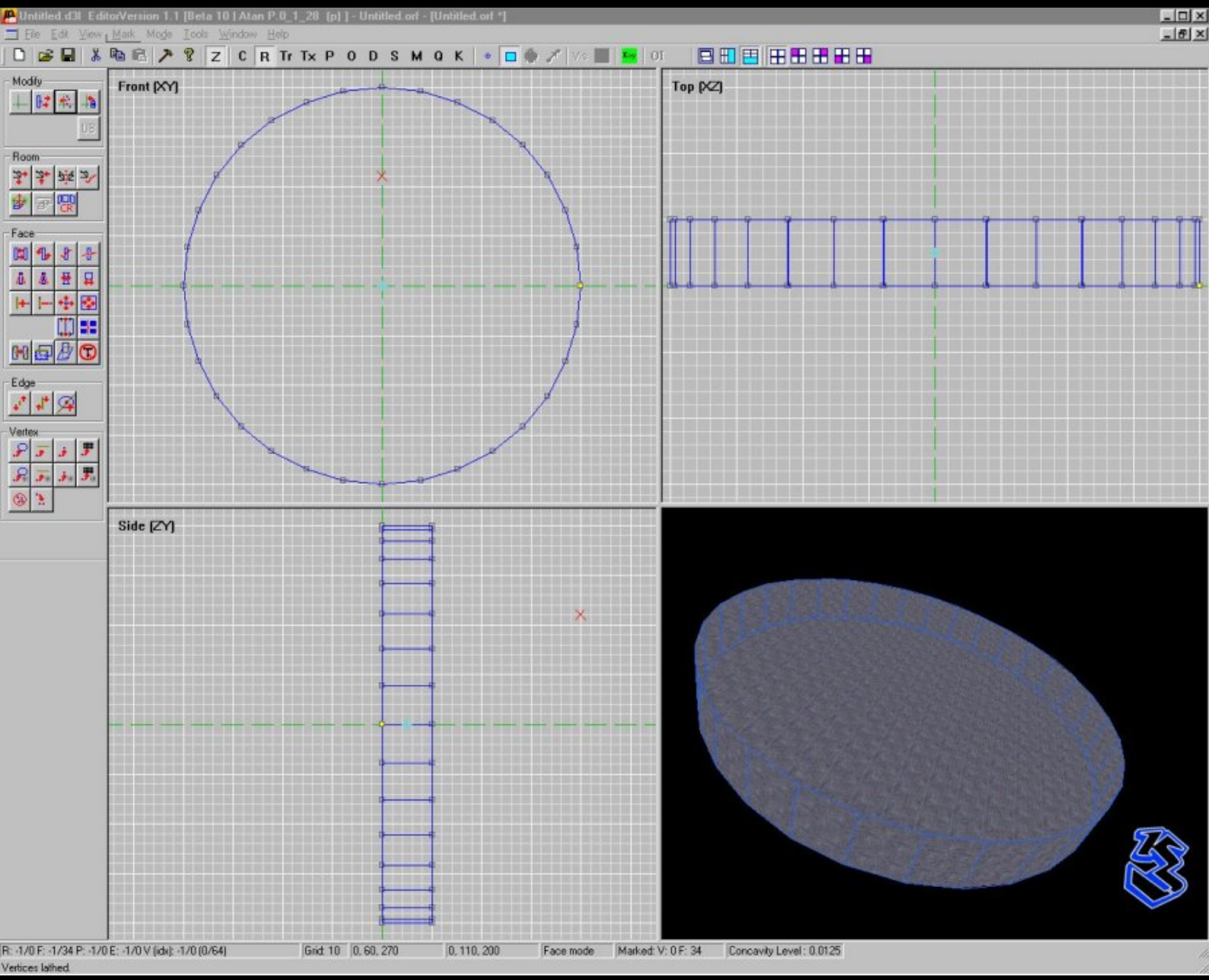


Lathe braucht mindestens zwei Verts, damit es ausgeführt werden kann.

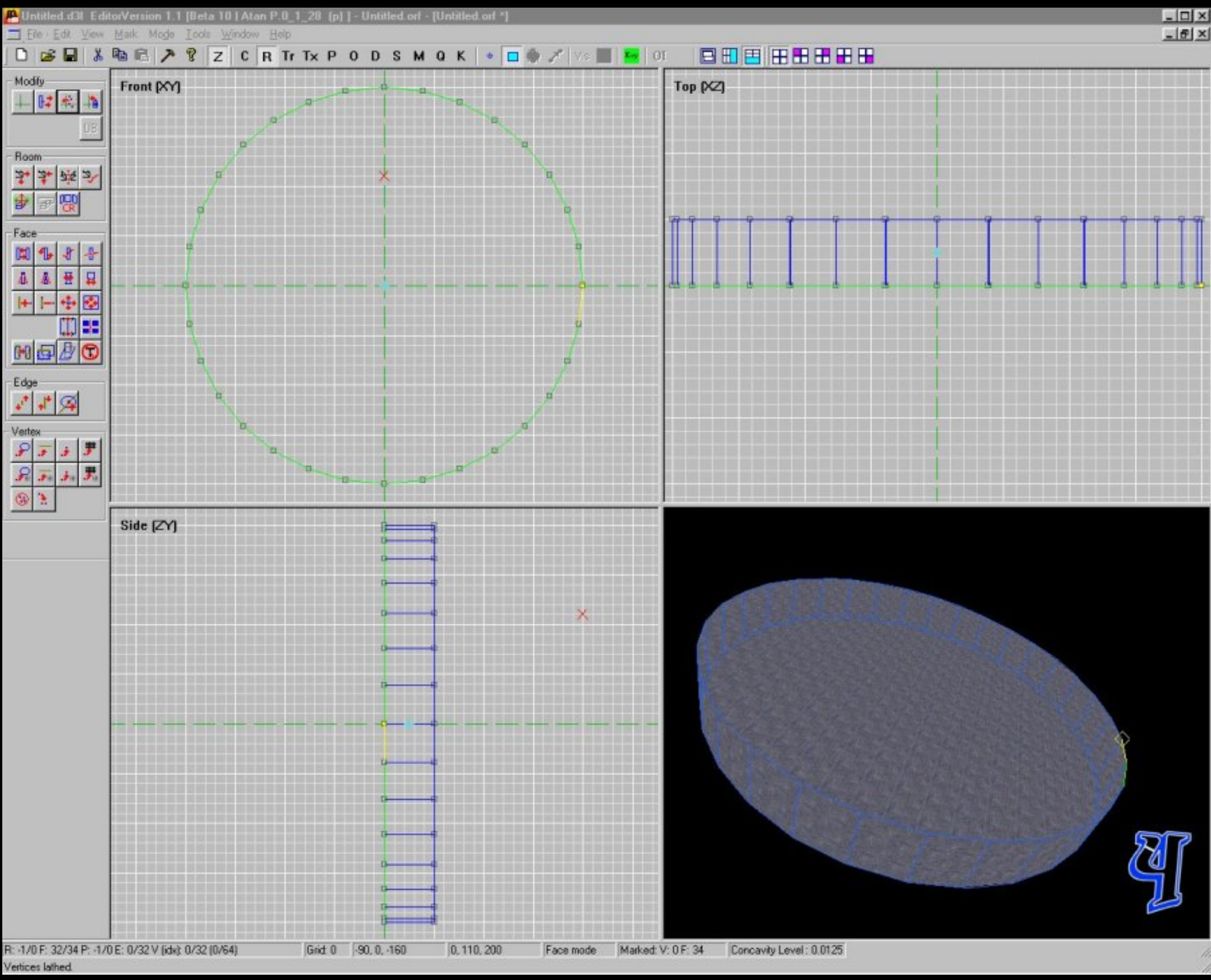
Zylinder lathen



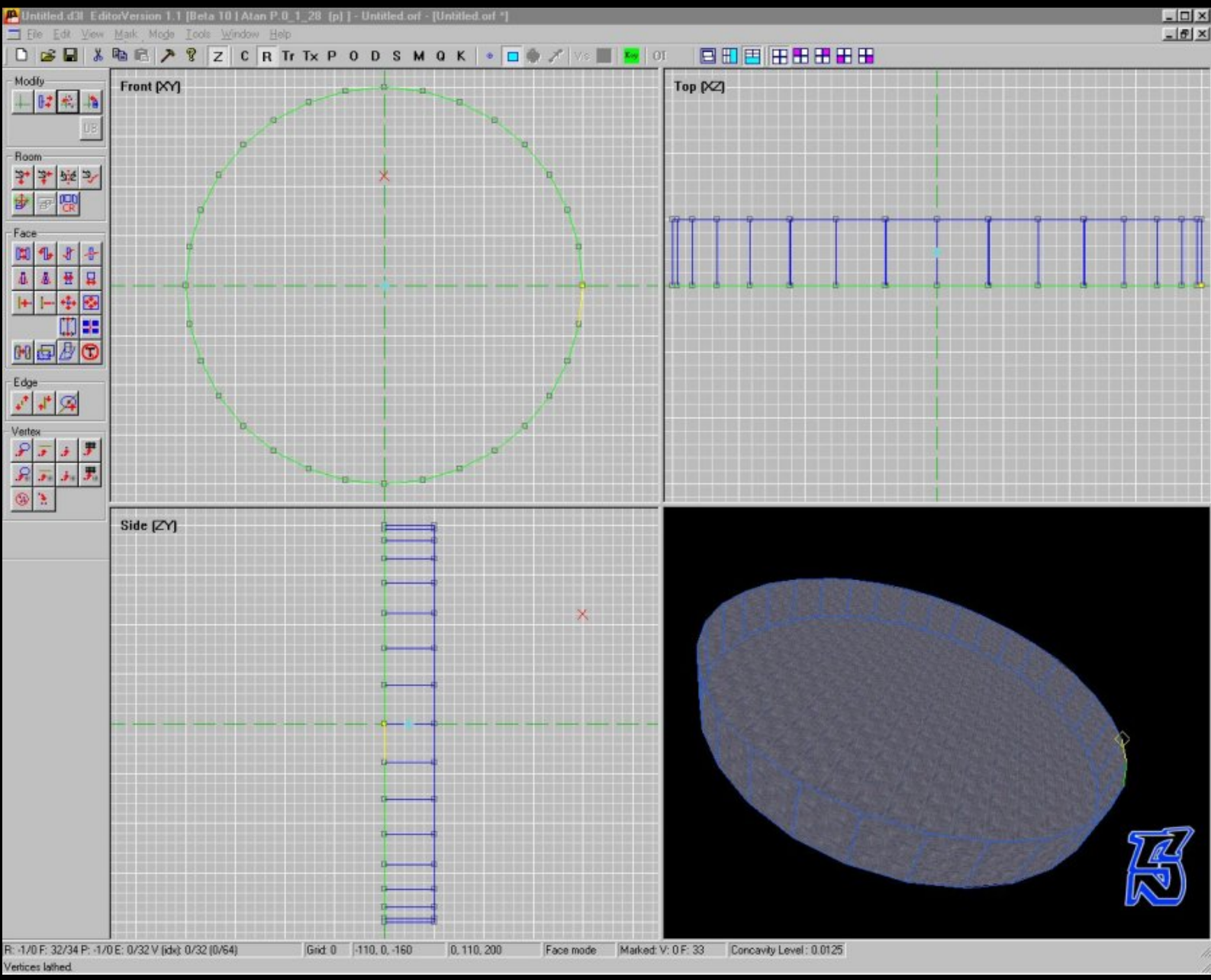
Alle Faces markieren



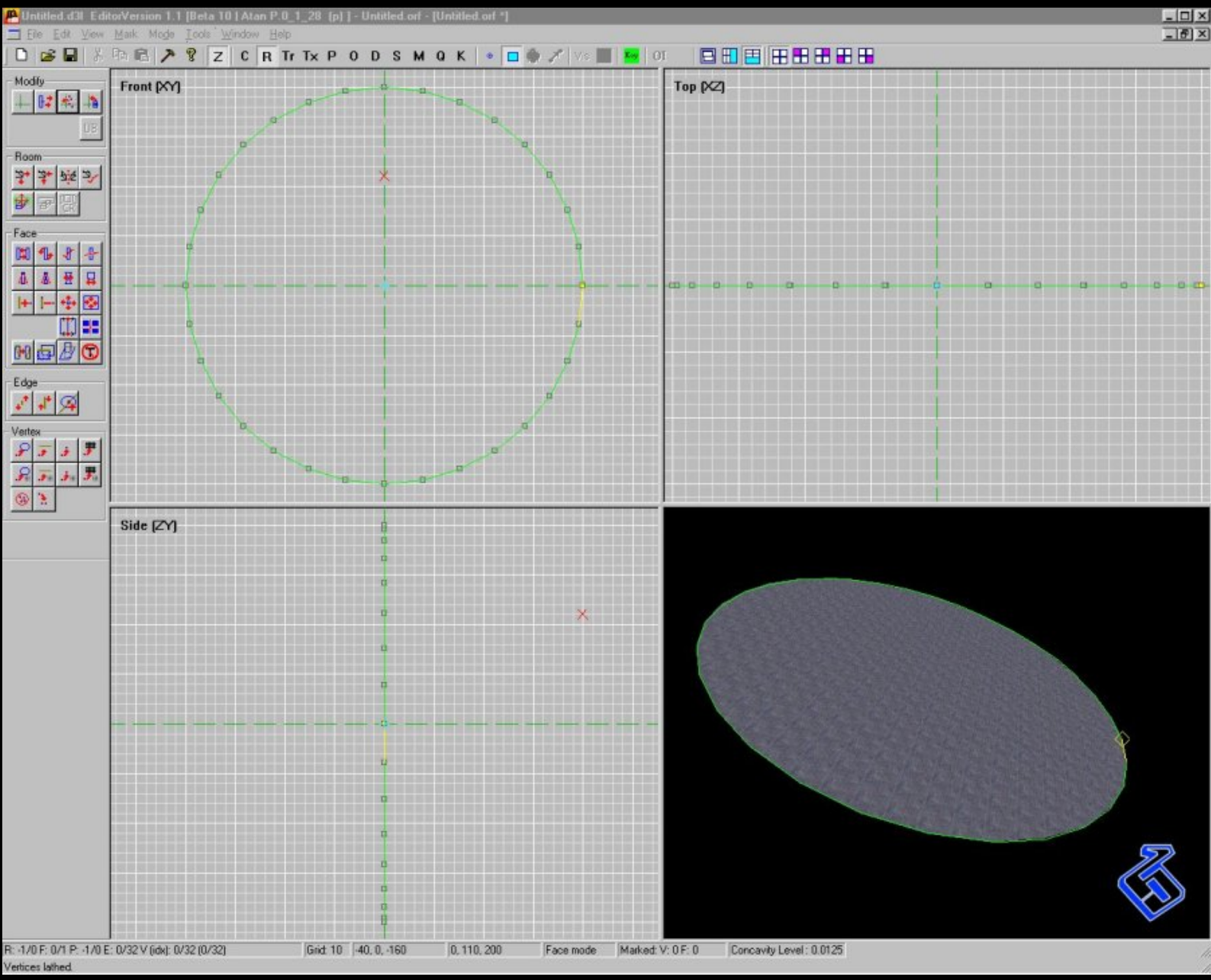
Face auswählen



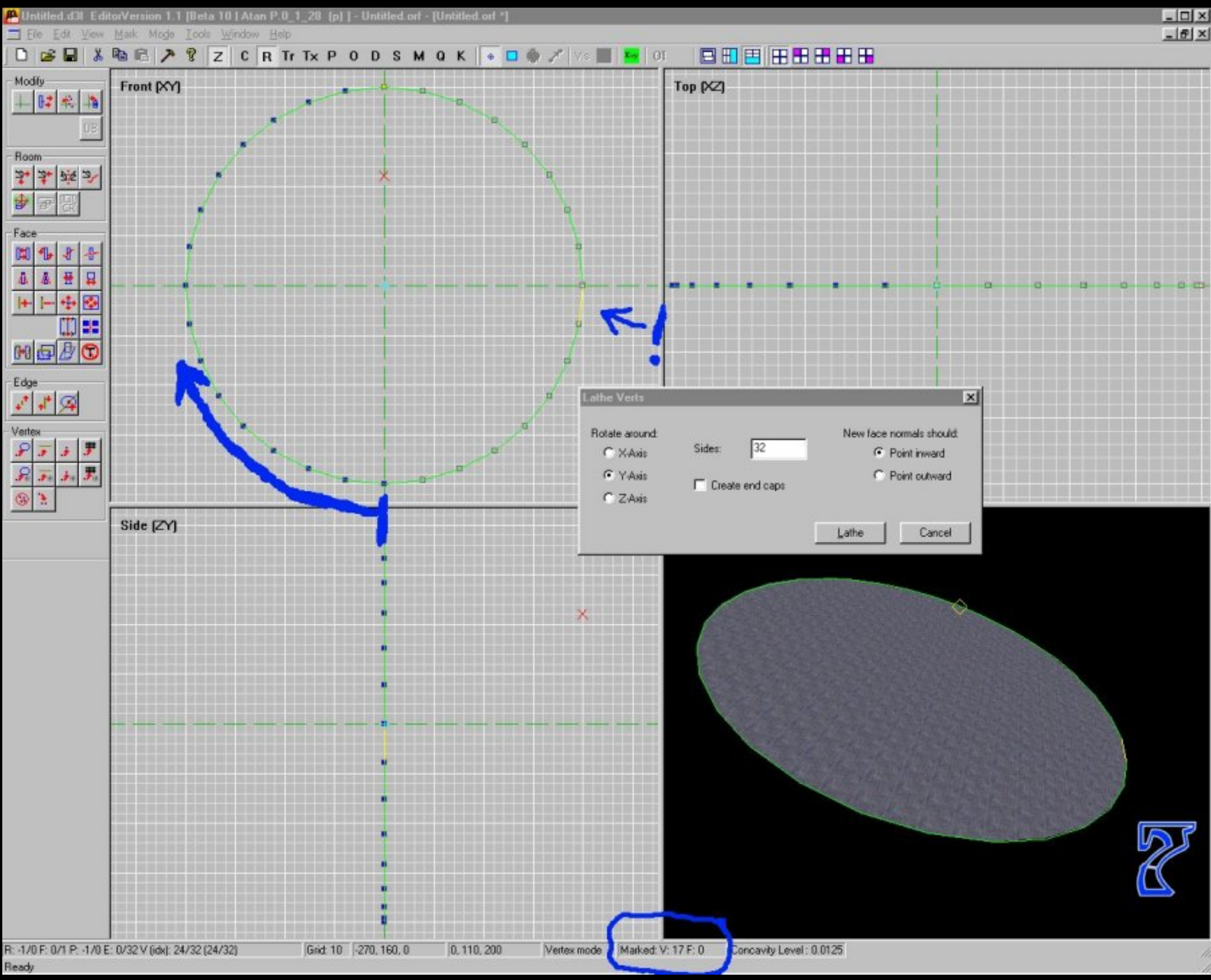
Current Face abmarkieren



Markierte Faces löschen

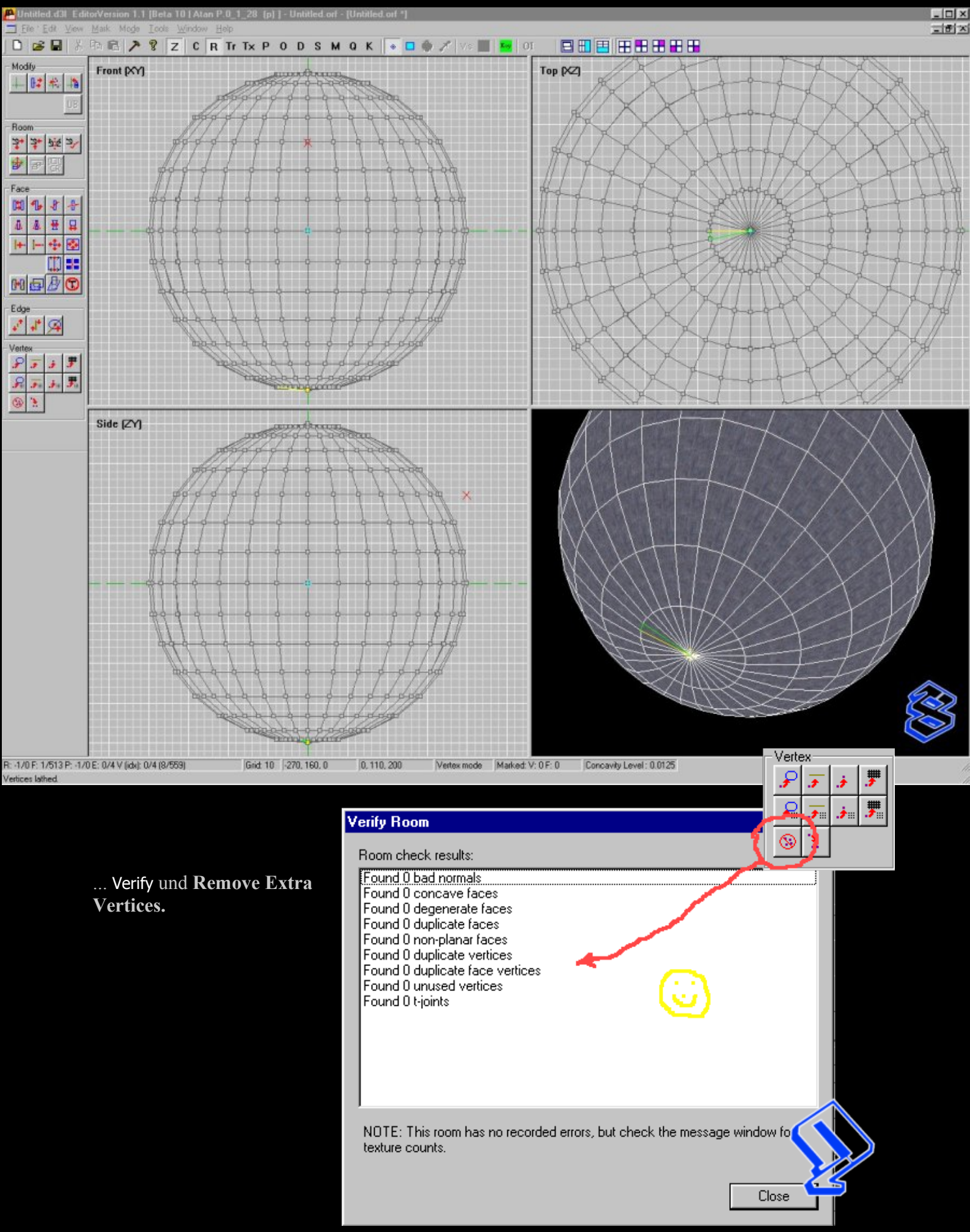


Vertices in Reihenfolge nacheinander markieren



Vertices Lathen...

... Verify und **Remove Extra Vertices.**



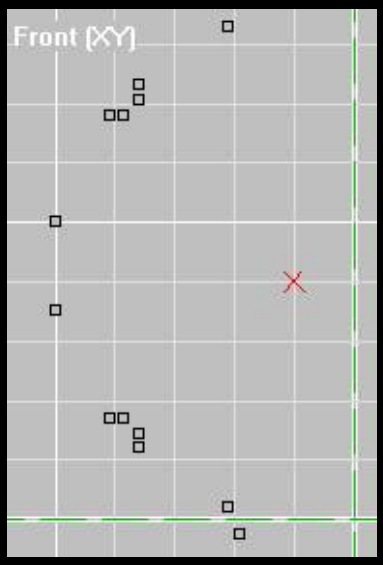
046 - Komplexität ganz einfach

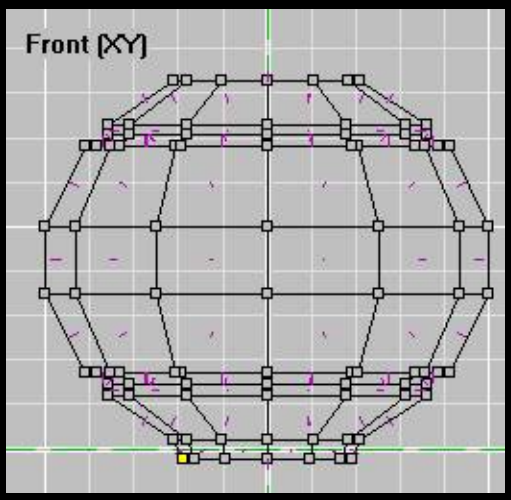
Papacat

In den nächsten Seiten gehe ich durch den Bau eines Raumes unter Verwendung von Lathe, Extrude, Rotieren und Kopieren / einfügen.
Du wirst diese Tools in Action und ein paar Techniken sehen, wie man sie anwendet um komplexe Strukturen einfach zu erstellen.
Ich werde einen Raum aus Vortex1 nachbauen, sodaß Du eine Idee hast was wir versuchen werden.

Die Idee dahinter ist das alte ‚das richtige Werkzeug für die richtige Aufgabe'-Konzept zu lernen.
Wie bei allem gib es mehr als einen Weg. Dies sind Techniken, die ich für effektiv befunden habe.

Wir werden den Vortex1-Raum mit den Fenstern an den Blicköffnungen an beiden Seiten.
Ich habe auch eine `.zip` inkludiert. Öffne `Sctut01.orf`, dort sind die Verts aufgelegt und bereit zum lathen.

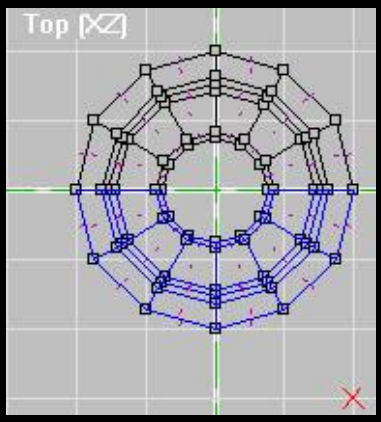




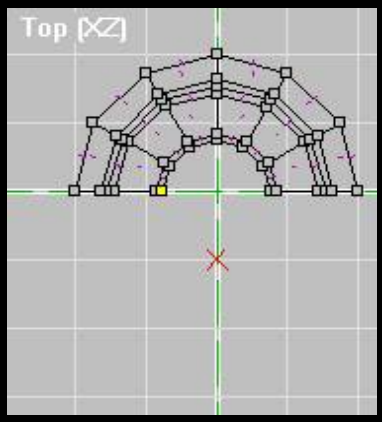
Zur Linken habe ich die Verts ausgelegt, die ich zur Sphäre lathen werde. Ich habe auch die Röhren eingearbeitet. Vergiß nicht, Verts immer fortlaufend einzufügen. Du kannst nicht Verts mitten in eine Kette stopfen; das Lathe wird ein Verhau werden. Ich ging von unten nach oben.
Markiere all die Verts, setze 12 Seiten, Normals pointing outward und lathe um die Y-Achse. Speichere Deinen Raum.
Vergiß nicht, es gibt kein Undo also speichere oft und manchmal unter einem neuen Namen.
Du kannst Extras immer löschen, wenn Du mit den Resultaten zufrieden Bist.

Nun werden wir Copy (**Strg-C**)und Paste on Top (**Strg-Shift-V**)verwenden um das in Viertel aufzuteilen. Als erstes mache die Top-Ansicht aktiv, dann setze das Grid auf 50. Gehe in den Face Mode.

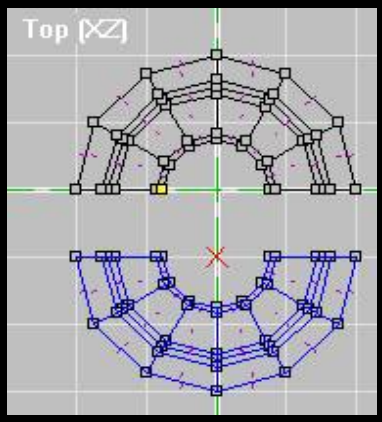
Als erstes markiere die gesamte untere Hälfte indem du einen Rahmen drüberziehst. Dann kopiere das.

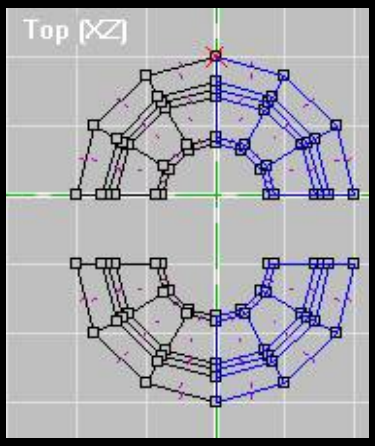


Markiere sie erneut und lösche sie.

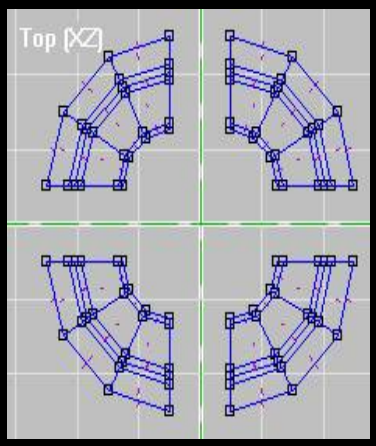


Dann verwende Paste on Top. Während der eingefügte Anteil noch markiert ist, drücke den Cursor-unten um ihn 50 Einheiten zu bewegen.



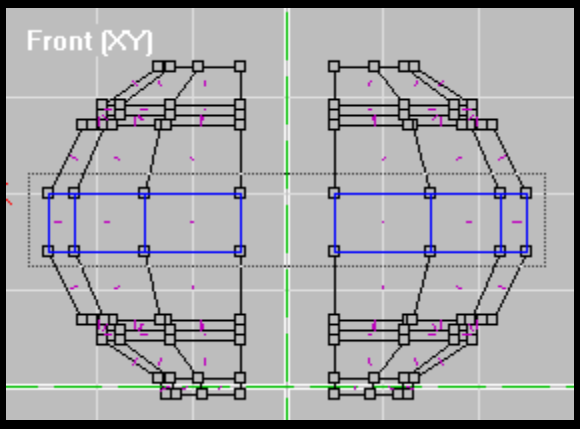


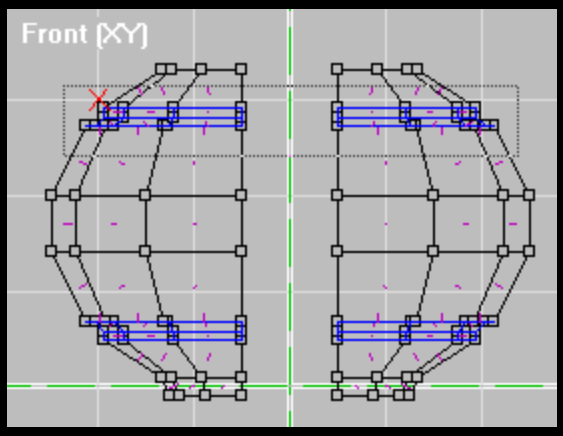
Das ist eine gute Technik,um Räume zu segmentieren. Mach jetzt dasselbe in der Vertikalen.



Markiere die eine Hälfte (links), dann kopiere sie. Erneut markieren, löschen, Paste on top und 50 nach rechts bewegen.
Ich möchte auf null zentriert sein, also setze das Grid auf 5, markiere alle (**M**) und bewege Sie 25 Einheiten nach links und oben. Es sollte so aussehen wie rechts.

Bevor wir zuweit kommen, laß uns ein paar Texturen draufklatschen. Während alle Faces markiert sind mache sie unter Verwendung von **To Marked** zu ‚LWwall'. Wer hier aufgepaßt hat, sieht daß ich bisher die Faces falschherum hatte. Ooooops... Markiere und flippe Deine Faces, wenn Dein Raum ebenfalls falschherum ist.

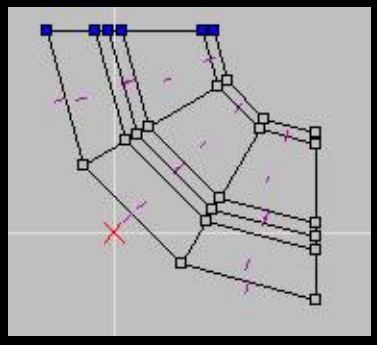




Demarkiere alles (**U**) und ziehe einen Rahmen um die Zentralfaces zu markieren. Mache sie zu ‚LWPend'. Demarkieren. Markiere die Faces wie links im unteren Bild. Mache sie zu ‚P-NovPipes1'. Die Textur war in meinen falsch herum. Überprüfe Deine, und falls nötig gehe nach **Align**, setze es auf Marked Faces und **Rot90**. Es sollte dann so wie rechts aussehen.
Wir werden die Texturen später ausrichten. Ich wollte nur was um ein wenig Tiefe zu bekommen.

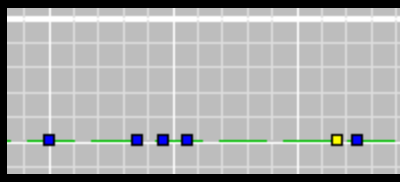
Nun werden wir die Metallrahmen für die Sphäre machen. Wir werden Copy, Paste on top und Extrude verwenden. Durch kopieren von Teilen eines Raumes und deren Verwendung als Template um andere Teile zu bauen habe wir es leichter wenn es Zeit ist, sie zusammenzubringen. Aktuelle Datei zu diesem Zeitpunkt ist `sctut02.orf`.

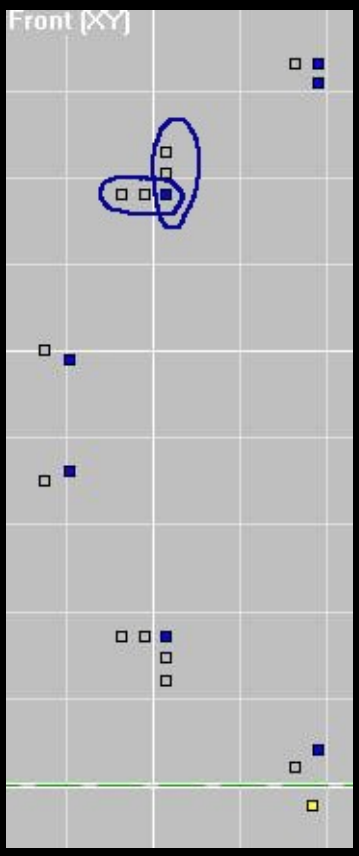
Als erstes mache die Top-Ansicht aktiv und ändere das Grid auf 100. Markiere die Verts wie rechts und kopiere sie. Wähle Paste on top, bewege die neuen Verts dann 100 Einheiten nach links Durch Verwendung von Paste on top weiß ich, daß der Teil genau zu seinem Platz kommt wenn er fertig ist. Ein schön großes Gitter bringt es einer Bewegung aus dem Weg um zu arbeiten.
Dann bewege Deine Top-Ansicht so daß die neuen Verts sichtbar sind.



Navigationstip: Drücke die **Shift**-Taste, klicke in die Top-Ansicht, halte den Mausbutton unten und ZIEHE das Fenster.

Hier rechts sind die neuen Verts. Wir werden jetzt ein paar Verts hinzufügen, um Faces zu machen also brauchen wir den Reference Frame auf der selben Ebene. Setze das Gitter auf 5 und bewege den Refernzrahmen auf die Linie indem Du **Ctrl** drückst während Du mit der Maus auf die Linie klickst.

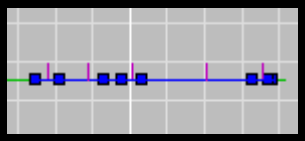




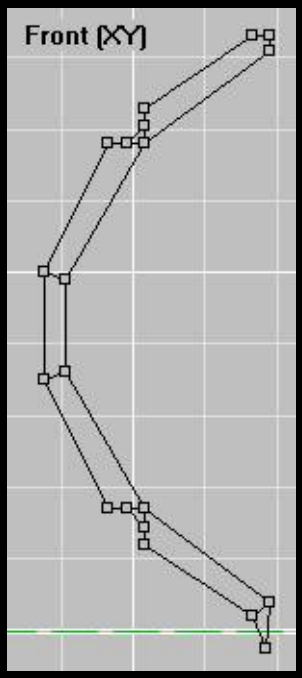
Mache die Frontansicht aktiv und bewege Dich zu den Verts (**Shift-Ziehen**).
Zur Linken siehst Du unmarkierte Verts. Da werden wir beginnen. Setze das Gitter auf 1 und setze die Verts die Du hier markiert siehst. Anstatt mit der Maus bewege Deinen Cursor (das X) mit **Ctrl-Cursor-Tasten**. Es ist dadurch einfacher akkurat zu sein.

Ich habe die Verts links eingekreist, um Dich zu warnen. Stelle sicher daß sie in einer geraden Linie sind, sodaß Du keine konkaven Faces kriegst.

Demarkiere alle Verts bevor Du beginnst Faces einzufügen. Ich empfehle, nicht mehr als vier auf einmal zu markieren. Dadurch kannst Du **Twist Face** verwenden, wenn das Face verdreht ist anstatt **Swap Verts**. Dann verbinde alle Faces die Du verbinden mußt.
Beobachte auch die Top-Ansicht um sicherzustellen daß die Normals alle in die selbe Richtung zeigen - nach oben.

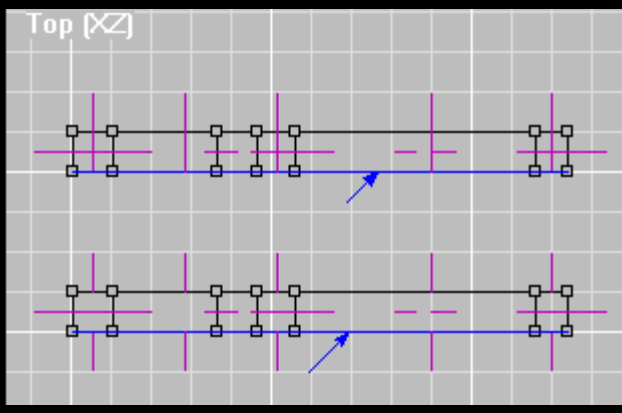


Wenn Du fertig bist, sollte es wie das Beispiel rechts aussehen.



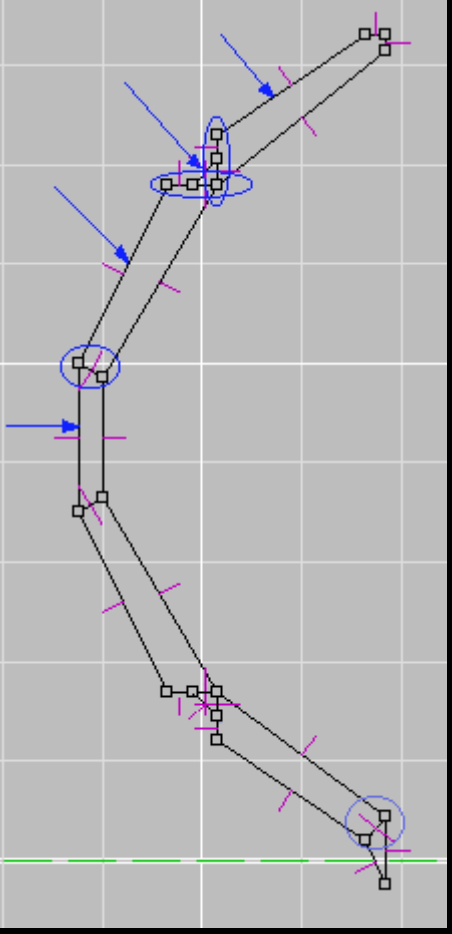
Zeit zu EXTRUDIEREN. Bevor wir das tun, behalte im Kopf daß alle neuen Faces gleich texturiert werden wie das/die Eltern-Face/s. Markiere die Faces und ändere sie in LWFlor.

Du hast sichergestellt, daß alle Normals in die selbe Richtung schauen – nach oben. Markiere alle neuen Faces, wähle **Extrude** und setze es auf: Extrude from: Marked face(s), Extrude along: Normals, 5 Einheiten, Point outward. Es ist wichtig, daß du sicherstellst daß Delete base faces NICHT angetickt ist. dann Extrudiere.
In der Top-Ansicht sollte Dein Metallrahmen wie meiner hier rechts aussehen. Wie Du siehst, schauen die Eltern-Faces noc immer nach oben. Sie sind noch immer vom Extrudieren markiert, also wähle das Flip-Tool sodaß sie nach unten schauen.



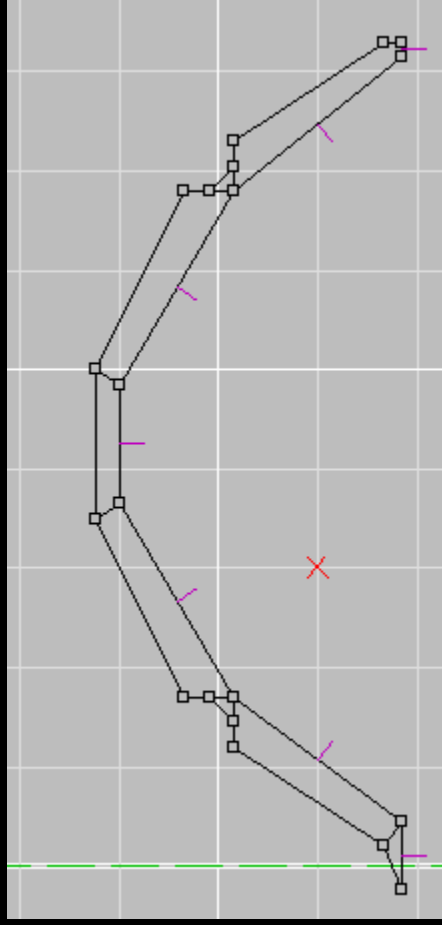
Eine Multiface-Extrusion zu machen macht zwei Dinge auf die Du aufpassen mußt. Eines sind Extra Verts das andere sind doppelte Faces. Das ist, weil D3Edit noch immer jedes Face individuell behandelt, wenn du extudierst. Das bedeutet, daß eine für eine Naht zwischen den Original-Faces für jeweils beide ein Seiten-Face erzeugt wird. Wenn die Original-Faces miteinander auf einer Ebene waren, werden die zwei neuen Seiten-faces auch auf der selben Ebene sein aber dort ihre individuellen Verts haben.

Also bevor Du weitermachst: **Remove Extra Verts.**



Zur Linken habe ich ein paar Beispiele von doppelten Faces eingekringelt, welche während des Multiplen Extrudierens entstanden. Markiere und lösche diese.

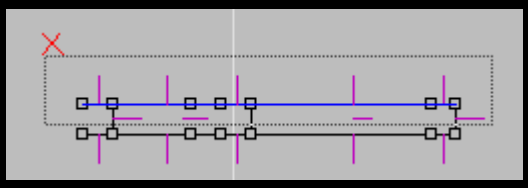
Die Pfeile zeigen zu einigen Faces, die außerhalb der Shell sein werden. Lösche alle diese Faces. Wenn Du fertig bist, sollte das Teil so wie rechts aussehen. Speichern!



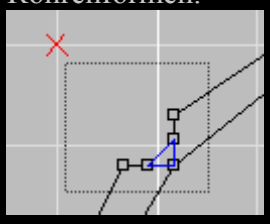
Ein Rahmen ist fertig. Nun zum Raum zwischen den Rahmen. Wir werden ihn so machen daß dazwischen keine Röhren sind. Wir werden kopieren was die Innenseiten-Faces der Rahmen wird, ändern wo die Röhren hinkommen, extrudieren und dann die Faces löschen die wir nicht brauchen. Es mag scheinen, daß wir hier nicht besonders schnell weiterkommen, aber wenn wir erst einmal fertig sind, Werden die Dinger schnell und einfach zueinanderkommen.

Der aktuelle Stand ist `sctut03.orf`.

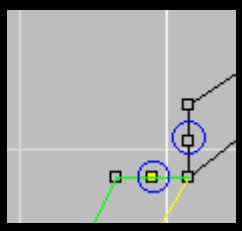
In der Top-Ansicht ändere das Grid auf 100, markiere die Faces wie rechts gezeigt. Kopieren, Paste on top, und 100 Units nach links bewegen.



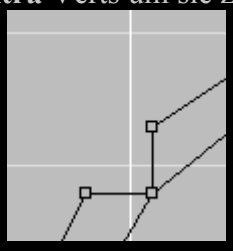
Nun werden wir die Röhrenform beseitigen. Gehe in die Frontansicht und lösche die kleinen Faces von den beiden oberen und unteren Röhrenformen.



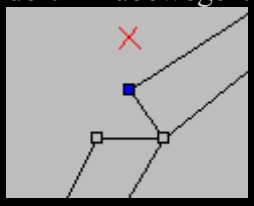
Jetzt entferne die Verts aus beiden Bereichen wie gezeigt..



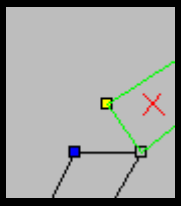
Vergiß nicht, daß bei Anwendung von **Remove Vert from Current face** das Vert immer noch da ist, es nur aus der Edge genommen wird. Verwende **Remove Extra** Verts um sie zu löschen..



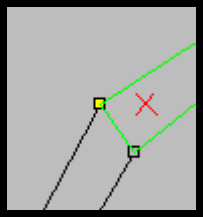
Markiere eines der hinteren Verts und bewege es auf halbem Wege zum gegenüberliegenden wie unten gezeigt. Da alles noch immer am Grid ist, wäre es einfach nur den anderen Vert dorthinzubewegen.



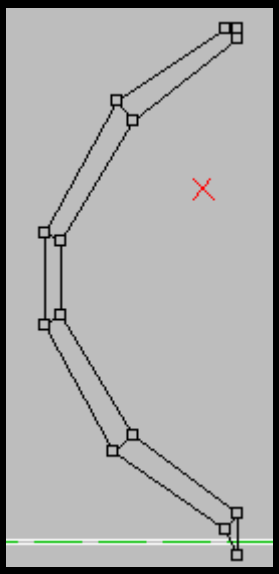
Hier lernst Du das **Snap Marked Verts to Current Vert**-Tool zu benutzen. Mache das andere Face und das Ziel-Vertex current.



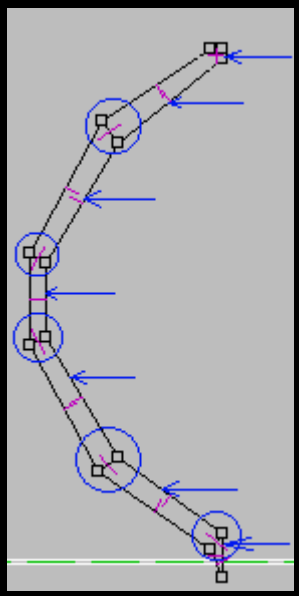
Dann wähle (**Snap Marked Verts to Current**) und erzeuge ein Resultat wie unten. Das Tool bewegt das Vert, aber Du mußt immer noch Remove Extra Verts machen, sodaß die Kanten verbunden werden. Mach das bei beiden Stellen.



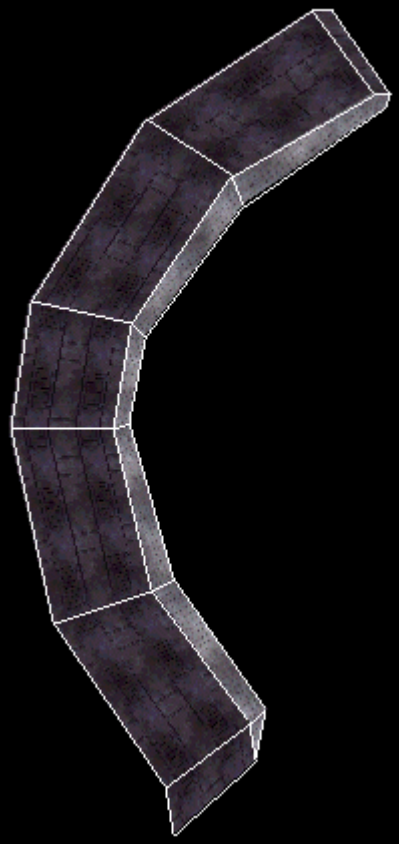
Die neue Face-Gruppe sollte so wie unten aussehen. Markiere sie und extrudiere (in der Top-View) 40 Einheiten nach oben, Normals pointing inward.



Markiere und lösche die Faces die ich eingekringelt und mit Pfeilen markiert habe. Markiere die Wand-Faces und texturiere sie mit LWwall.



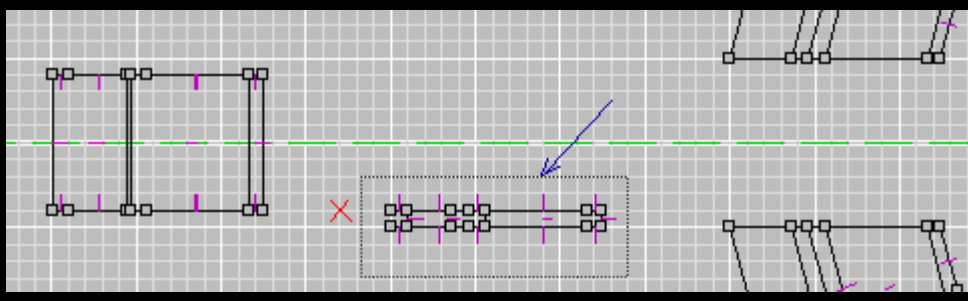
So sollte es dann etwa aussehen:

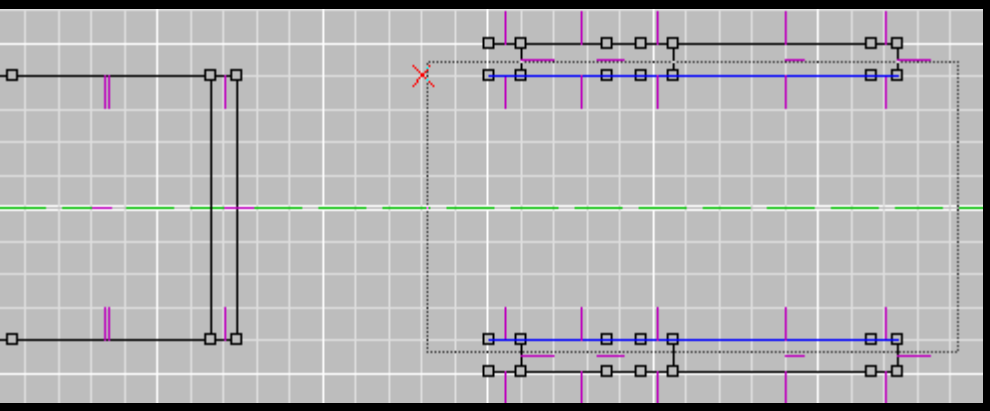


Der aktuelle Stand ist jetzt `sctut04.orf`.

Rechts siehst Du, wo wir jetzt sind

Nun markiere die Faces des Rahmenteils wie rechts gezeigt und kopiere das.





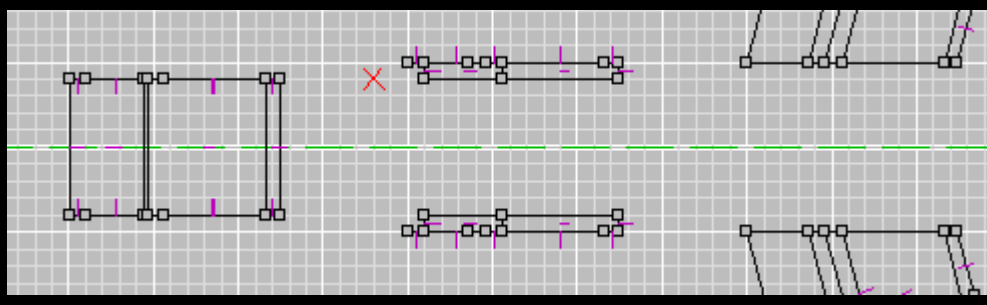
Paste on top (**Ctrl-Shift-V**), dann um 45 Einheiten nach oben bewegen.

Alles demarkieren.

Markiere die inneren Faces und lösche sie (links)

Dein Raum sollte jetzt so aussehen (rechts)

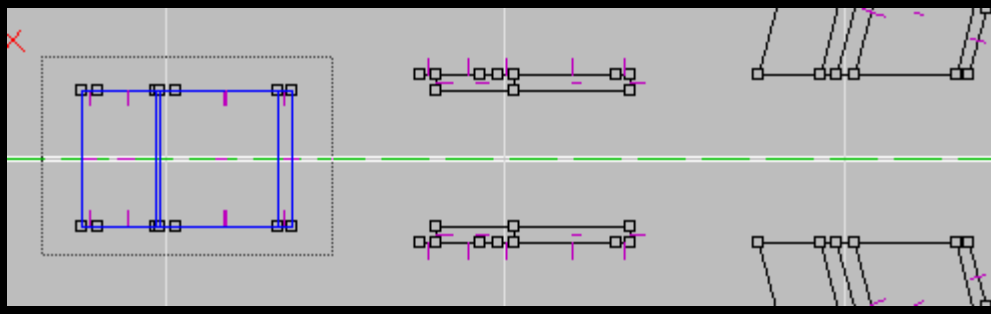
Nun sind wir bereit, das alles zusammenzusetzen.

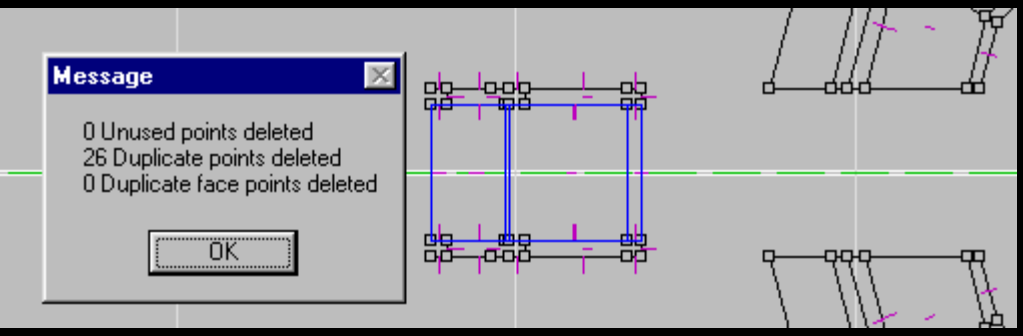


Der aktuelle Stand ist jetzt `Sctut05.orf`.

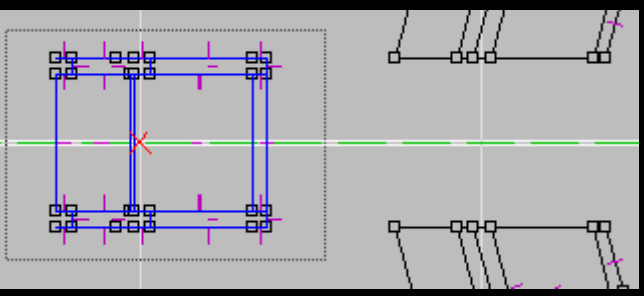
Ein Weg sicherzustellen, daß die Teile richtig zusammenkommen ist auf doppelte Verts zu checken. Wenn sie sich treffen bekommst Du Duplikate. Wenn nicht, dann nicht. Drücke Remove Extra Vertices, so wissen wir, daß wir sauber starten.

Als erstes aktiviere die Top-Ansicht und setze das Grid auf 100.
Dann markiere den rechts gezeigten Teil an Faces und bewege sie 100 nach rechts (eine Bewegung).

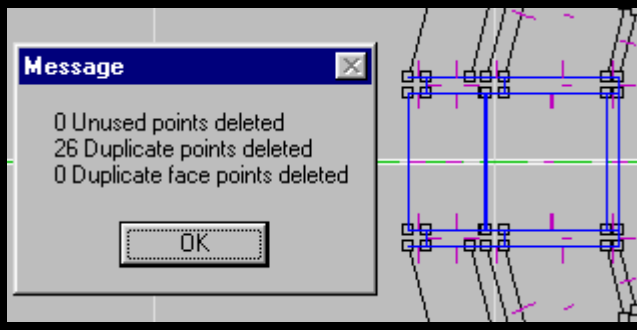




Dann **Remove extra Verts**. Wie Du links sehen kannst, haben sich alle Verts getroffen. Durch Verwendung von Kopieren in den vorigen Schritten haben wir sichergestellt, daß die Verts exakt gleich ausgelegt sind. Durch die Verwendung von Paste on Top und anschließendem Bewegen haben wir sichergestellt, daß alles ausgerichtet bleibt und einfach zusammengeht.

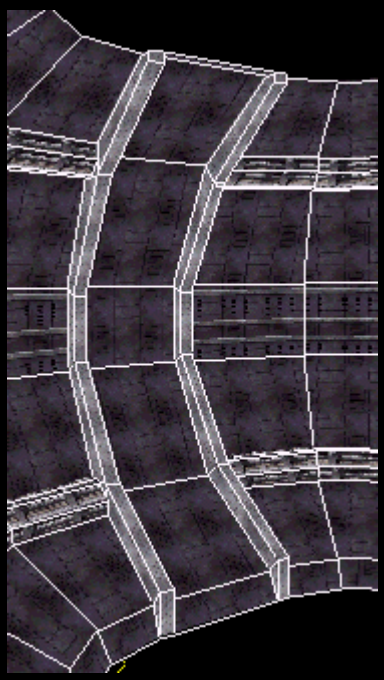


Markiere den Teil an Faces wie links gezeigt und bewege sie 100 Einheiten nach rechts (eine Bewegung). Dann klicke **Remove extra Verts**.



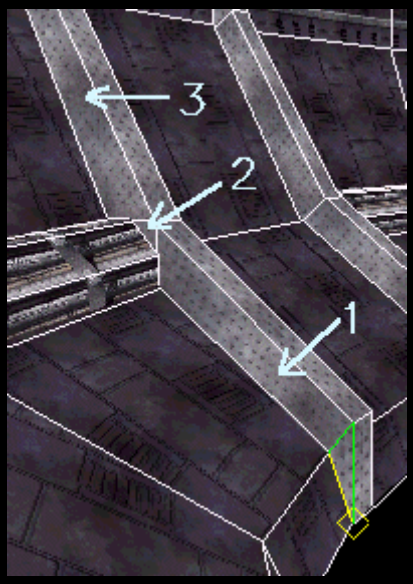
Zur Rechten sehen wir das Resultat bis jetzt. Sieht gut aus. Nun laß uns die Texturen ausrichten. Wir wollen nicht Sachen kopieren und dann viermal soviel auszurichten haben.

Während der gesamte neue Teil markiert ist, wie in den Bildern der 2d-Ansichten oben gezeigt, gehe zu Align in der Texture-Bar, setze es auf marked faces und wähle Default UVs. Jetzt sind wir bereit, Texturen auszurichten.



Der schnellste, einfachste Weg Texturen auszurichten ist die Control-Klick-Methode. Das wird im 3d-Fenster gemacht, daß auf Texture with outline eingestellt ist. Fange damit an, daß Du ein Face current machst wie rechts gezeigt. Dann halte die Ctrl nieder und klicke ein Face, das eine Kante mit dem current gemeinsam hat. Dieses Face richtet sich nach dem current aus und wird selbst zum current.

Als Randnotiz, Du kannst Texturen änder indem Du Shift drückst und ein Face anklickst.



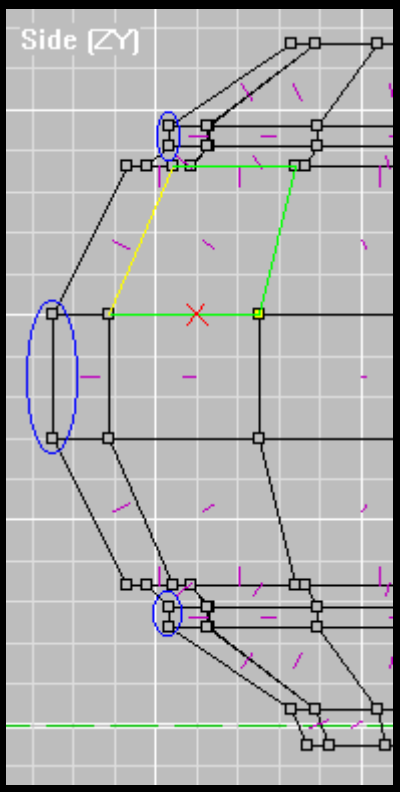
Das appliziert die current Texture in der Texturen-Palette auf das Face, das du anklickst.

Halte einfach die Ctrl gedrückt und gehe von Face zu Face wie links beschriftet.
Mache beide Rahmen und die Wände dazwischen. Rotiere die Ansicht, sodaß Du alle Faces erreichen kannst die Du mußt. Fange noch nicht mit dem runden Teil oder den Röhren an.

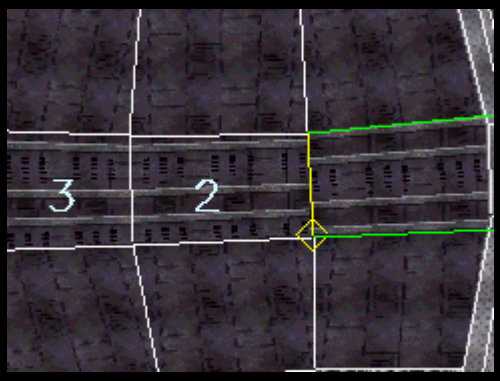
Lathen hat eine in Beziehung zur Textur-Ausrichtung interessante Eigenschaft. Jedes Face, das parallel mit der Achse ist, um die Du lathest sind nett und rechtwinklig zueinander (rechts eingekringelt). Das macht es möglich, sie aneinander auszurichten. Du kannst auch Faces ausrichten, die in einer vertikalen Reihe parallel zur Achse stehen. Ist ein Face einmal in einem Winkel zur Achse, kannst Du nicht von Seite zu Seite ausrichten (gut, Du kannst schon, aber Du bekommst seltsame Resultate).

Das macht runde Wände und Röhren pizzlig auszurichten. Du mußt wirklich über die Reihenfolge nachdenken, in der Du sie anklickst. Bei unserer Sphäre wirst Du die Zentral-Faces als erstes ausrichten wollen, dann nach oben und unten weiterarbeiten, von den Zentral-Faces aus.

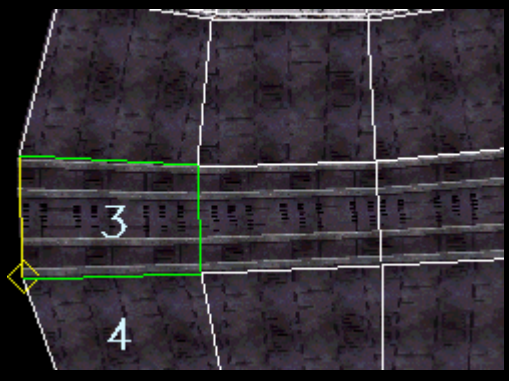
Wir werden uns jetzt auf ein Viertel der Sphäre konzentrieren.



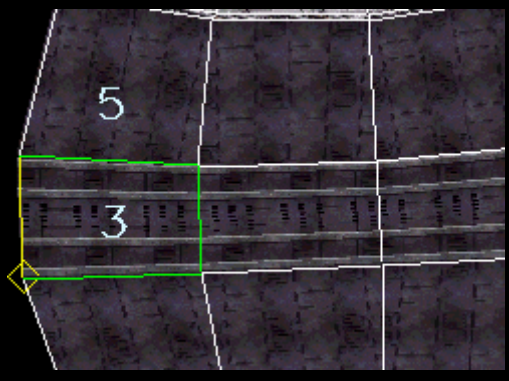
Alsl erstes mache jenes Face current, das im Bild gezeigt ist und verwende die Align-Box um es so auzurichten wie hier. Dann mache Ctrl-Klick um Faces 2 und 3 danach auszurichten.



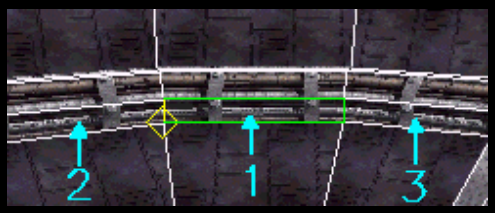
dann richte das Face unter 3 aus.



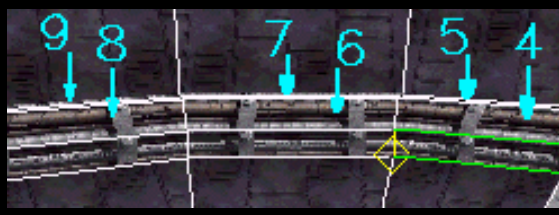
Mache Face 3 erneut current und richte Face 5 aus.
Mach die anderen zwei Reihen, indem Du von den Zentral-Faces aus arbeitest.



Die Röhren machen wir ähnlich wie oben. Wähle eine Reihe Faces, richte sie der Röhrenlänge entlang aus, dann geh zurück und arbeite um die Segmentreihen herum, immer von der selben Reihe beginnend. In unserem Fall werden wir vom Zentrum nach außen arbeiten. Wir wissen noch vom Bild (blaue Kringel), wo die Faces liegen die sich von Seite zu Seite ausrichten lassen.



Mache 1 current, **Ctrl-Klick** 2
Mache 1 current, **Ctrl-Klick** 3
Dann nach oben:
Mache 3 Current, richte 4 aus, dann 5
Mache 1 Current, richte 6 aus, dann 7
Mache 2 Current, richte 8 aus, dann 9



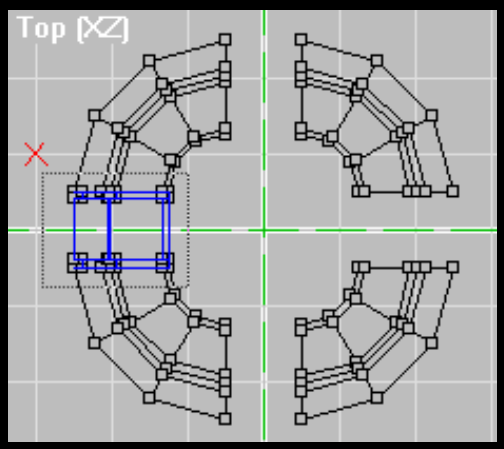
Oh, und bevor ich's vergesse, gehe zum Wandsegment zwischen den Rahmen und richte dies auch noch aus. Mach das mittlere Face current, arbeite Dich nach unten, mach das Mittel-Face wieder current, dann nach oben weiter.

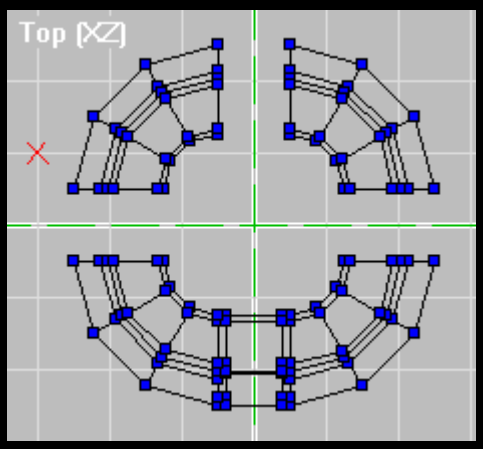
Ok, jetzt laß uns den Raum fertigmachen; aktueller Stand ist jetzt `Sctut06.orf`.
Es gibt zwei Wege wie wir das machen können.

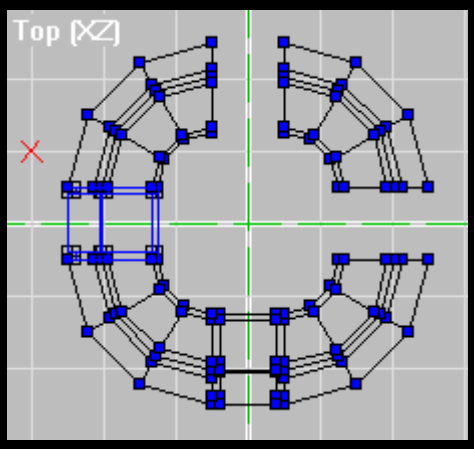
Markiere in der Top-Ansicht die Faces wie unten gezeigt.

Ctrl-R (->Vertex-Mode)
M (alle Verts markieren)
Achtmal **3** (Rotieren)

Ctrl-F (->Face-Mode)
Ctrl-Shift-V (Paste on top)



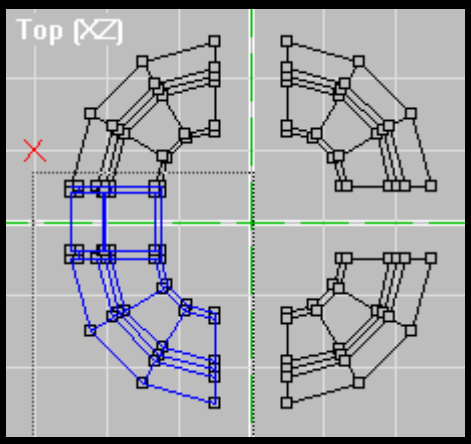




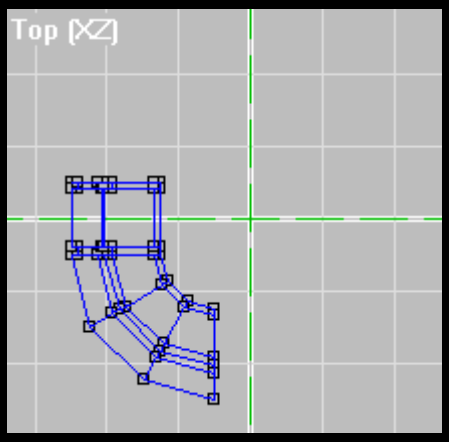
Gehe weiter durch den Zyklus **Ctrl-R**, **M**, Rotieren (**3**), **Ctrl-F**, **Ctrl-Shift-V** bis es fertig ist. Wenn Du das hast mach **Remove Extra Verts**.
Vergiß nicht, wenn Du erst einmal was kopierst, bleibt es in der Zwischenablage bis Du etwas Anderes kopierst. Deswegen können wir einmal kopieren und sooft einfügen wie wir brauchen.

Wenn wir diese Methode verwenden, würden wir immer noch zurückgehen müssen und die drei runden Sektionen texturieren/ausrichten, die wir zuvor nicht gemacht haben. Aber da ich 1) Texturen aussrichten hasse und 2) faul bin, werde ich Dir einen einfacheren Weg zeigen.

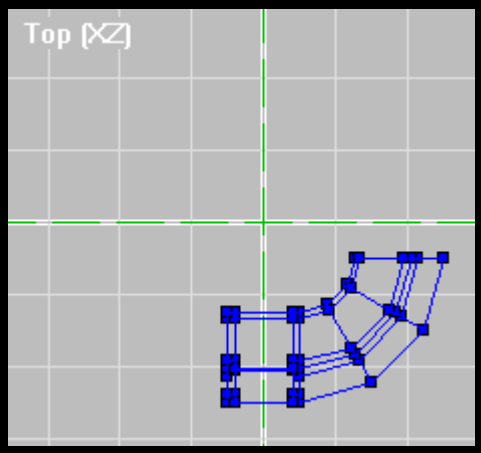
Markiere in der Top-Ansicht die Faces wie unten gezeigt.
Ctrl-C (kopieren)



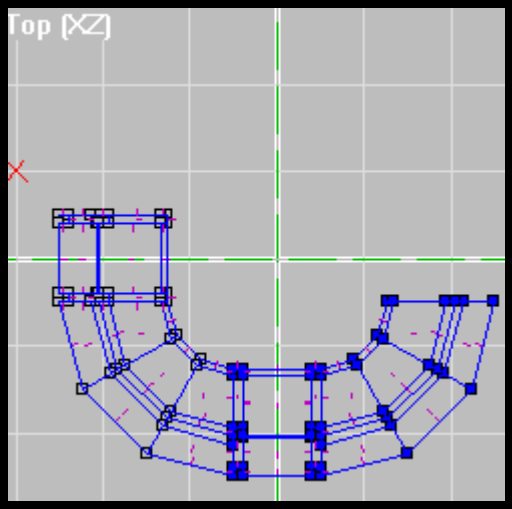
M (alle Faces markieren)
alle Faces löschen.
Ctrl-Shift-V (Paste on top)



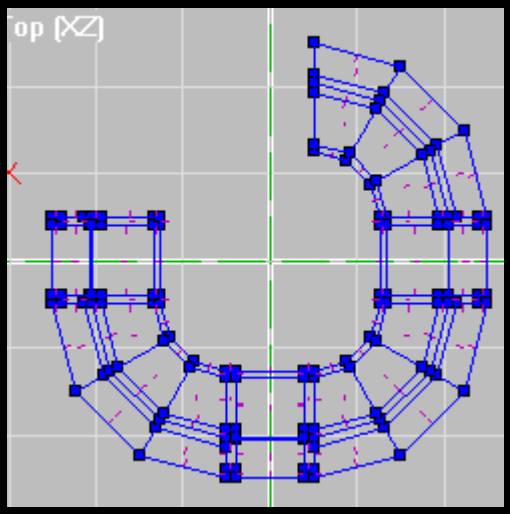
Ctrl-R (->Vertex-Mode)
M (alle Verts markieren)
Achtmal **3** (Rotieren)



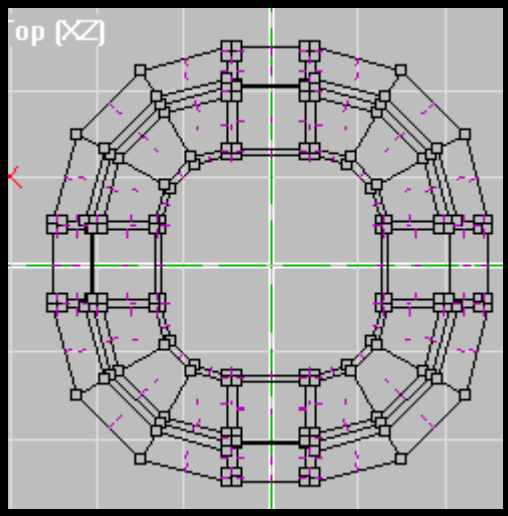
Ctrl-F (->Face-Mode)
Ctrl-Shift-V (Paste on top)



Ctrl-R (->Vertex-Mode)
M (alle Verts markieren)
Achtmal **3** (Rotieren)
Ctrl-F (->Face-Mode)
Ctrl-Shift-V (Paste on top)



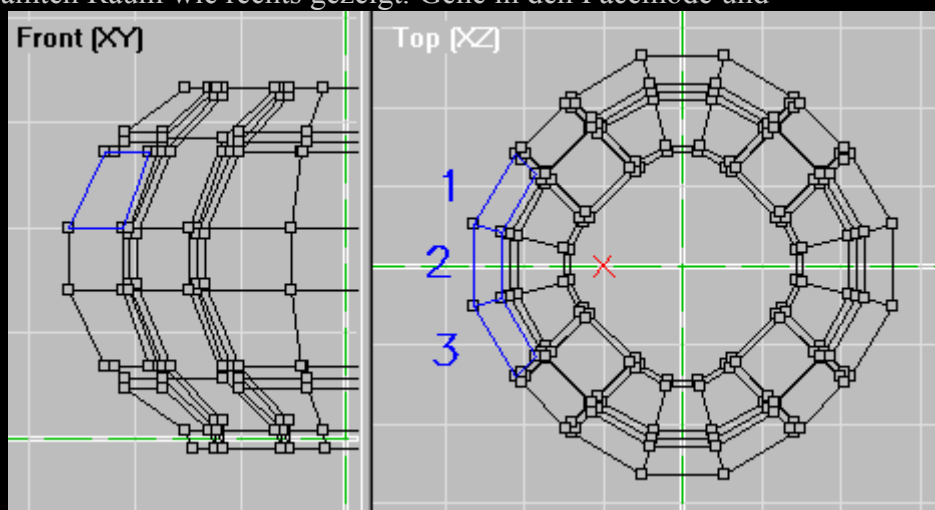
Ctrl-R (->Vertex-Mode)
M (alle Verts markieren)
Achtmal **3** (Rotieren)
Ctrl-F (->Face-Mode)
Ctrl-Shift-V (Paste on top)



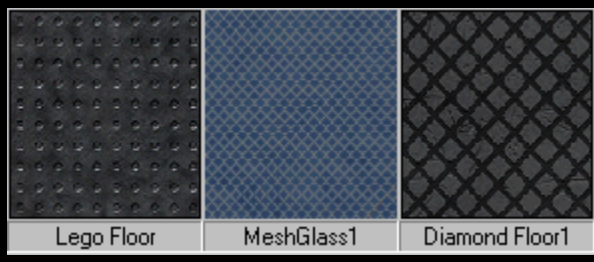
Remove Extra Verts und speichern. Aktueller Stand ist jetzt `sctut07.orf`.
Alles was nun noch zu tun bleibt, sind die Fenstervorsprünge und Boden bzw. Plafond abzudecken.

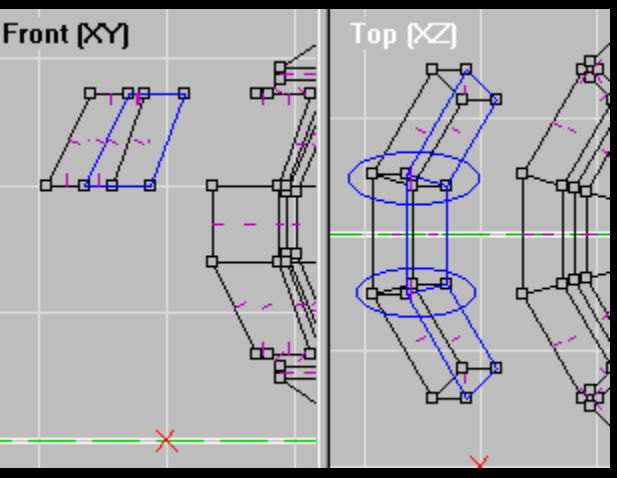
Als erstes die Fenstervorsprünge:

Markiere alle Verts und rotiere den gesamten Raum wie rechts gezeigt. Gehe in den Facemode und markiere die drei gezeigten Faces. In diesem Fall ist es am besten, das Face current zu machen und dann mit der **Space**-Bar zu markieren.
Kopieren, dieselben drei markieren und löschen. Wir setzen die Fenstervorsprünge auch auf die rechte Seite, also lösche auch diese drei Faces.

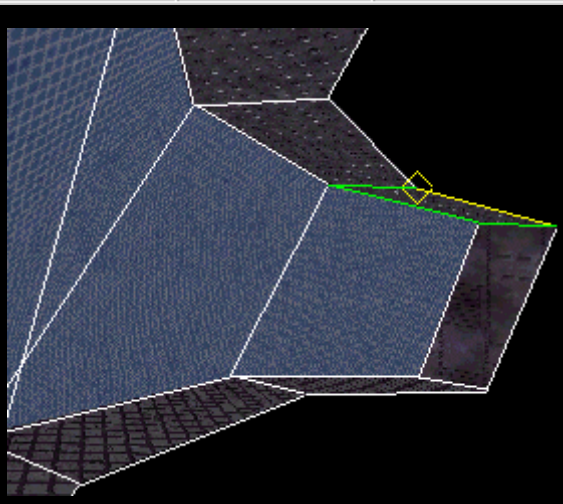


Setze das Gitter auf 50, **Ctrl-Shift-V**, dann bewege die Faces 50 Einheiten nach links um daran zu arbeiten.
Markiere die drei Faces und extrudiere sie entlang der X-Achse, um -15 Einheiten (=15 Einheiten nach links). **Remove Extra Verts**.





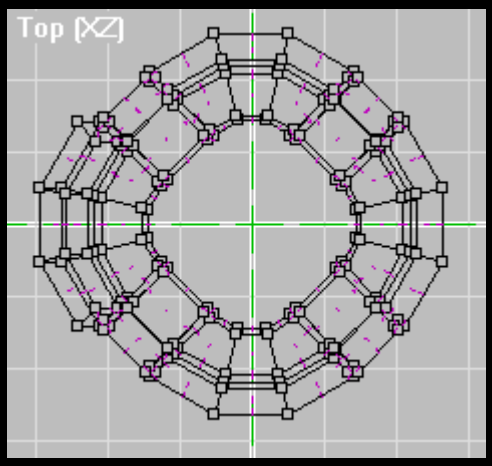
Lösche die drei Faces welche wir zum Extrudieren verwendet haben und die zwei die ich eingekringelt habe. Im 3d-Fenster setze auf Texture with Outline, und verwende **Shift-Klick** um die neuen Faces so zu texturieren, daß es wie rechts aussieht. Vergiß nicht die Default UVs zu setzen und die Texturen auszurichten.

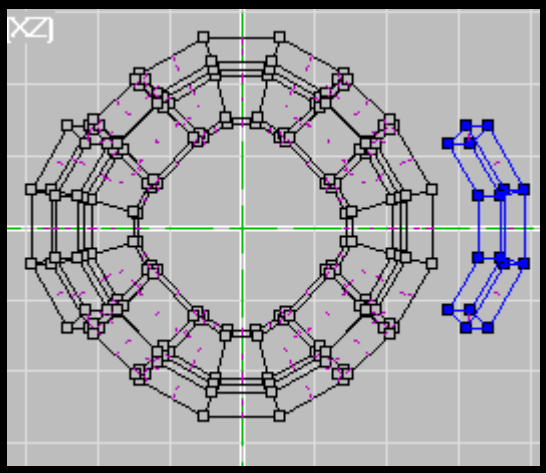


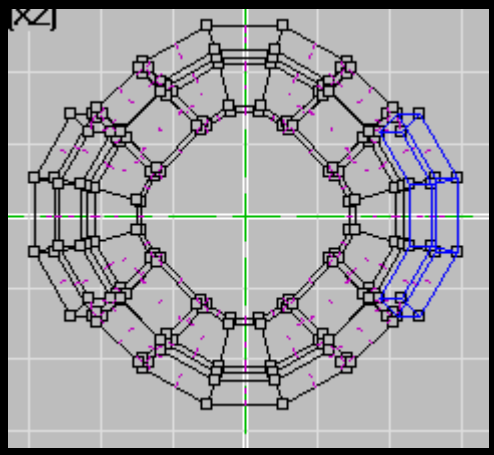
Nun markiere und kopiere die Faces unsrer neuen Sektion.
Markiere sie erneut und bewege sie zur Sphäre (50 Einheiten)

Ctrl-Shift-V (Paste on top)
In den Vert-Mode gehen und die Verts der gerade eingefügten Sektion markieren.
Rotiere sie auf die andere Seite.

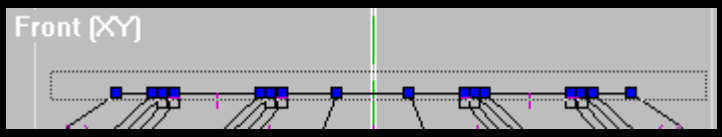
Bewege sie zur Sphäre (50 Einheiten nach links)
Remove Extra Verts
Speichern



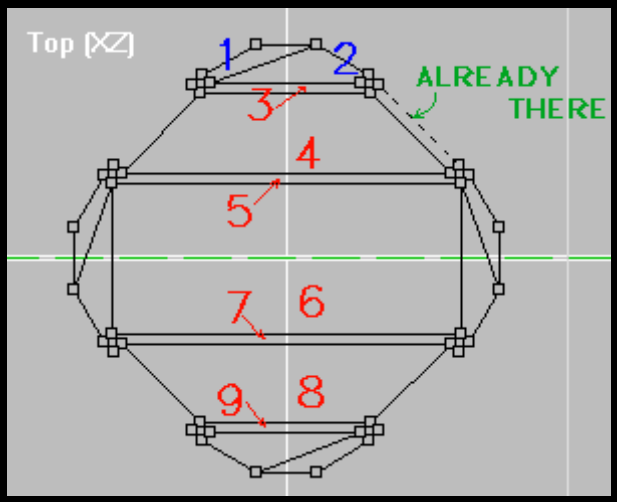




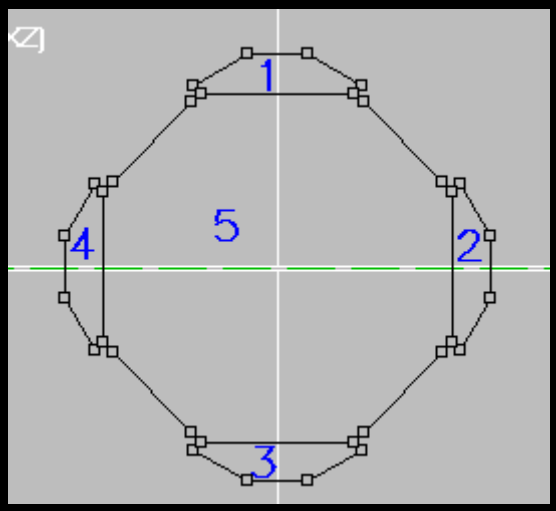
Ok, laß uns noch den Boden und den Plafond abdecken und wir sind fertig.



Mache die Frontansicht aktiv und setze das Grid auf 100. Markiere die Verts wie hier gezeigt. **Strg-C**, **Strg-Shift-V**, und dann bewege die neuen Verts 300 Einheiten nach rechts um daran zu arbeiten.



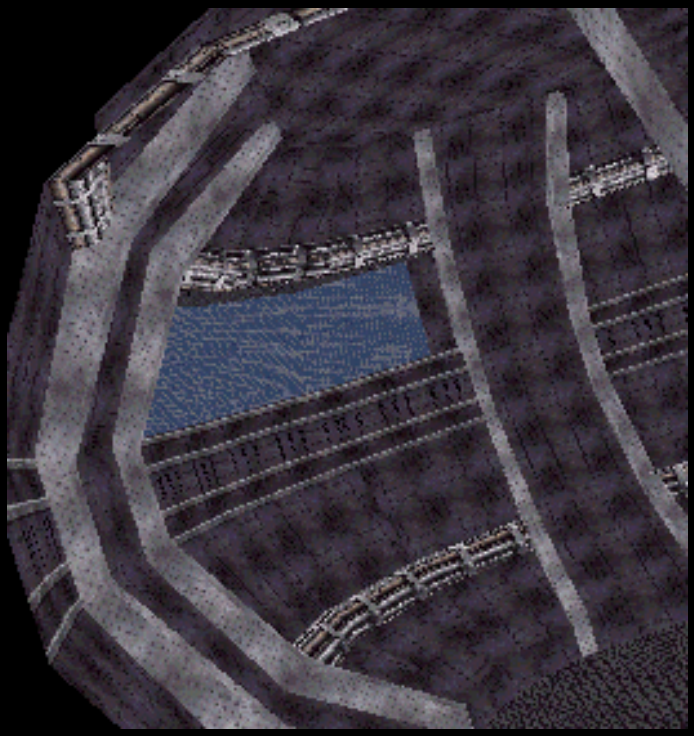
Markiere jeweils vier Verts und setze die Verts wie links gezeigt ein. Beobachte die Front-Ansicht um sicherzustellen daß alle Normals nach unten schauen Vereinige Faces 1 und 2, dann 3 und 4. Schließe das Vereinigen ab, so daß Du mit fünf Faces endest (rechts).



MArkiere die fünf Faces und schiebe sie 300 Einheiten nach links. **Remove Extra Verts**.

Mache den Boden auf die selbe Weise. Texturiere und richte die neuen Faces aus.

Speichern..... Du hast fertig. Vergleiche es mit rechts.



`sctut08.orf` ist der fertige Raum. Es mag dir lang vorgekommen sein, Schritt für Schritt hierherzukommen aber nun versuche das Tutorial nochmal zu durchlaufen. Ich wette, Du wirst förmlich drüberfliegen.

Du hast gelernt, wie man Lathe, Extrude, Kopieren, Rotieren, Paste on top und einige andere Werkzeuge anwendet, um eine interessante Struktur einfach schnell und kurzweilig aufzubauen.

Ich hoffe, diese Techniken helfen Dir dabei, mehr Zeit zu haben um kreativ zu sein.

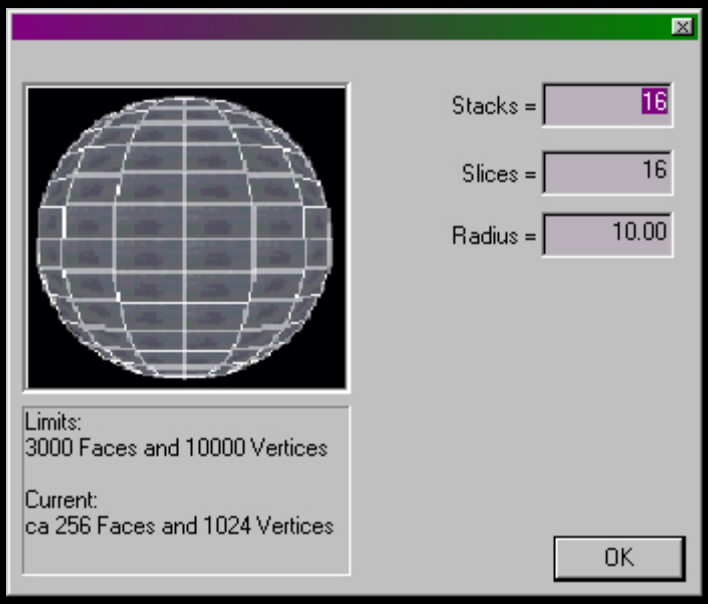
047 - Primitives

Ragil Ral

Seit v40 gibt es etwas, was die Arbeit enorm erleichtert - nämlich die Möglichkeit Grundformen on-the-fly zu erzeugen. Der Dialog für File->New wurde überarbeitet, und bietet nun weitaus mehr Möglichkeiten:

Die einzelnen Figuren und deren Einstellungen sollen in Folge hier beschrieben werden.
Für jede Figur gibt es eine Reihe an Einstellungen, das meiste ist eigentlich selbsterklärend. Ist die Figur als neuer Raum erstellt, so befindet sich die Konstruktion exakt in der Raummitte.

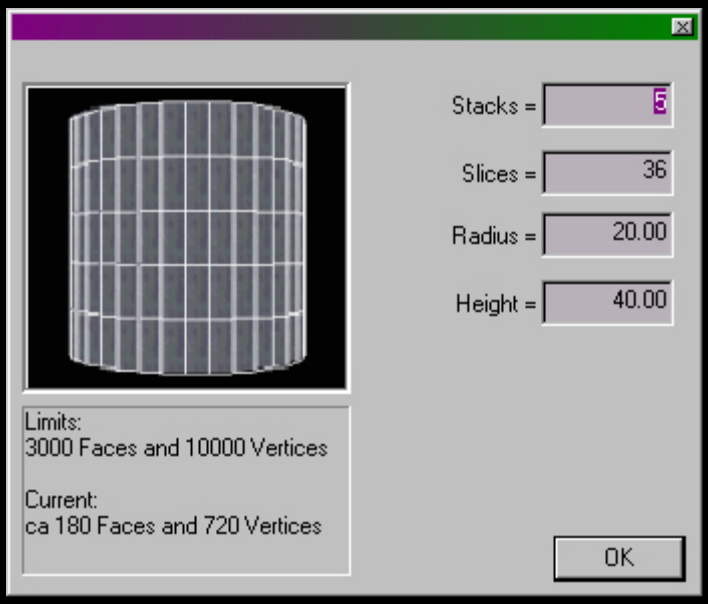
Sphere



Erzeugt eine Kugel.

Stacks: Wieviele Schichten die Kugel haben soll
Slices: Wieviele Segmente (Ecken ☺) soll die Kugel haben
Radius: ...

Cylinder

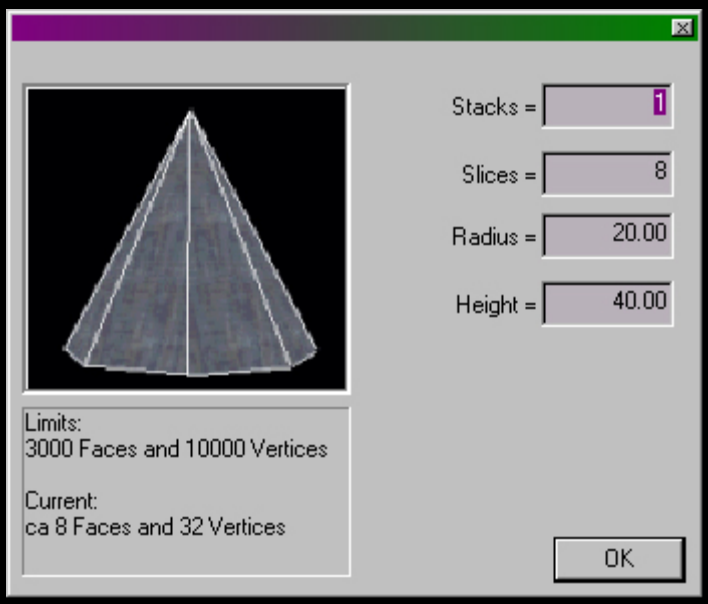


Erzeugt einen ...(Trommelwirbel)... Zylinder!

Stacks: Wieviele Scheiben soll der Zylinder haben
Slices: Wie 'rund' soll der Zylinder werden
... also wie bei Sphere.

Radius :Halbmesser, und
Height: Höhe des Zylinders.

Cone

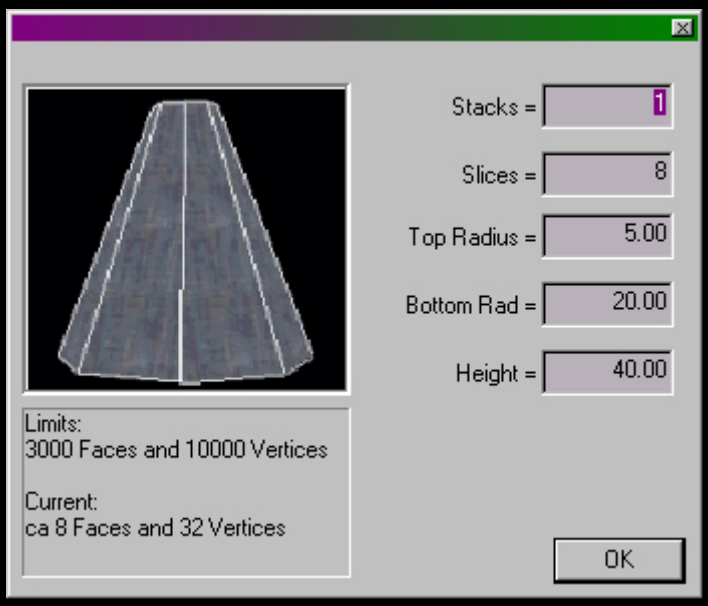


Erzeugt einen Konus.

Stacks: Wieviele Scheiben
Slices: wieviele Seiten

Radius und Height: die gewünschten Abmessungen

Flat Cone

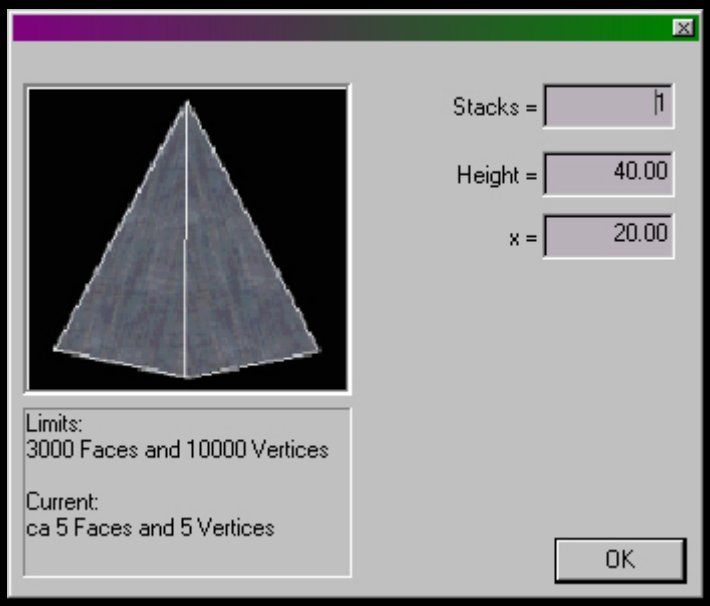


Erzeugt einen Konusstumpf.

Stacks und Slices: wie bei Cone,

Top Radius: Radius des oberen Deckfaces,
Bottom Rad: Radius des Bodenfaces
Height: ...

Pyramid

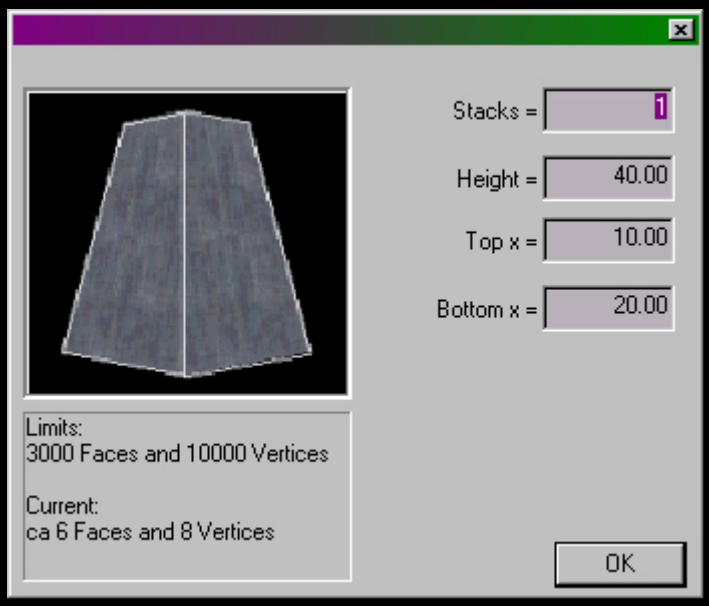


Erzeugt eine vierseitige Pyramide.

Stacks: wie gehabt, die Anzahl an 'Scheiben'
Height: wie hoch solls denn sein,
x: Seitenlänge der Basisfläche.

Diese Primitive hat den Vorteil, daß im Gegensatz zu Cone (mit Slices=4) die Seitenlänge, nicht die halbe Diagonale (=Radius) angegeben wird.

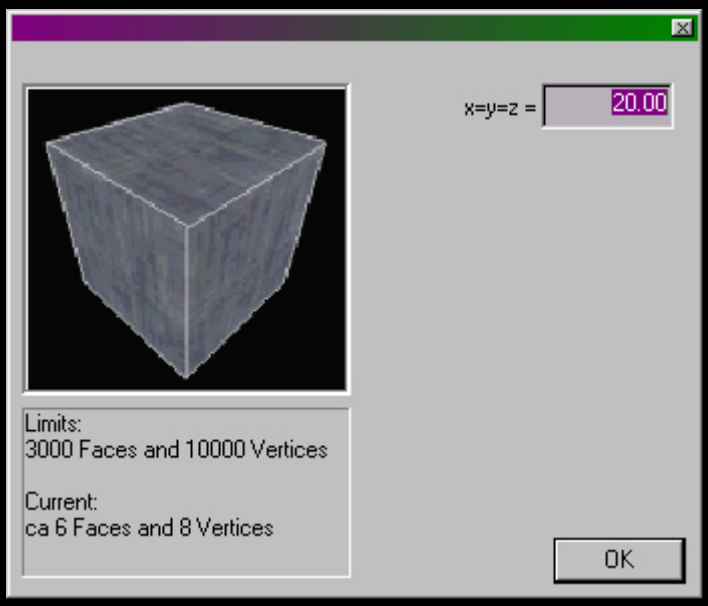
Flat Pyramid



Erzeugt einen Pyramidenstumpf, ähnlich wie Flat Cone.

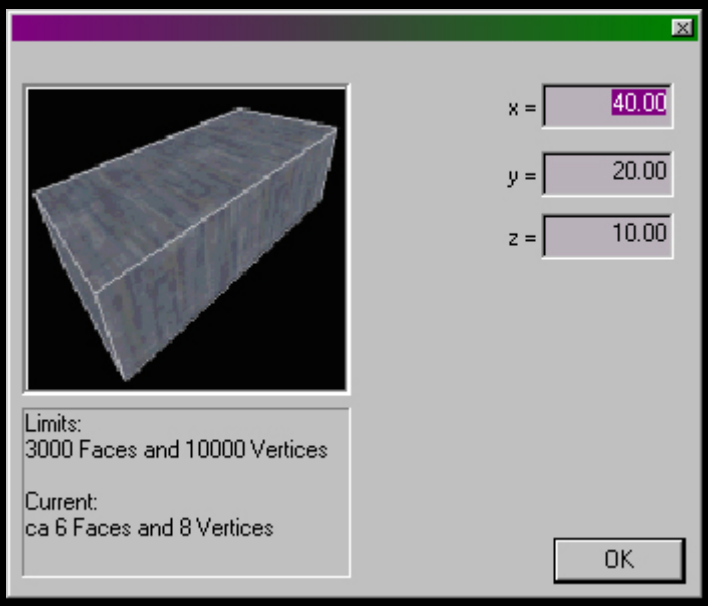
Stacks: ...
Height: ...
Top x: Seitenlänge des Deckfaces
Bottom x: Seitenlänge des Bodenfaces.

Cube



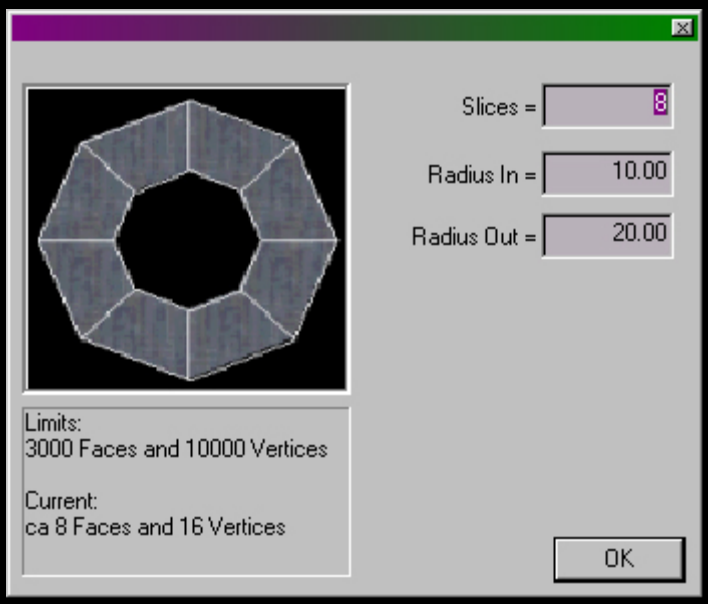
Falls es mal schnell gehen soll - Dies erzeugt einen Würfel mit der angegebenen Kantenlänge.

Box



Erzeugt einen Quader mit den angegebenen Maßen.

Ring

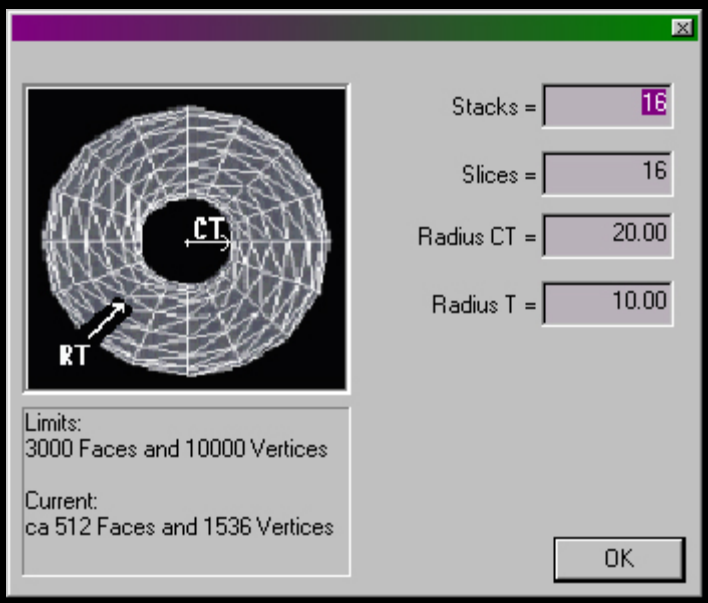


Dies erzeugt einen Ring aus Faces.

Slices: wieviele Flächen es sein sollen
Radius In: Innerer Radius
Radius Out: Äußerer Radius

Sehr praktisch für Bodenflächen, von denen man nach Bearbeitung der Grundform wegextrudiert.

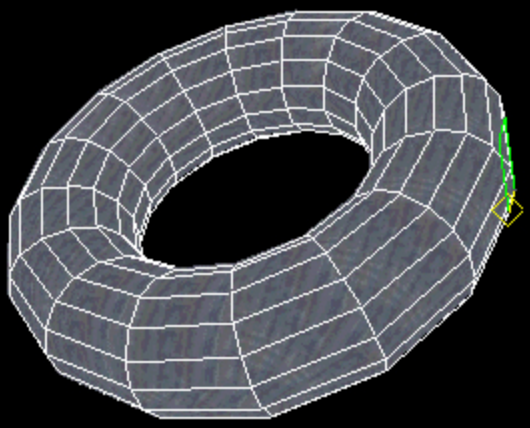
Torus



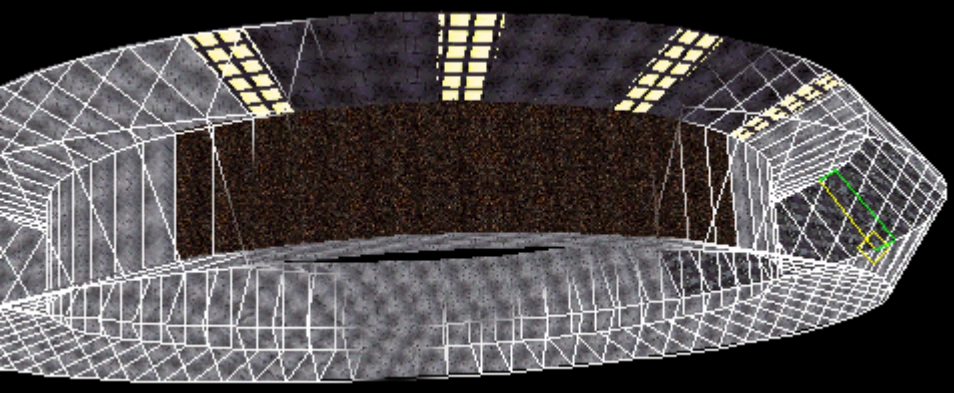
Der hier ist schon etwas spezieller: Ein Torus wird erzeugt, die berühmte Krapfenform.

Stacks: Wieviele Segmente,
Slices: wie 'rund'

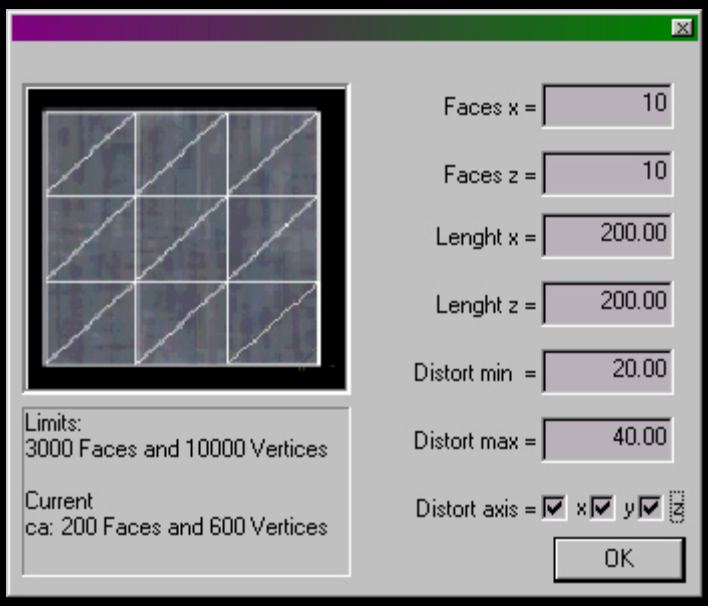
Radius CT: Innenradius der Form,
Radius T: Radius der 'Röhre' selbst.



Ein Torus mit Stacks=64 und Slices=5: ein perfekt symmetrischer, fünfeckiger Rundgang in fünf Sekunden



Grid



Damit erzeugt man eine auftriangulierte Fläche.

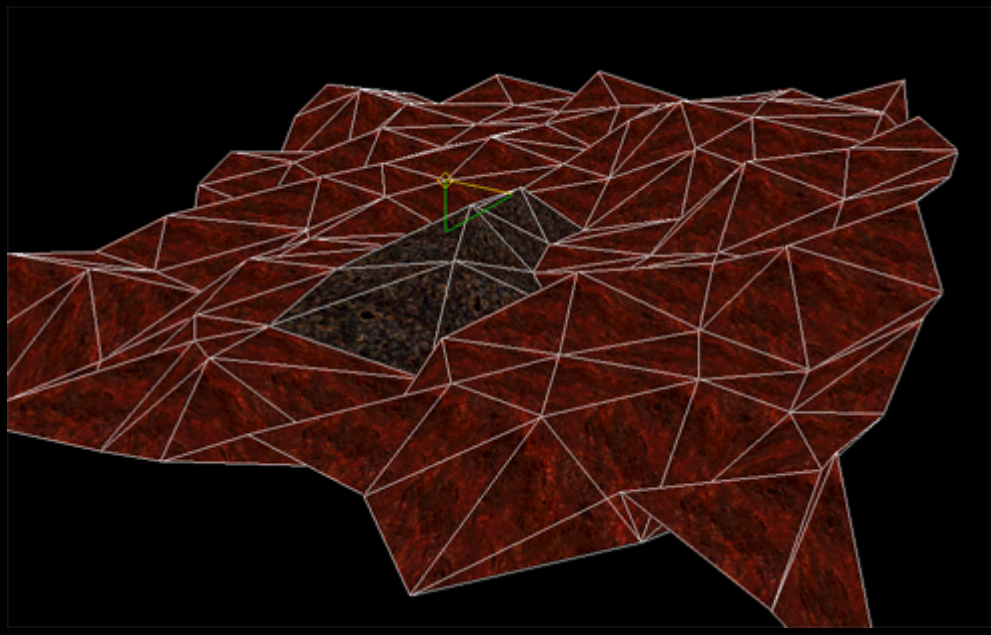
Faces x: In Wieviele Edges soll in Richtung x unterteilt werden,
Faces z: ....in Richtung z
Length x und Length z: die Ausmaße in die jeweilige Richtung

Und jetzt kommts: Es gibt die Möglichkeit, die Verts in bestimmte Richtungen 'zufallsverteilen' zu lassen.

Distort min: Minimaler,
Distort max: Maximaler Verschubwert.

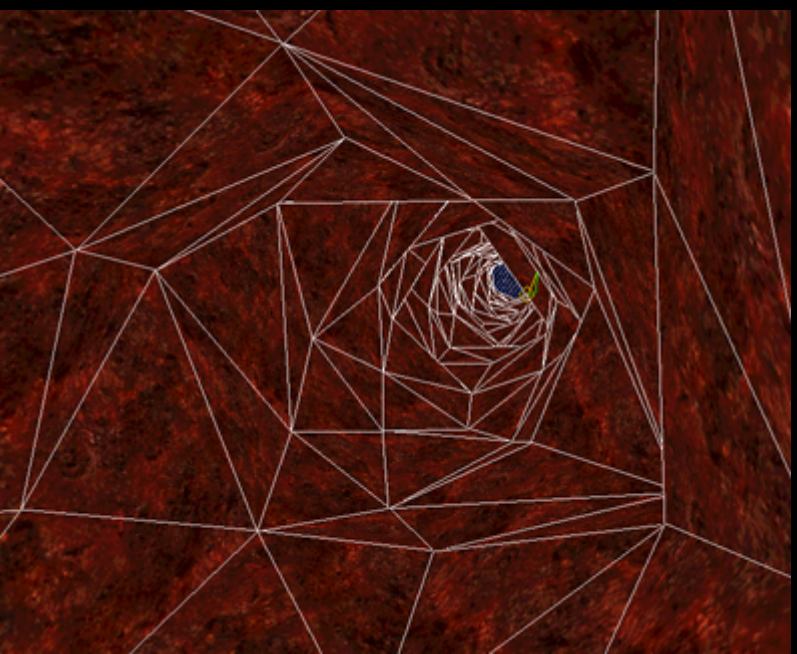
Distort axis: Hier gebt Ihr an, in welche Richtungen die Verts verteilt werden sollen; dies ist frei wählbar.

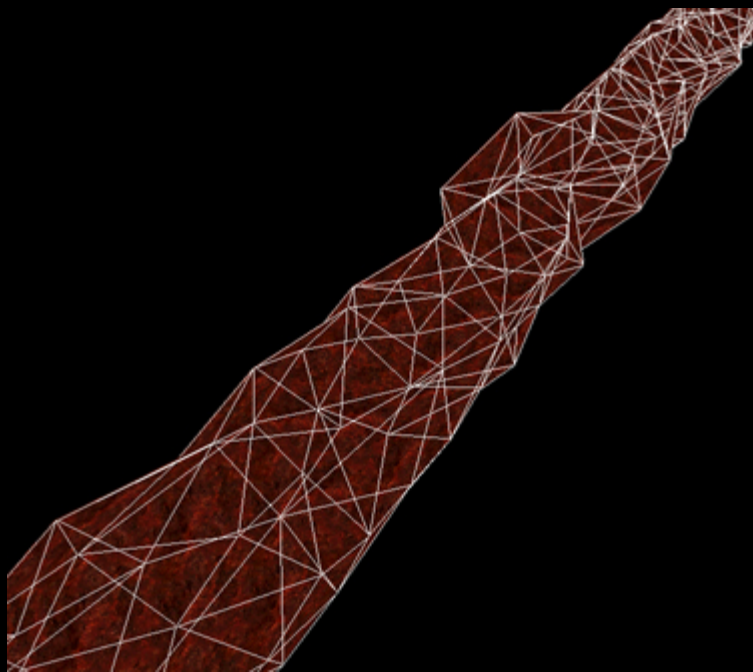
Die Werte die oben im Screenie stehen sind nicht die Default-Werte, wie bei den anderen. Die Werte erzeugen etwa folgende Figur:



Damit wird es ein Kinderspiel, unregelmäßige Formen zu erzeugen. Man beginnt einfach mit sowas wie da rechts, biegt und formt es - schmiedet es gewissermaßen, und dann näht man's zusammen.

Hier eine Ansicht von innen wie außen eines Ganges, den ich auf diese Weise erzeugt habe:

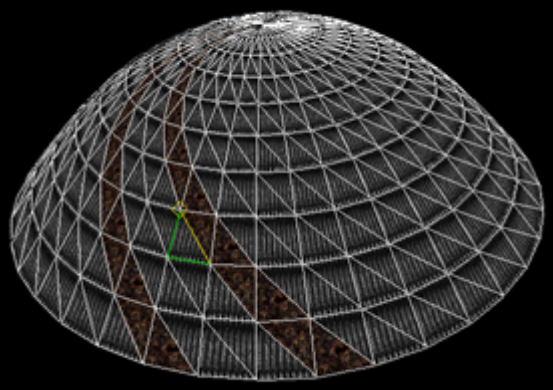


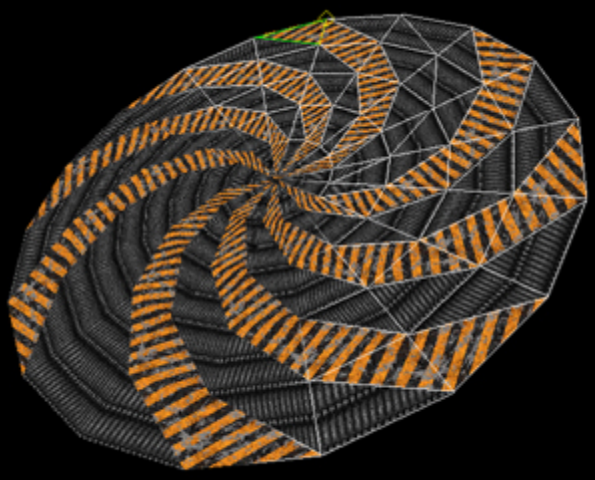


Es läuft immer gleich, Ihr erzeugt Euch ein 'Grundgitter', das 'schmiedet' Ihr anschließend in Form (durch verdrehen, verschieben oder auch mit Bend) und schließt es. Die Form wird mit steigenden Werten für Distort min und max natürlich immer aufgelöster, hier müßt Ihr dann beginnen auf die Lage einzelner Faces zu achten. Es empfiehlt sich außerdem, unbedingt bei Regionen die nicht besonders auffällig sind Faces zusammenzufassen, indem ihr sie Mit Snap-To-Vert verschwinden läßt; ansonsten erreicht Ihr sehr schnell hohe Face-Counts.

Solche Funktionen finden sich in der Regel nur in großen, professionellen 3d-Programmen!

Man kann aber auch sowas damit machen:

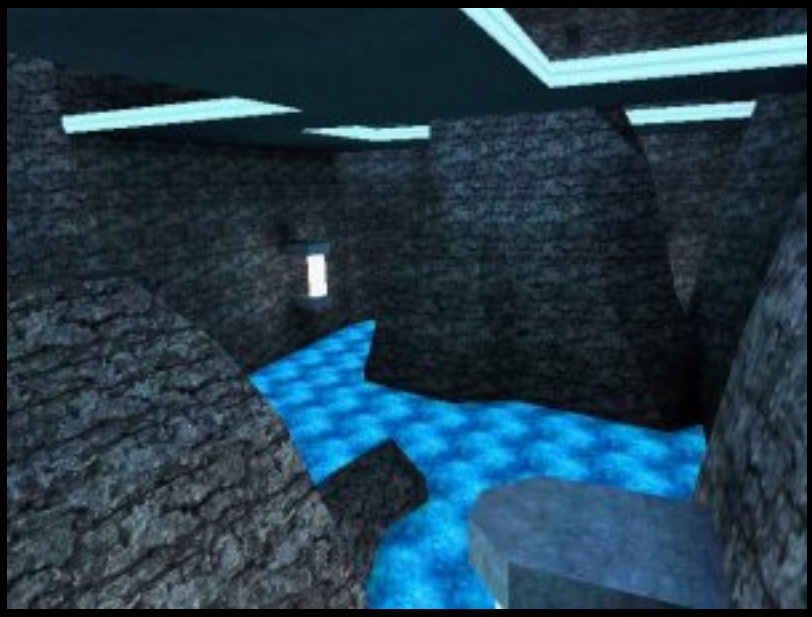




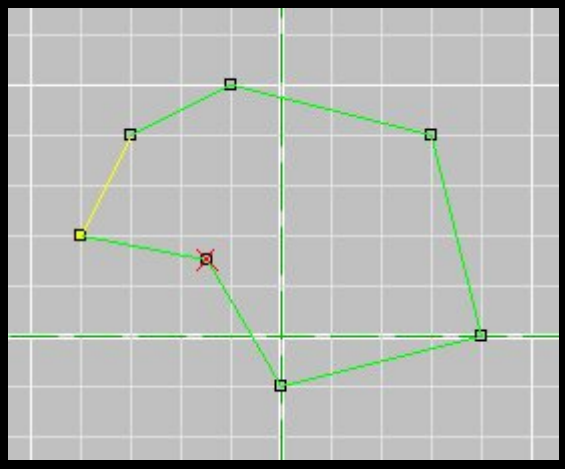
Diese Figuren wurden aus Grids erzeugt, die keine Distortion hatten. Danach habe ich sie gebendet, bei der linken die Kuppel hochgezogen.

048 - Felsformationen erzeugen

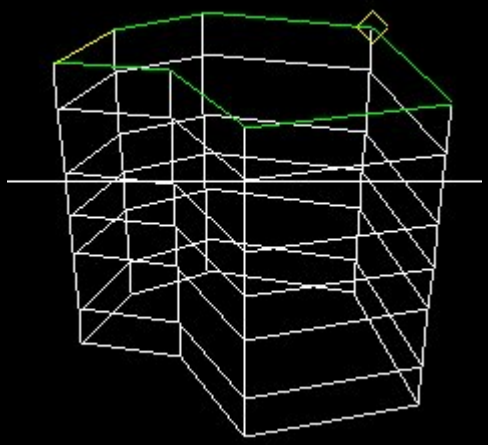
Hydra



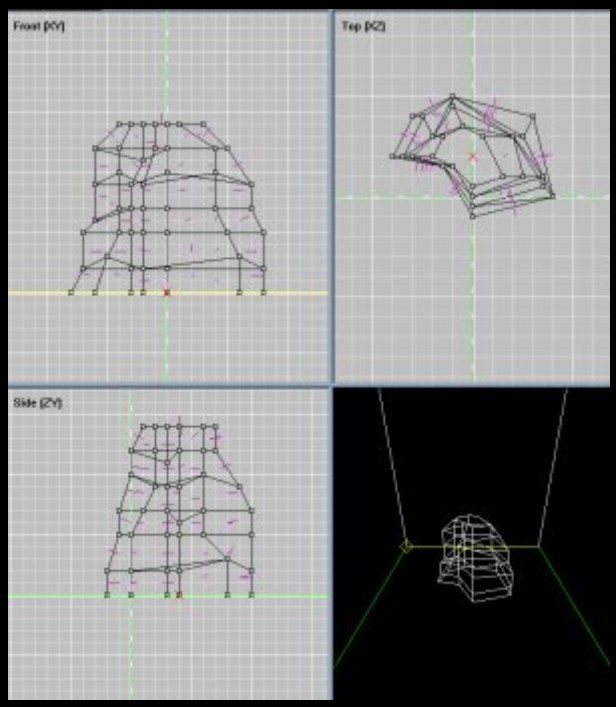
Felsformationen können echt kool aussehen, wenn sie ordentlich gemacht sind, und es gibt einen schnellen und einfachen Weg dafür. Aber nur eine Warnung, abhängig davon wie groß der Felsen ist und wieviel Details Du hineingearbeitet hast, können sie viele Polygone haben.



Als erstes, Du solltest eine grobe Idee haben wie der Felsen aussehen soll wenn er fertig ist. Etwa wie er gekrümmt ist, ob er geneigt ist oder was auch immer. Danach mache den Grund-Umriß des Felsens (etwa wie im Bild links) auf dem Boden (oder der Decke). Dieses Face muß nicht konvex sein, da es bald gelöscht werden wird.

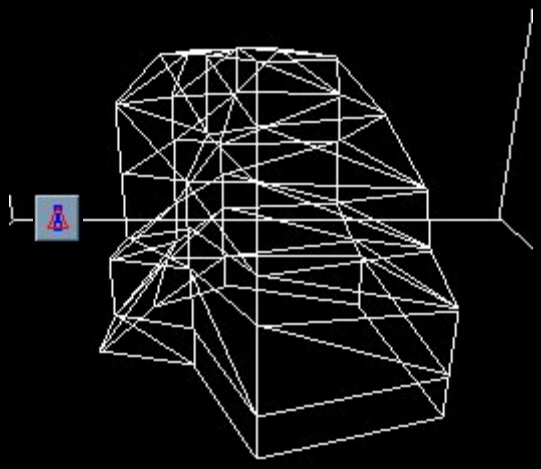


Wenn Du ein Face hast daß Dir zusagt, extrudiere es. Ich mache üblicherweise Extrusionen zwischen 10 und 15. Die Extrusionen zu staffeln läßt den Felsen natürlicher aussehen. Höre auf zu Extrudieren, wenn Du eine Höhe erreicht hast, die Dir paßt. Wenn Du während des Extrudierens vergißt Delete base face anzuticken, gehe zurück und lösche alle diese Faces. Vergiß allerdings nicht, das oberste zu behalten.



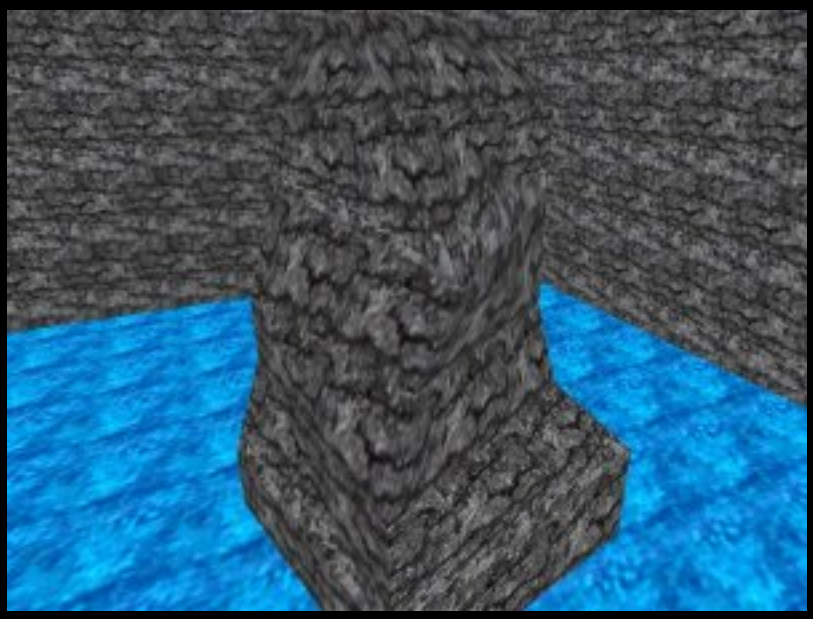
Nun zum spaßigen Teil. Stelle die Gitter in allen drei Orthogonal-Ansichten auf 5 Einheiten und fange dann an, Deine Verts wahllos herumzubewegen. Bewege sie nicht zuviel in eine Richtung (bewege eins oder zwei auch vertikal) vielleicht zumeist nur ein oder zwei Sprünge weit. Du solltest die Verts generell einwärts, zum Zentrum des Felsens hin bewegen. Danach solltest Du in D3Edit etwa sowas wie links sehen.

Nun bin ich mir sicher, daß Ihr jetzt alle realisiert habt daß dieses Objekt gerammelt voll mit Fehlern ist, wir werden das gleich richten. Gehe in den Face Mode und markiere Dein ganzes Objekt. Dann klicke den Button **Triangulate non-planar faces**.



Wow schau Dir all diese neuen Faces an, markiere Dein Objekt erneut und klicke wieder auf den Button. Wiederhole das bis D3Edit sagt daß alle Faces planar sind. Es sollte vier oder fünf Mal brauchen. Da Deine Faces bereits alle markiert sind, könntest Du sie gleich texturieren.

Danach schau Dir den Felsen im D3 an. Manchmal wurden die Verts in schlechte Positionen bewegt und haben ein paar Faces plattgemacht, sodaß Du zurück in die `.orf`-Datei gehst und entweder Den Felsen tweakst oder von vorn anfängst. So oder so, diese Technik ist ein schneller und einfacher Weg um kool aussehende Felsformationen zu machen.



049 - Tunnel im Höhlen-Stil

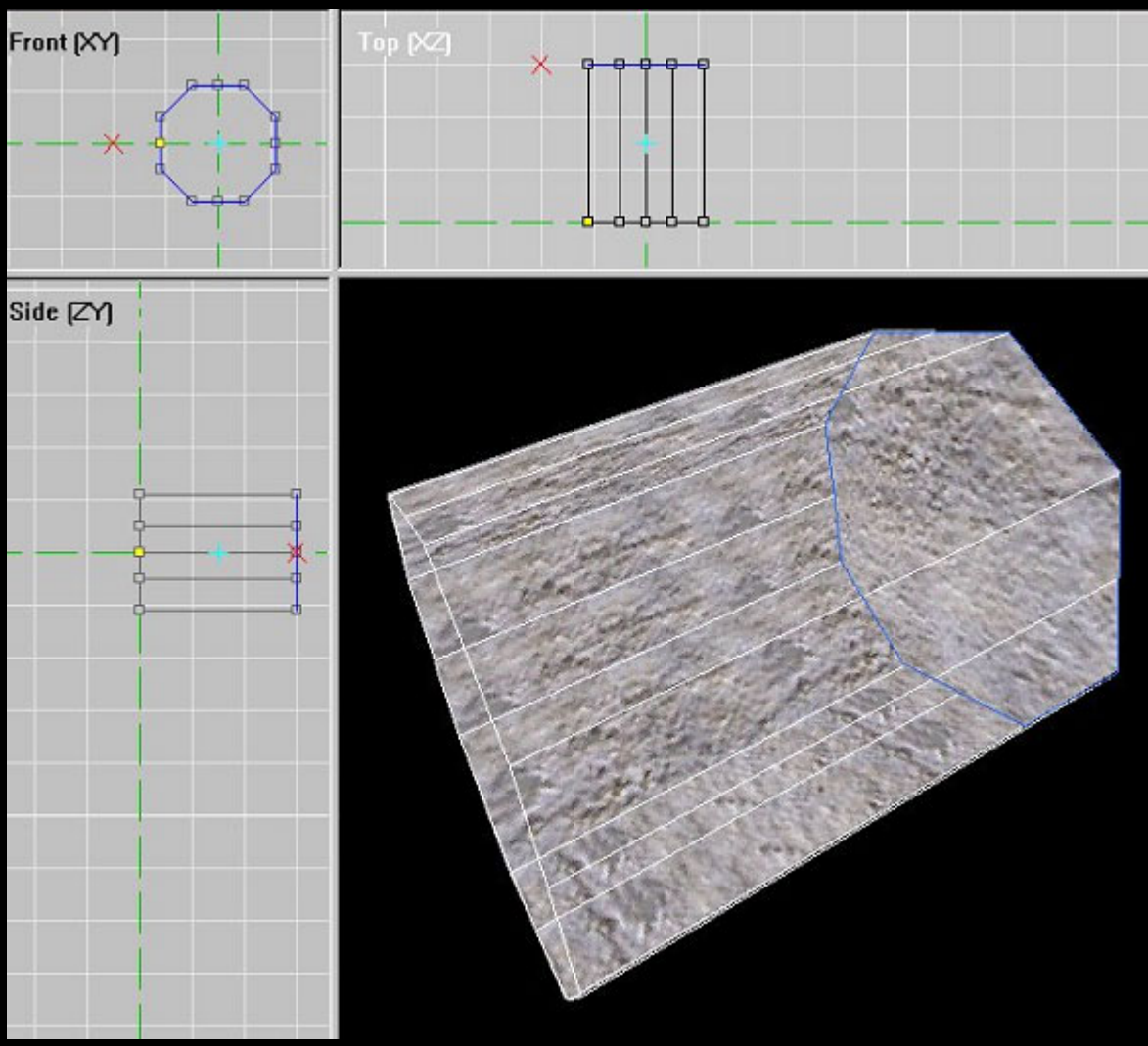
WillyP

Überarbeitet von Fischlein

Dies hier ist nichts mehr für blutige Anfänger. Das bisher Behandelte sollte nun gut sitzen!

Einen Raum machen

Zuerst wirst Du einen Raum erstellen müssen. Das erste Polygon (Face) das Du machst sollte die Größe des Eingangsportals haben. *Das größte Schiff, der Phoenix, hat einen Radius von 8,018578 Einheiten, so dass das Polygon mehr als doppelt so groß sein muss (20 ist ein gutes Minimum)*. Je mehr Verts Du benutzt, desto mehr Polygone wirst Du erhalten, und desto detaillierter werden die Wände sein. Wie man an dem Bild feststellen kann, habe ich ein Achteck erstellt, aber ein extra Vert an jeder Seite hinzugefügt. Benutze aus dem selben Grund ziemlich kurze Extrusionen (im Bild sind 30 sichtbar).



Eine andere Option... mein Favorit... ist, in einem existierenden Raum ein Face auszuwählen, daß ein Portal werden wird. Dann, in der World View, drücke **Ins** und presto, Du hast einen neuen Raum! Du mußt nur vorsichtig sein, daß Du das Portal in keinster Weise verzerrst, veränderst.

Ein Tip: Die meisten Anleitungen (zumindest die, die ich gelesen habe) instruieren Dich, ein Polygon zu markieren, die Normale umzudrehen und dann zu extrudieren. Ich überspringe die Markieren-und-Drehen-Routine, indem ich eine negative Zahl im Extrusions-Dialog benutze. Wenn Du vom ersten Polygon eines neuen Raumes extrudierst, benutze eine positive Zahl und wähle Delete Base Face ab. Zweite Extrusion, gib eine negative Zahl ein, wähle „Point outward“ für die Polygonnormalen (für einen Innenraum; die negative Zahl dreht sie um) und wähle die Delete Base Face Option aus. Für spätere Extrusionen wähle ein Polygon aus und drücke **Strg+E**, dann extrudiere um den selben Wert.

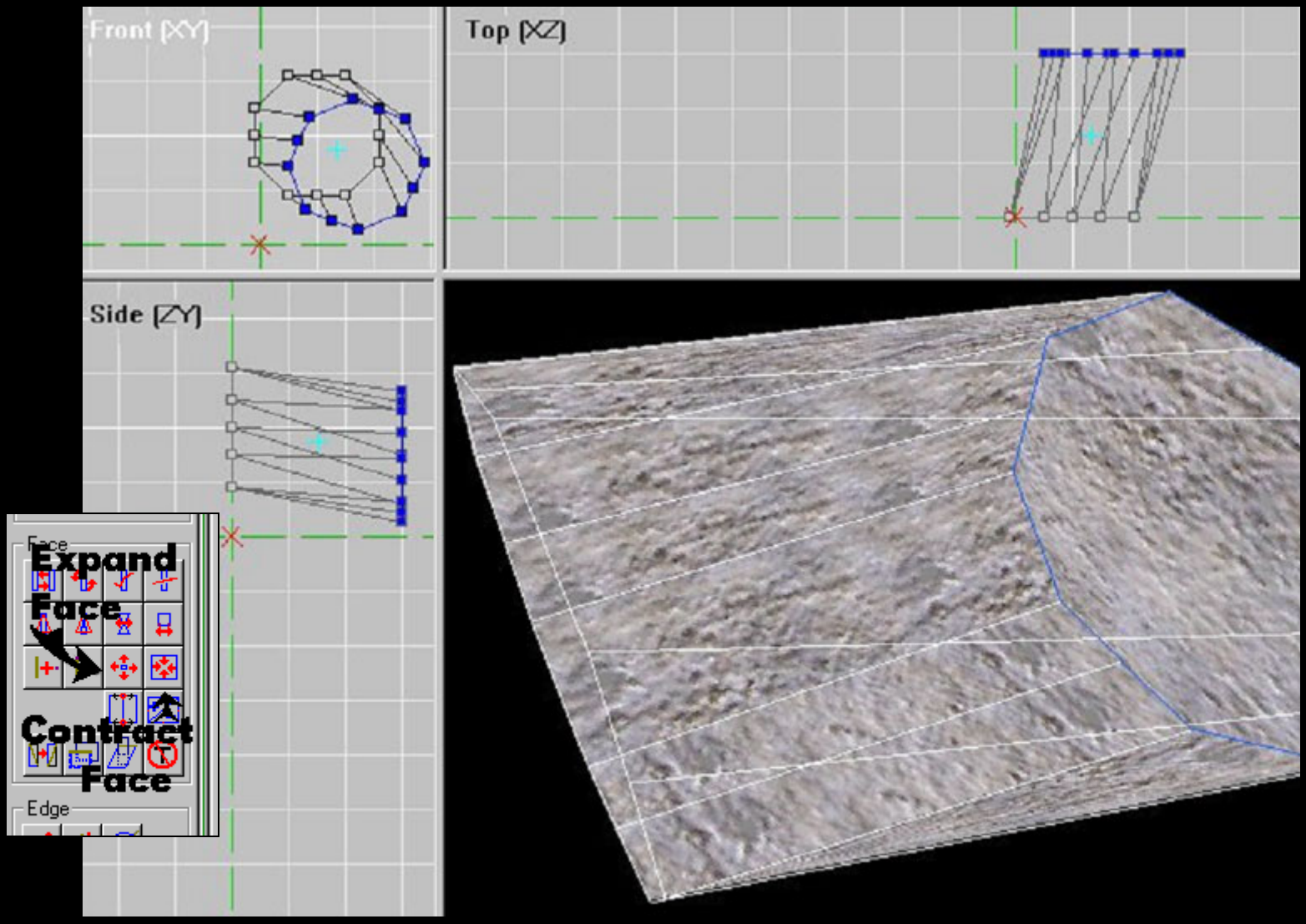
Ein Polygon bewegen

Beachte das markierte Polygon (blaue Umrandung) im vorigen Bild. Wir wollen das ganze Polygon bewegen, ohne es zu verzerren. Drücke den Key-Button und scrolle runter bis Modify items... Moving marked items …“ und probiere die verschiedenen aufgelisteten Möglichkeiten aus, um das markierte Polygon zu bewegen (Du hast den Raum vorher gespeichert… oder?) Beachte, dass hoch und runter relativ auf das Fenster bezogen ist, in dem Du Dich befindest und in der 3D-Ansicht nicht funktionieren werden.

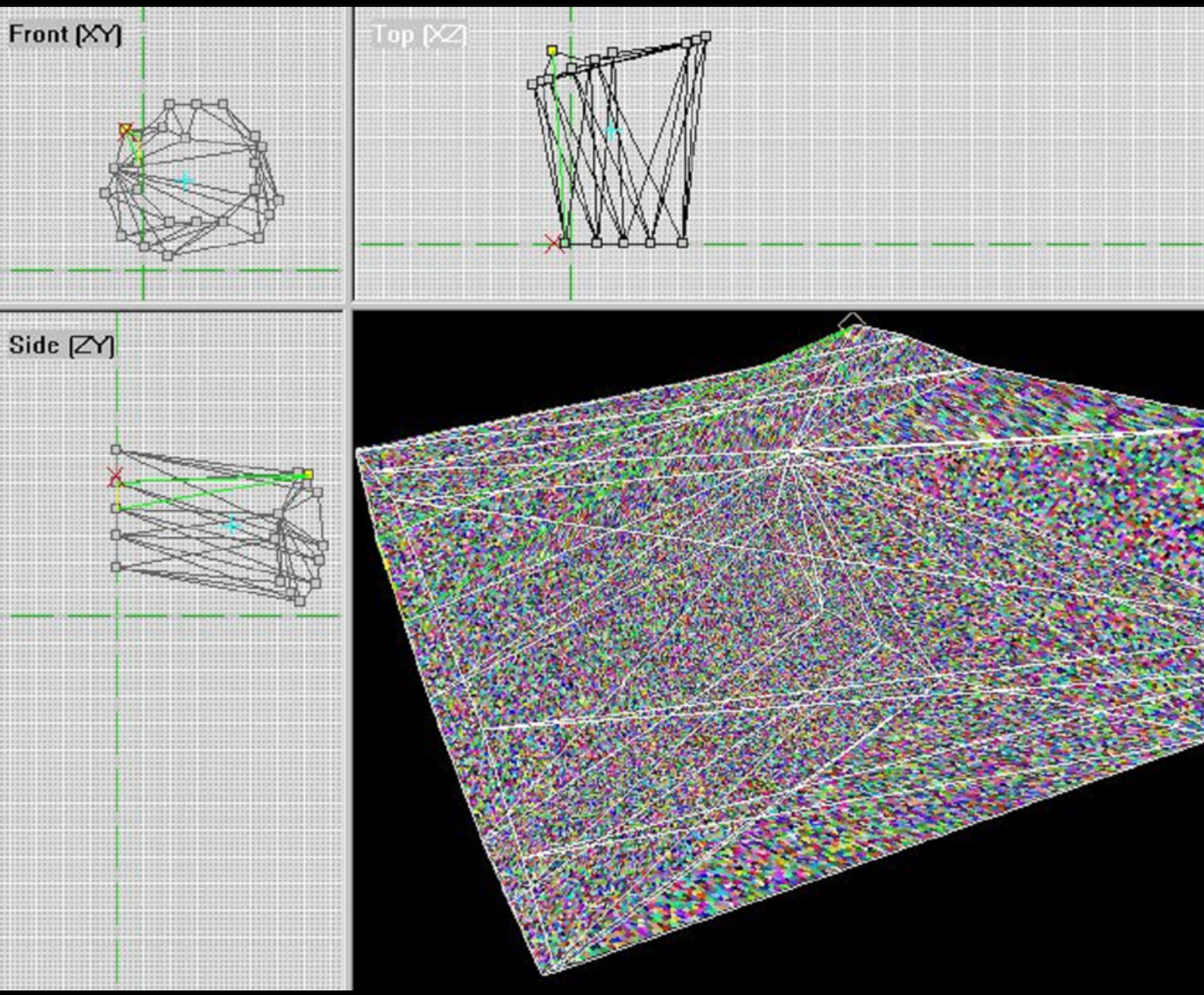
Ein paar Sachen, die nicht direkt aus Key hervorgehen: Die Punkte (Verts) oder die Polygone (Faces) können markiert und bewegt werden, je nachdem in welchem Modus Du Dich befindest. Punkte und Polygone sind keine Objekte, d.h. „rotate object“ funktioniert damit nicht. „Twist“ wird Markiertes um den Schnittpunkt der grünen Referenzlinien drehen. Bewege diese mit **Strg-Klick**. Benutze alle vier Ansichten, um die Auswirkungen dessen zu sehen, was Du tust.

In Bewegung Bleiben!

In diesem Bild habe ich die markierten Punkte in der Vorderansicht leicht gedreht. Jetzt sind meine zwei Achtecke verschoben und leicht verdreht. Mach Dir jetzt keinen Kopf über nicht plane Polygone, außer dem, von dem aus Du weiter extrudieren wirst. Benutze die **Expand**-und **Contract**-Buttons, um die Größe des Polygons zu verändern. Schaue auf die markierten Punkte in der Draufsicht und der Seitenansicht rechts, sie sind alle auf dem selben Raster. Wenn nicht, bewege sie auf eine Linie, um das Polygon wieder plan zu machen, so dass Du wieder davon extrudieren kannst. Das Polygon sollte auch nicht konkav sein. Wenn Du eine Linie zwischen zwei Punkten ziehen kannst, die außerhalb des Randes des Polygons liegt, hast Du ein konkaves Polygon. Behebe das, indem Du Punkte in der Vorderansicht bewegst.



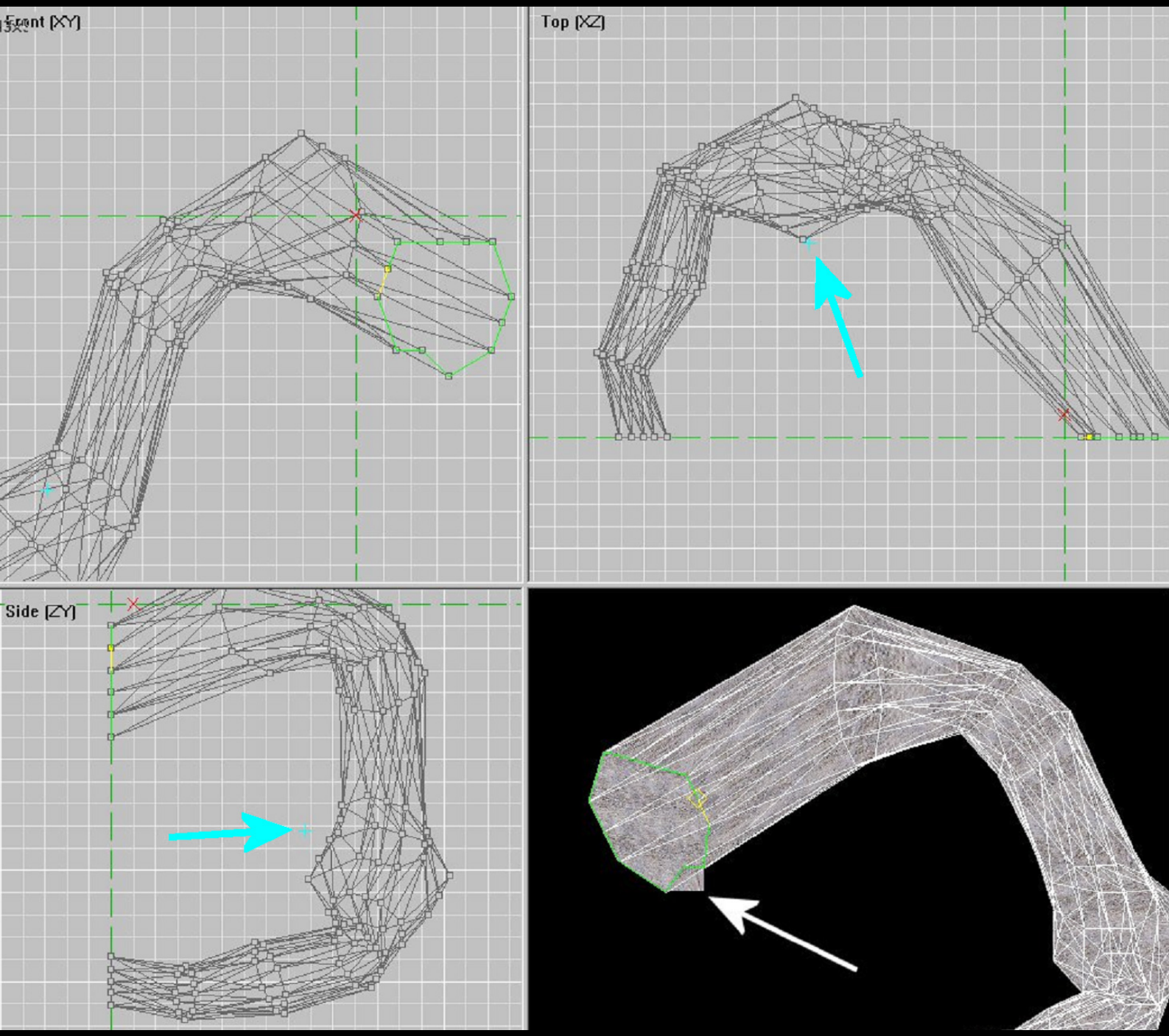
Hier (unten) habe ich das Polygon noch etwas mehr bewegt. Aber ich habe auch einen Punkt aus der Reihe mit dem Polygon gezogen. Ich habe dann den Modifizierungs-Button **Face planar check** benutzt, nachdem alle Polygone markiert wurden. Dies hat alle nicht planen Polygone repariert, indem es sie in Dreiecke zerschnitten hat. Da das Polygon, von dem wir extrudieren wollen nicht plan ist, ist es es ebenfalls in Dreiecke zerlegt worden. Das größte von denen ist zu klein, um durchzufliegen, so daß es keinen Sinn macht, davon zu extrudieren. Ich werde zu meiner vorherigen Sicherung zurückkehren welche ich gemacht habe, bevor ich den Punkt aus der Reihe gezogen habe. Beachte auch, daß die Textur zur „Default Texture“ wechselt, wenn auf diese Weise Polygone zerlegt werden.



Sind wir schon da?

Schaust Du auf das folgende Bild, kannst du sehen, dass ich den Winkel und die Größe des Polygons geändert, es um -30 Einheiten extrudiert und das ein paar Mal wiederholt habe. Dann habe ich die Punkte der letzten Extrusion markiert und diese in der Seitenansicht auf die gleiche Rasterlinie gelegt, mit **Snap to grid**. Jetzt habe ich ein paar Fehler, das letzte Polygon ist durch die Ausrichtung der Punkte konkav (weißer Pfeil in der 3D-Ansicht). Das ist einfach zu beheben, ich werde einfach ein paar Punkte in der Vorderansicht bewegen, bis es stimmt. Die türkisen Pfeile in der Drauf- und Seitenansicht zeigen auf ein gleichfarbiges Fadenkreuz. Das ist das Raum-Zentrum, D3 benutzt es für die Berechnung der Geräuschfortpflanzung, Raketenaufschaltung und KI-Bewegungen, unter anderen Dingen.

Es ist essentiell, daß das (geometrische) Zentrum innerhalb des Raumes liegt. Seine Position wird als Zentrum aller Eckpunkte des Raumes berechnet.



Wir könnten das beheben, indem wir eine Handvoll Punkte nehmen und in der Seitenansicht nach links bewegen. Du müsstest außerdem einige in der Draufsicht nach unten bewegen. Und Du müßtest sie weiter bewegen als Du denkst, da sich das Raumzentrum auch nur ein bisschen bewegt. Eine andere Methode wäre, ein Unterobjekt in den Raum einzufügen, an eine Stelle, wo die Verts das Zentrum wieder in den Raum zurück ziehen würden. Im Moment werden wir akzeptieren, dass der Raum einen Fehler hat.

Wir haben trotzdem noch ein wenig mehr an diesem Raum zu arbeiten. Würdest Du durch diesen Raum fliegen, würdest Du klar sehen, dass er eine Serie von Achtecken ist. Und obwohl diese in zufälligen Positionen sind, sind sie trotzdem als Achtecke erkennbar und stören den Effekt einer natürlichen Höhle. Der nächste Schritt wird dem abhelfen.

Greif Dir eine Handvoll Verts

Ziehe einfach an einem oder mehreren Punkten herum, markiere sie, bewege sie herum, dorthin und dahin, benutze alle drei 2D-Ansichten. Entferne die Markierung wieder, wiederhole das… Du weißt schon was ich meine.

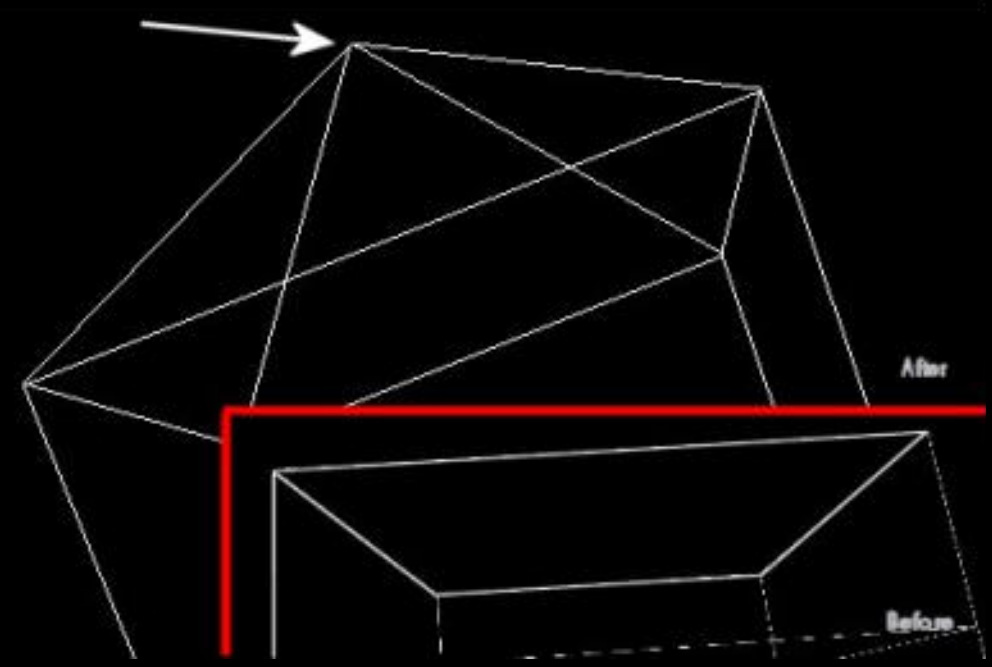
Verifiziere den Raum!

Wir haben jetzt eine recht nett aussehende Höhle, aber auch ein paar Probleme...

Non-planare Faces

Zum einen, die meisten Faces sind non-planar. Der einfachste Weg wäre alle Faces zu markieren (K, Errors, Non Planar Faces->**Mark**), und dann mit dem **Face Planar Check** zu splitten.

Ein anderer Weg wäre, sie mit der Einstellung Needle zu extrudieren. Needle gibt Dir viel mehr Faces zum herumspielen. Es teilt das Polygon auf, indem es ein Vert in die Mitte dessen setzt. Das Polygon wird in Dreiecke zerlegt, eins für jede Kante des Polygons. Danach wird der eingefügte Punkt (Pfeil) um den Betrag entlang der eingestellten Richtung (X, Y, Z oder Normal) verschoben, welcher für die Extrusion eingegeben wurde. Wenn der Tunnel genug Platz bietet, kannst Du das neue Needle-Vert rauf- oder runterziehen, um Tropfsteine nachzubilden. (Eselsbrücken sich zu merken welche wohin zeigen -> http://de.wikipedia.org/wiki/Tropfstein)



Wenn ich **Lathe** oder **Extrude** benutze, um einen großen Raum zu konstruieren, werde ich zuerst alle Polygone durchgehen und **Extrude**, Needle drücken, die Entfernung hier und da ändern. Dann gehe ich zurück und „Extrude-Needle“ einige der resultierenden Polygone hier und da, dann drücke und schiebe ich Punkte soweit notwendig, um einen Zufalls-Look zu bekommen, mit Felsenvorsprüngen in den Raum gezogen, Felsspalten herausgedrückt, gebrochene Felsen, die aus dem Boden herausragen. Und herunter von der Decke.

Room Center nicht im Raum

Gut, es ist recht klar daß – im Bild mit den türkisen Pfeilen– das Raumzentrum sich nicht im Raum befindet.

Wenn Du sicher bist, dass das Raumzentrum innerhalb des Raumes ist, aber die Verifizierung sagt, es ist nicht so, dann aktualisiere zuerst auf die letzte D3-Edit Version. Ältere Versionen haben irrtümlicherweise festgestellt, das Zentrum wäre nicht im Raum, wenn man an einem einzelnen Raum gearbeitet hat. Bekommst Du dann immer noch diese Nachricht? Mit diesem Stil eines Raumes ist es sehr schwer zu sagen, indem man auf die 2D-Ansichten schaut, ob es wirklich so ist oder nicht.

Ein Weg das zu prüfen, ist den Raum in einen neuen Level zu importieren. Beim ersten Raum wird der `player0`-Start im Zentrum des Raumes sein. Du kannst in die aktuelle Raumansicht gehen und in der 3D-Ansicht klar sehen, ob sich der Startpunkt im Raum befindet oder nicht.

Portale nicht in Sichtlinie

Man sollte zwischen den Portalen eine Linie ziehen können, ohne den Raum zu verlassen. An jedem Portal befindet sich ein BNode. Diese führen die AI von Portal zu Portal, die AI versucht dann von einem zum anderen Portal zu gelangen. Durch die U-Form des Raumes kann sie das hier nicht. Die Bots haben diesbezüglich etwas Spielraum, da sie sich ein wenig bewegen, wenn sie steckenbleiben.

Wir alle haben schon gesehen wie der Guidebot in einer Wand steckenbleibt, im vergeblichen Versuch durch die Wand zu kommen während er versucht Dich zu Deinem Ziel zu führen. Homing-Waffen habe ein etwas härteres Leben, da sie an derselben Wand ‚Selbstmord' begehen, während Dein Opponent nur um die Ecke im nächsten Raum ist. (siehe dazu das Tut über BNodes.)

Texturierung

Den Höhlen-Tunnel zu texturieren ist recht einfach, markiere alle Faces mit **M** und dann kannst Du eine Felsen-Textur auf alle Faces legen, mit einem Klick. Oder, wenn Du Sub-Objekte hinzugefügt hast kannst Du, im Face-Mode und in der 3d-Ansicht, die 0 auf dem Nummernblock drücken. Dies markiert nur die Faces, die mit dem current verbunden sind. Im Ausrichtungs-Dialog wähle Marked faces und drücke Default UVs.

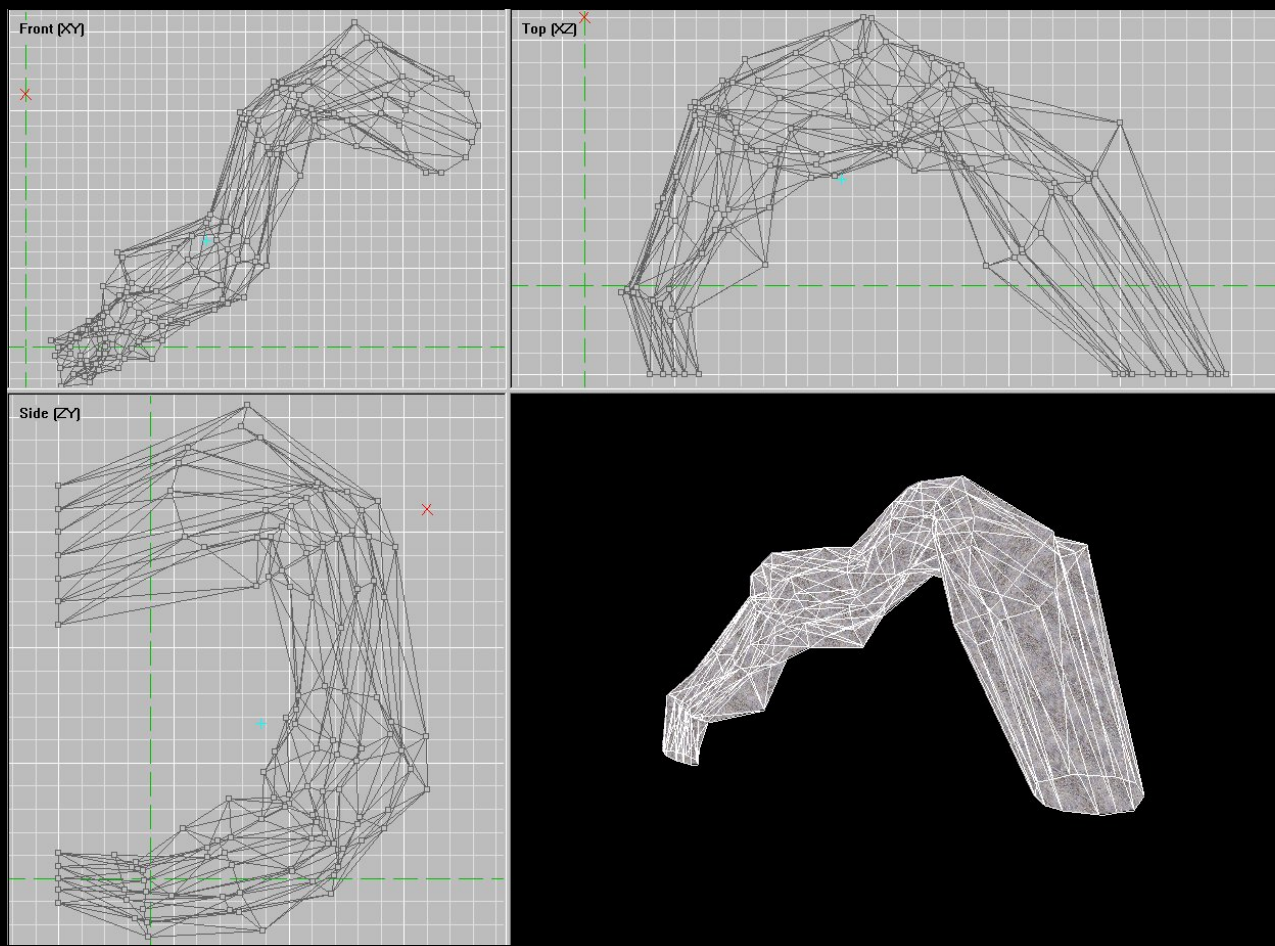
Da dieser Raum ein Durcheinander von Felsen ist, ist es realistischer die Texturen nicht aneinander auszurichten.

Beleuchtung

Das einfachste wäre es, den Raum nur mit Ambient auszuleuchten (Umgebungslicht), aber das wäre nicht das Realistischste. Einige andere Optionen zur Beleuchtung wären Kristalle, die aus den Felsen wachsen, Senken mit Lava, Lampen und natürlich Kombinationen aus allem.

Lavapfützen zu machen ist recht einfach. Ziehe einfach ein paar Verts nach unten, mach den Boden der Höhle ein wenig niedriger und ändere die Texturen zu Lava. Diese ist sehr hell, so braucht es nicth viel, und es ist etwas was Du in einer Höhle erwarten würdest. Zumindest in einer Descent-Höhle jedenfalls ☺

Das ist der fertiggestellte Tunnel. Ich werde eine Lampe hinzufügen, um das Raumzentrum in den Raum zu bekommen. Der Raum braucht sowieso Licht. Ich könnte Umgebungslicht benutzen, aber ich mag den Effekt des Übergangs von dunklen zu hellen Bereichen, Punktlicht ermöglicht das.



Bildschirmfoto aus dem Spiel heraus:

Von gut beleuchteten Gegenden in dunkle zu kommen kann sehr gruselig sein!

Abschnitt D – Debug

Bis jetzt wird denke ich jeder bereits so seine Erfahrungen mit Baufehlern gemacht haben bzw. deren (vielleicht sogar geglückte? ☺) Beseitigung.
Dank der neuen Editorversionen wurde es dankenswerterweise wesentlich vereinfacht, Fehler aufzuspüren und zu reparieren.

Wenngleich es hier nicht darum geht Etwas zu bauen; fahre dennoch den Editor hoch und baue die Fehler nach – es gibt Dir ein Gefühl, wie sich der Editor dann verhält, wie sich die Fehler verhalten, wie das Ganze aussieht etc. etc... Der Editor könnte dann zwar mal abstürzen, aber ich glaube das ist dann auch wurscht 😬

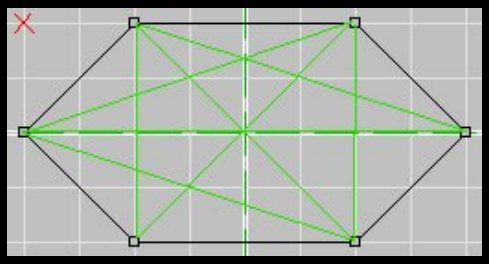
Grundlegende Fehler fixen				
050	**Konkave Faces**	Reparieren von konkaven Faces	Hydra	194
051	**Non-Planar Faces**	Loswerden von unebenen Polygonen	Fischlein	195
T-Joints zu Fuß reparieren				
052	**T-Joint**	T-Joints fixen: Anleitung	Fischlein	197
053	**T-Joints: Zweite Version**	T-Joints fixen in Screenshots	The Edge	198
Fehlerprüfungs- und Behandlungsroutinen.				
054	**Verifiziere Dein Level**	Kurz und bündig erklärt	Kyouryuu	202
055	**Verifiziere Deine Mine**	mit Beispielen	WillyP	207

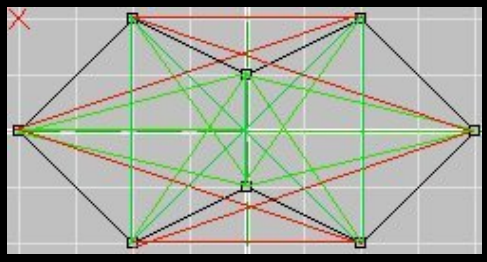
Toc

050 - Konkave Faces

Hydra

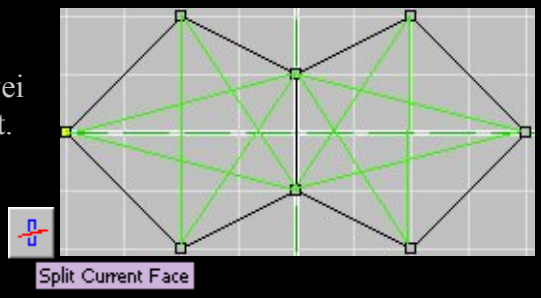
Jedes Face in Descent 3 muß konvex sein; wenn da eines ist welches nicht konvex (=konkav) behandelt das Spiel das strange. Im rechten Bild ist ein Beispiel eines konvexen Faces. Ich habe Linien zwischen allen Vertices eingezeichnet, beachte daß keine der Linien außerhalb des Faces verläuft.





Wie ich zuvor sagte, wenn Du zwischen alle Verts eine Linie setzt, sollte keine außerhalb sein. Wenn Du eine außerhalb hast (rote Linien), hast Du ein konkaves Face. wie das hier links. Glücklicherweise ist das Problem recht einfach zu beheben.

Dies ist ein Weg das Problem zu lösen. In D3Edit ist es Dir gestattet Faces zu splitten (=aufteilen). Ich habe die mittleren zwei Verts ausgewählt und auf den Button **Split current face** geklickt. Zur Rechten ist das Ergebnis dessen, nun habe ich zwei konvexe Faces, welche D3 korrekt handhaben wird.



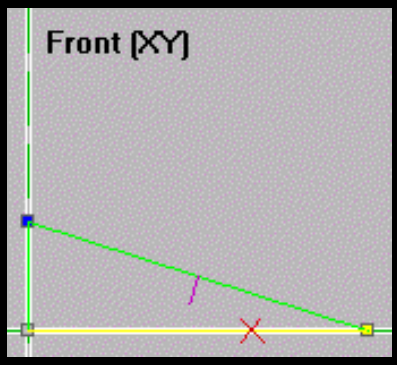
051 - Non-Planar Faces

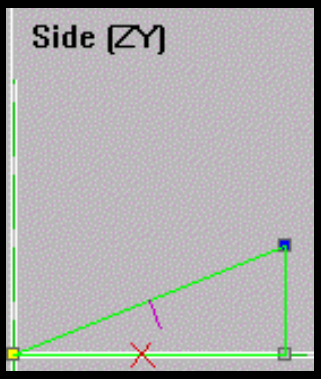
Fischlein

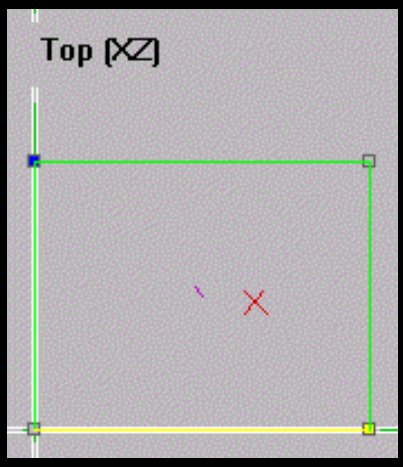
(überarbeitet)

Zu diesen Thema braucht Ihr noch den Windows Editor, wenn Ihr mit einer älteren D3Edit-Version arbeitet; in den neueren gibt es den Button M, der auch während des Bauens Meldungen über Vorgänge ausgibt.

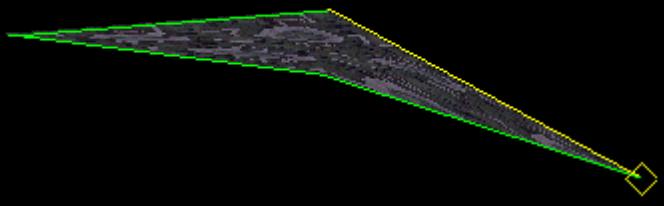
Ich habe hier mal ein paar Bilder vorbereitet, die ein Non-Planar Face zeigen.







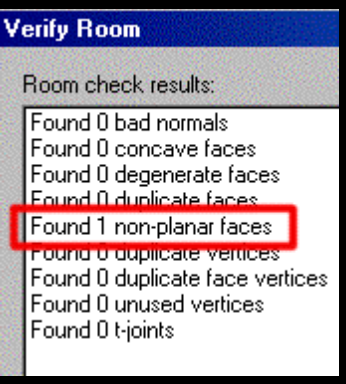
Stellt Euch eine Fläche mit vier Verts vor. Diese Verts sollten auf der selben Ebene liegen. Bewegt man einen dieser vier Punkte und die anderen nicht, dann hat man ein Non-Planar Face.



Es gibt nun zwei Möglichkeiten diese Faces zu finden und zu korrigieren.

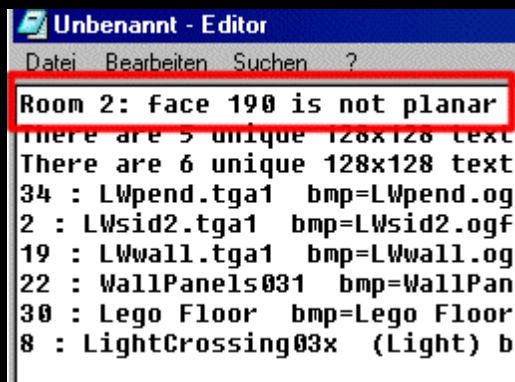
Lösung 1: Wechselt in den Raum in dem Ihr Non-Planar Faces habt. Nun wechselt in den Face Modus und markiert alle Faces. Wenn Ihr das gemacht habt, dann geht in die **Geometry Bar** und klickt auf diese Schaltfläche: Falls es in dem Raum ein Non-Planar Face gibt, dann klickt in der folgenden Dialogbox **Ja**. Wenn Ihr mehrere Non-Planar Faces in den Raum habt, müsst Ihr das solange wiederholen bis bei Verify Room keine Non-Planar Faces mehr angezeigt werden.
(**Wichtig !!! Es müssen alle Faces Markiert sein**.)

Lösung 2: Dieser Weg ist etwas umfangreicher. Wechselt zunächst in die World View Ansicht und last dann Verify Mine laufen. (Bild links)

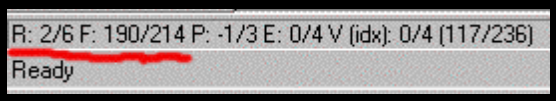


Klickt nun auf die Schaltfläche **Close**. Jetzt öffnet den Windows Editor und wählt im Menü bearbeiten, einfügen. Was Ihr nun seht, müsste etwa wie im Bild rechts aussehen.

Was sagt uns der rot umrandete Teil? Von Raum 2 ist das 190ste Face Non- Planar. Wir gehen jetzt in D3Edit in die World View Ansicht.

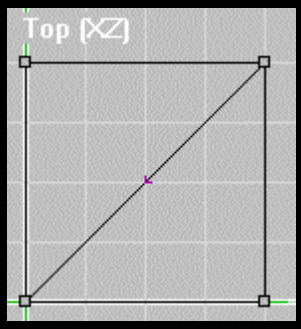


Achtet auf die Statuszeile (Bild unten). Drückt nun die Taste **R** um zu den Room 2 zu wechseln.



(R:2/6 sagt uns Raum 2 von 6 ist der aktuelle Raum) Jetzt drückt die Taste **F** auf Eurer Tastatur, drückt sie solange bis Ihr bei der gewünschten Face Nummer angelangt seid. In meinem Beispiel 190. (F:190/214 sagt uns Face 190 von 214 ist aktuell) Wechselt nun in die Raumansicht und markiert das Face (**Space**). Jetzt klickt Ihr auf diese Schaltfläche:

Achtet darauf, es sind unterschiedliche Schaltflächen !!! Mit der anderen könnt Ihr auch Planare Faces triangulieren, könnt es ja mal ausprobieren. Ok, das Face hat sich nun in zwei Dreiecke (zwei Faces) geteilt. (Bild rechts)



Last nun noch einmal Verify Mine laufen und Ihr werdet sehen, es gibt keine Non-Planaren Faces mehr (wenn Ihr nur ein Non-Planares Face gehabt hattet). So müsst Ihr mit allen Non-Planaren Faces verfahren.

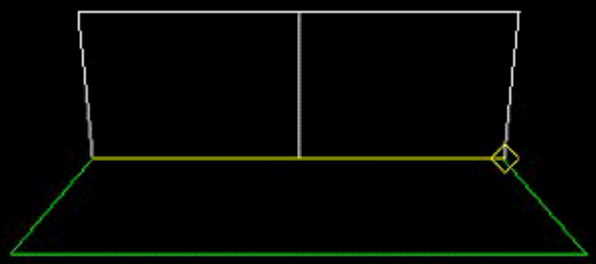
Viel Spass beim Level bauen !!!

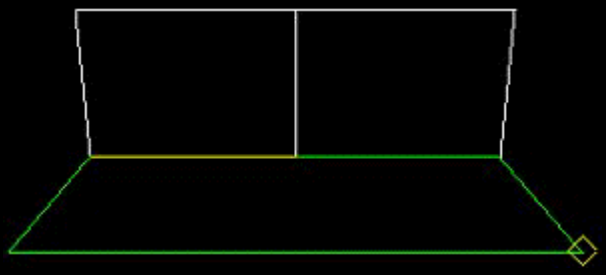
052 - T-Joint

Fischlein

Ich möchte in diesem Thema einmal erklären, wie man T-Joints findet und korrigiert.

Was sind eigentlich T-Joints ?
Das sind Verts, die in keiner Verbindung mit dem benachbarten Face bzw. Edge stehen. Im Bild rechts seht Ihr so einen T-Joint.

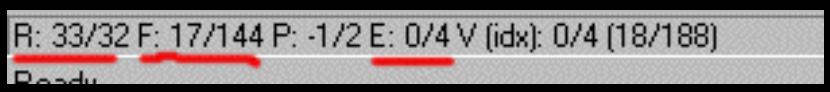




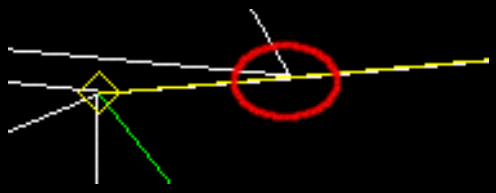
Die gelbe Linie ist das Edge, aber diese müsste aussehen wie im Bild links.

Ok, um nun die T-Joints zu finden, geht in das Menü File/Verify Mine (World View Ansicht). Klickt dann auf **Close**.

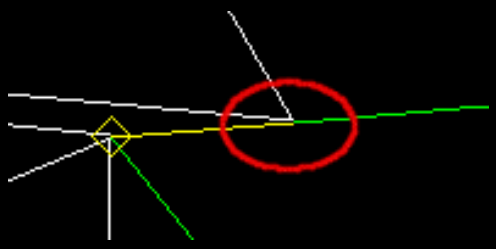
Öffnet nun den Windows Editor und wechselt in das Menü Bearbeiten und dann einfügen. Jetzt seht Ihr alle Detail über Eure Level. Sucht nach einer Zeile die in etwa so aussieht: "`Room 33: face 17 edge 0 has a T-joint`", das sagt uns das Raum 33, Face 17, Edge 0 einen T-Joint hat. Wechselt nun wieder zu D3Edit und drückt die **R** Taste, drückt sie solange, bis in der Statuszeile (siehe oben) die entsprechende Raum - Nummer angezeigt wird. Für Face Taste **F**, für Edge Taste **E**. Die folgende Angabe ist auch noch zu beachten "`Room 33 face 17: No connection for edge 0`", diese gehört mit zu der oberen und sagt uns das in Raum 33, Face 17 keine Verbindung mit Edge 0 hat.



Nun wechselt in die Room View Ansicht und zoomt das Face in der Vorschau heran. Schaut Euch die Kanten des Faces genau an, speziell da, wo ein Nachbarface angrenzt. In meinen Beispiel sieht das wie im Bild rechts aus.



Jetzt klickt das benachbarte Face an und wählt mit der Taste V das Vert, das keine Verbindung mit dem benachbarten face hat, markiert dieses Vert und wechselt zurück zu dem Face mit dem T-Joint, wählt wieder das Edge bei den der T-Joint ist und klickt auf diese Schaltfläche: - das Resultat seht Ihr im Bild unten.

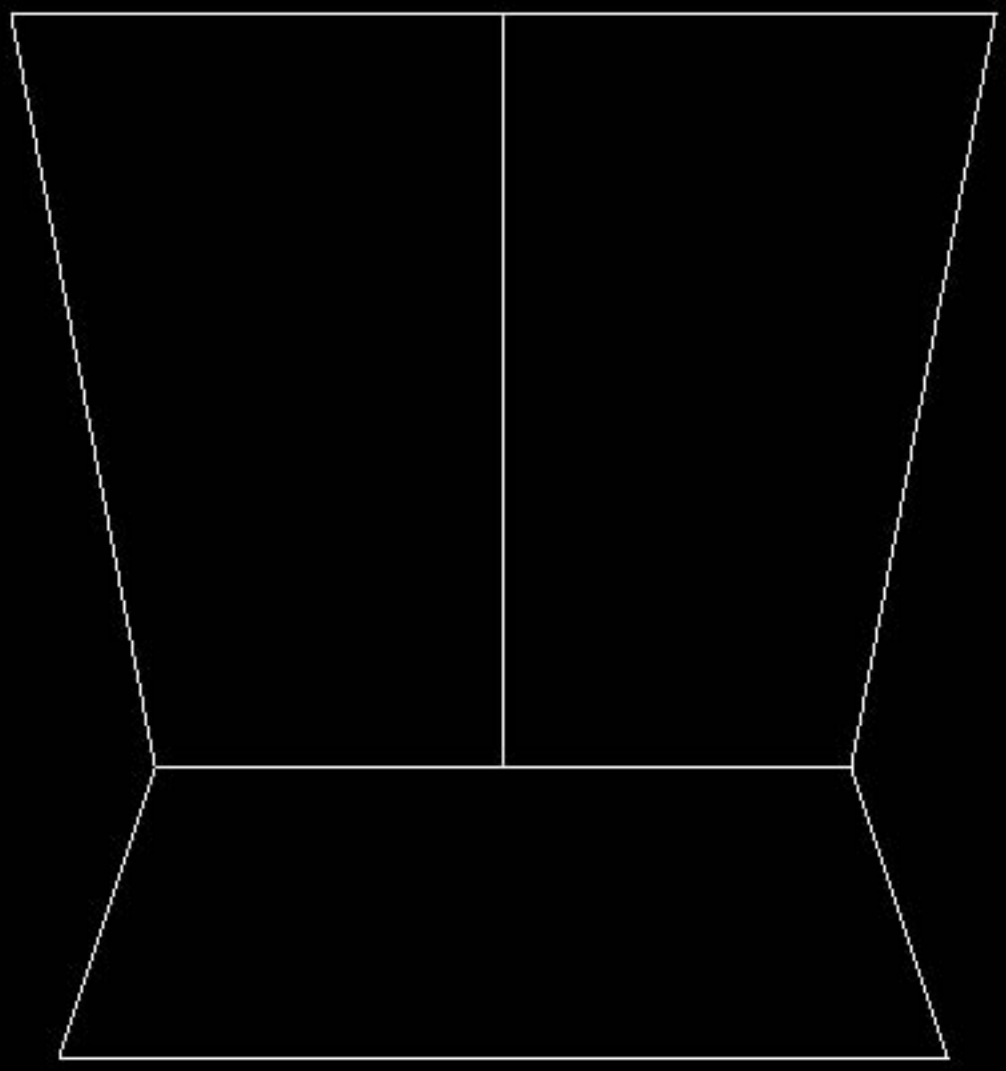


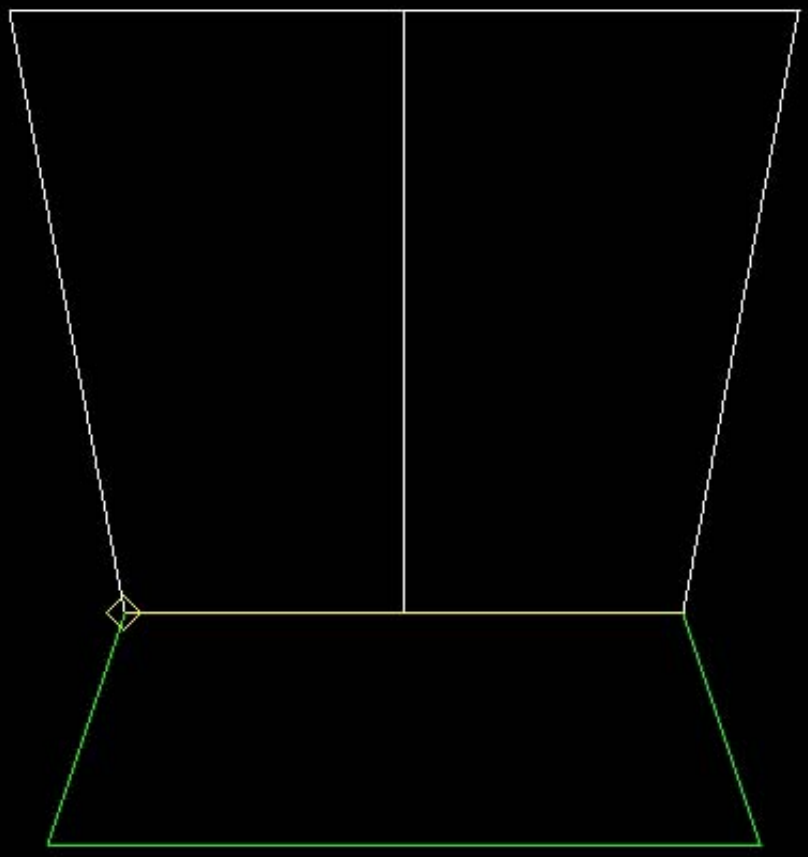
Wenn Ihr jetzt noch einmal File/Verify Mine laufen lasst, werdet Ihr sehen, daß ein T-Joint weniger ist. Wiederholt diese Schritte bis Ihr keine T-Joints mehr in Eurem Level habt.

053 - T-Joints: Zweite Version

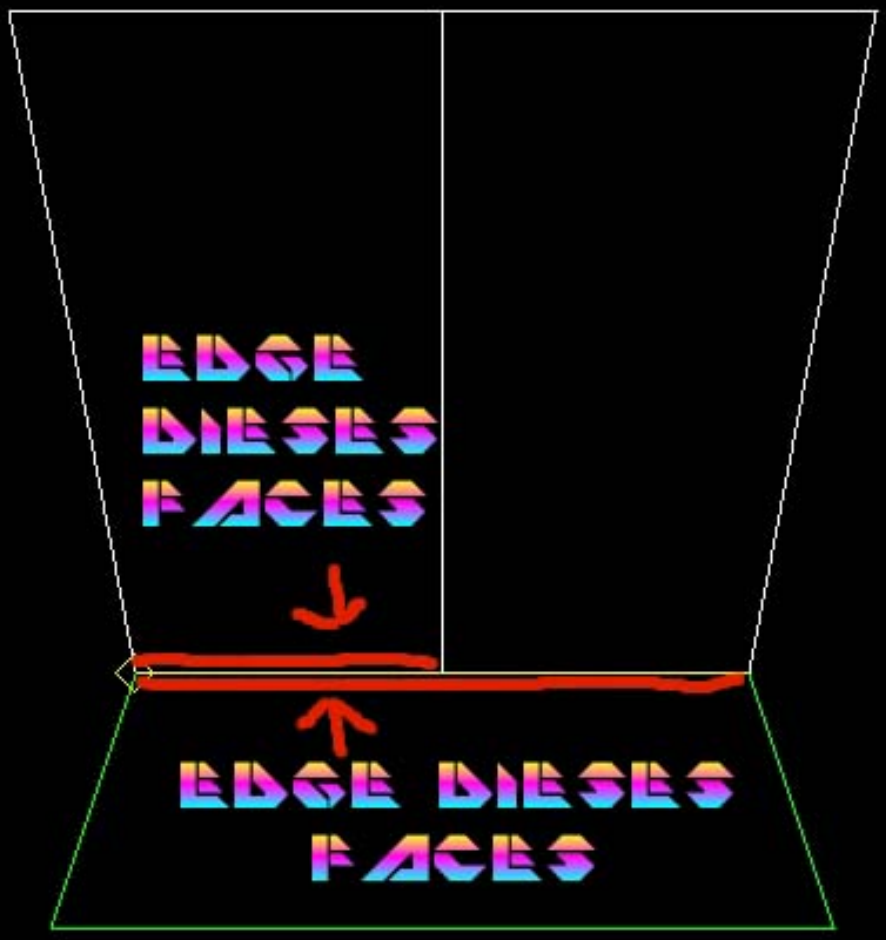
The Edge

So hier noch ein Tutorial zum Ausbessern von T-Joints.
Wir haben dann eine Situation, in der 1 Face auf nur einer Edge
Kontakt mit 2 Edges der beiden benachbarten Faces hat:



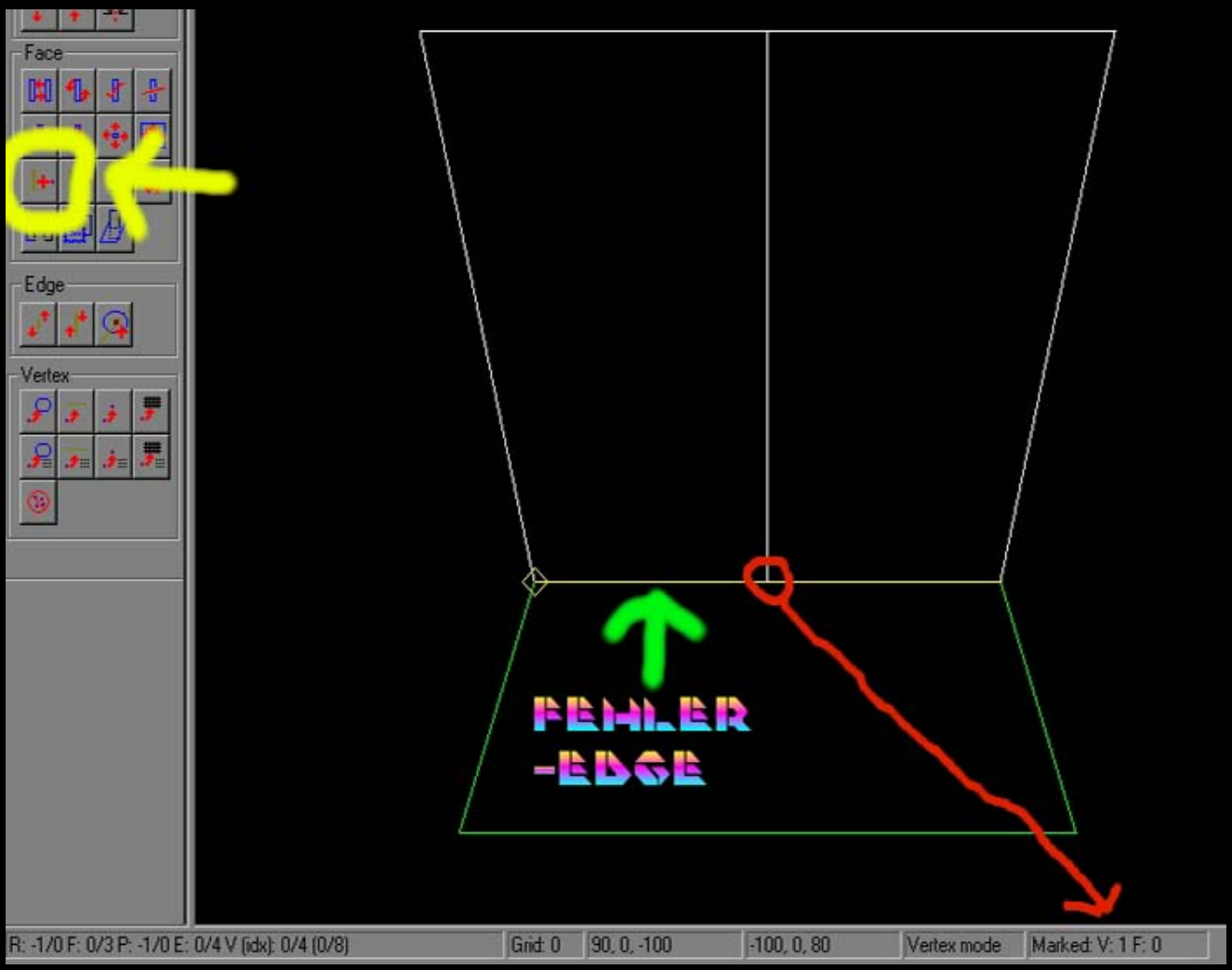


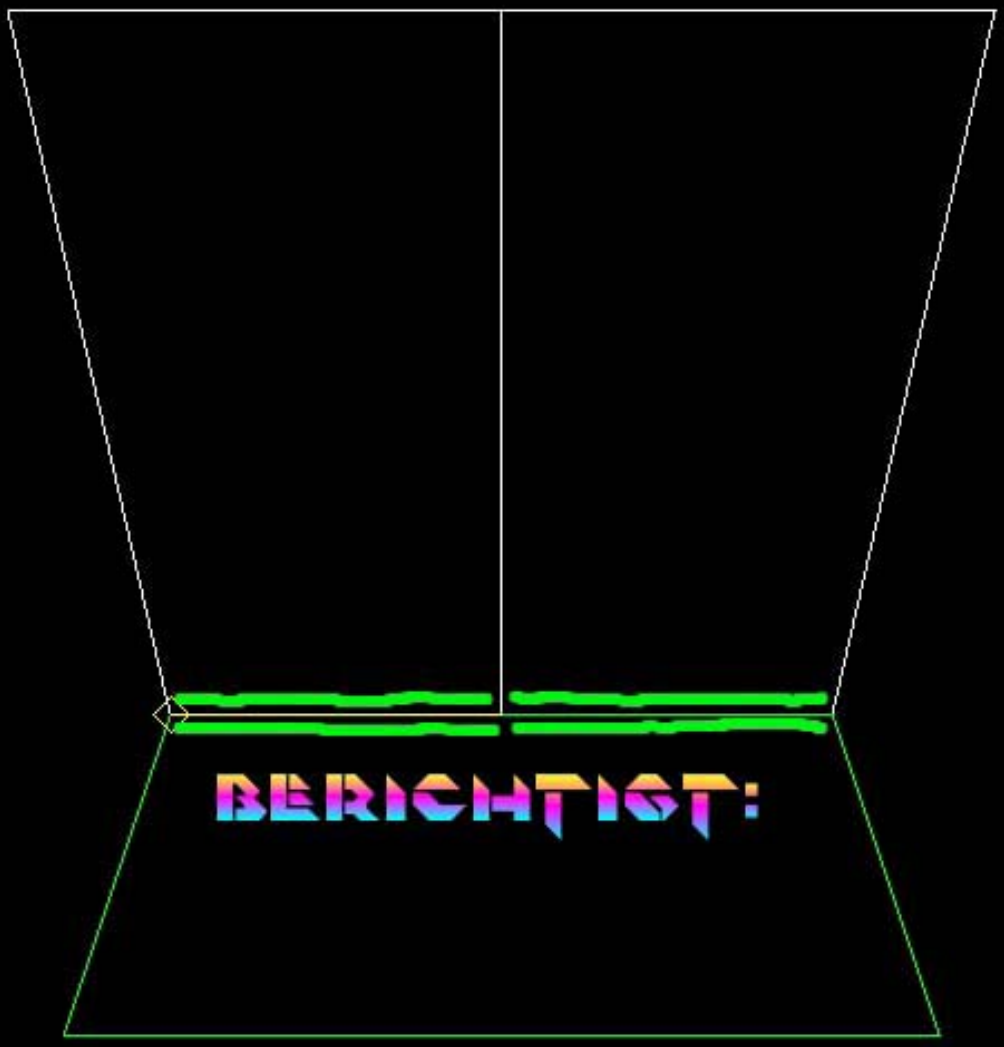
So ist die 100%ige Übereinstimmung nicht gegeben:



Um diesen Fehler zu beheben, wähle den Vert der den T-Joint zum Teil verursacht an (roter Kreis) und markiere ihn mit **M**:

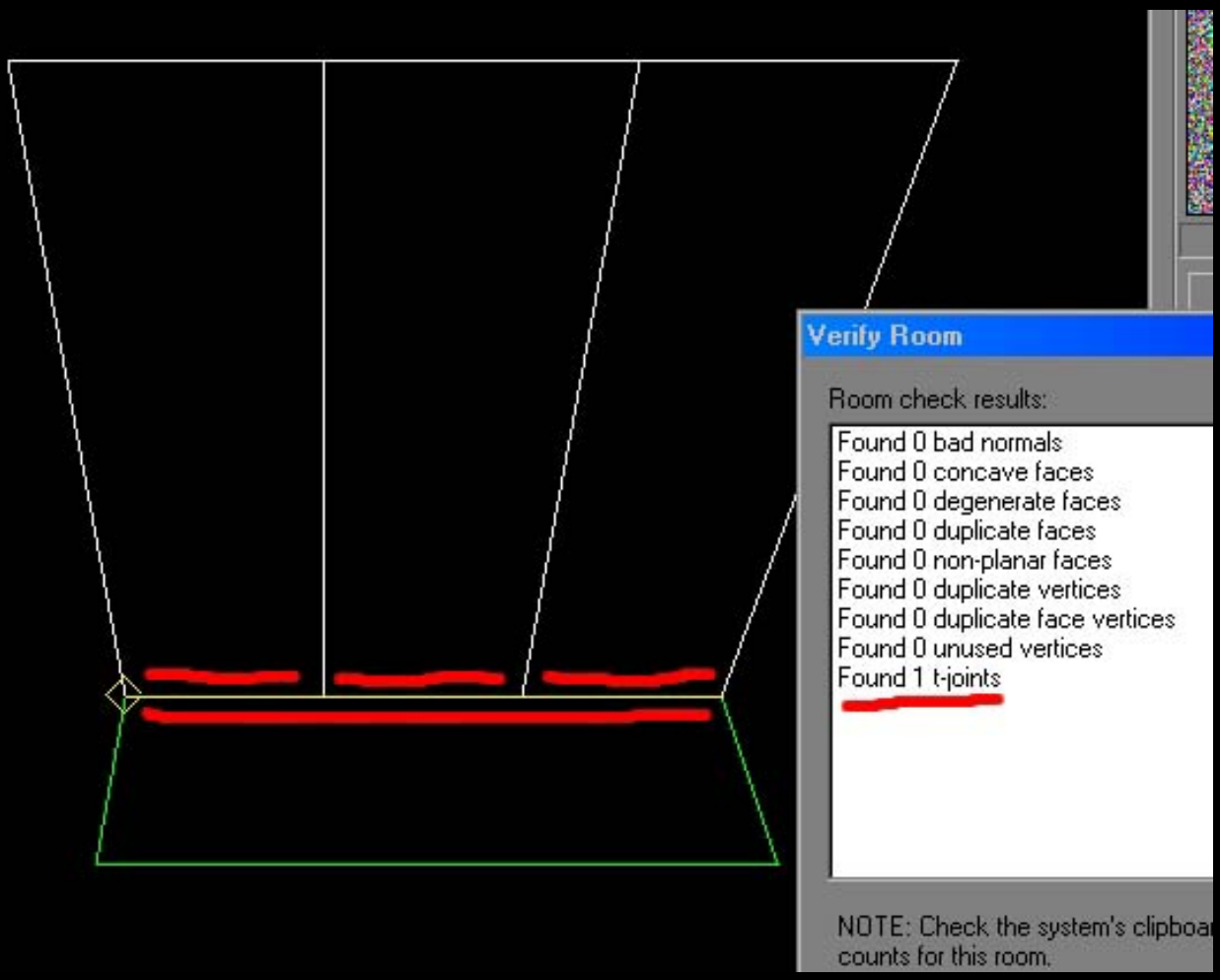
dnan klicke auf das angrenzende Face (hier z.B. unten) und wähle die lange Edge mit **E** an, die den Fehler auch mitverursacht. dann klicke auf den Button **Add Vert** (gelb markiert)





Auf diese Weise teilst Du die Fehler-Edge und zwei Andere, die mit den oberen genau übereinstimmen:

BTW: auch wenn eine Edge mit drei, vier, fünf (usw.) anderen zusammentrifft, zeigt D3Edit nur einen T-joint an:



054 - Verifiziere Dein Level

Kyouryuu

Einleitung

Seien wir ehrlich. Sei es Chaos, Murphy's Gesetz oder wie auch immer Du es nennen willst, an einem bestimmten Punkt geht etwas schief. Sicher, vielleicht zielst Du nicht darauf ab, aber es wird unweigerlich passieren, und Du wirst gezwungen sein mit den Nachwirkungen und der Reparatur umzugehen. Descent 3-Levels sind keine Ausnahme von diesem Phänomen. Früher oder später wirst Du herausfinden daß das Level das Du gebaut hast irgendwelche Fehler in sich hat. Dieses Tutorial wird Dir helfen zu verstehen, wie Du Dein Level verifizierst und die Bugs reparierst.

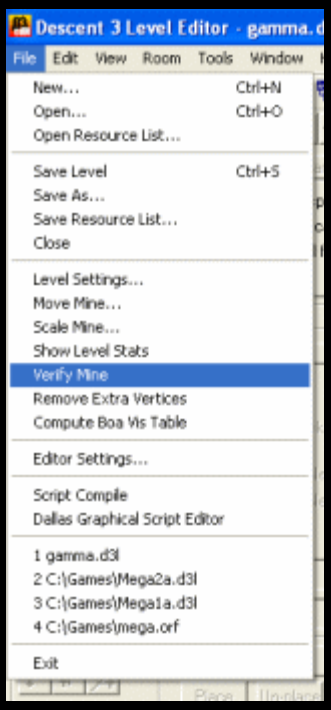
Warum würde ich mein Level verifizieren wollen?

Den Level so fehlerfrei wie möglich zu machen ist immer eine gute Sache. Fehler können es dem Spieler erlauben jenseits der Wände des Levels zu kommen und das ‚Nichts' zu betreten. Die Wände verhindern, daß der Spieler in das Nichts hinaussehen kann und diesen Hall-of-Mirrors-Effekt bekommt. Nicht zu erwähnen daß du die Geschwindigkeit und Performance des Levels insgesamt steigerst. Und es ist gutes Karma – Du willst Dinge richtig erledigt haben nicht nur vom Standpunkt des Gameplay aus, sondern genauso auch vom Technischen.

Aber Outrage hat's nicht immer gemacht!

Stimmt, aber nur weil PTMC Proving Grounds (Retribution Level 12) drei konkave Faces und 62 Bad Shells hat heißt das nicht daß Du Dich nicht drum sorgen mußt. Es sollte klar sein, wieauchimmer, daß es ein paar Levels gibt (vorwiegend Single Player) die beinahe unmöglich fehlerfrei zu kriegen sind. Viele von Outrage's Tür-Modellen, zum Beispiel, kommen vorausgestattet mit attraktiven T-Joints und anderem nervigen Zeug. Du kannst die reparieren, die Du willst... aber manchmal macht es mehr Probleme als es bereinigt. Für die Akten, beinahe keiner dieser Fehler wird Dir dein Level wirklich killen oder Descent 3 abstürzen lassen. Allerdings, aus Performancegründen solltest Du die Anzahl an Fehlern so gering wie möglich halten.

Verify Mine aufrufen

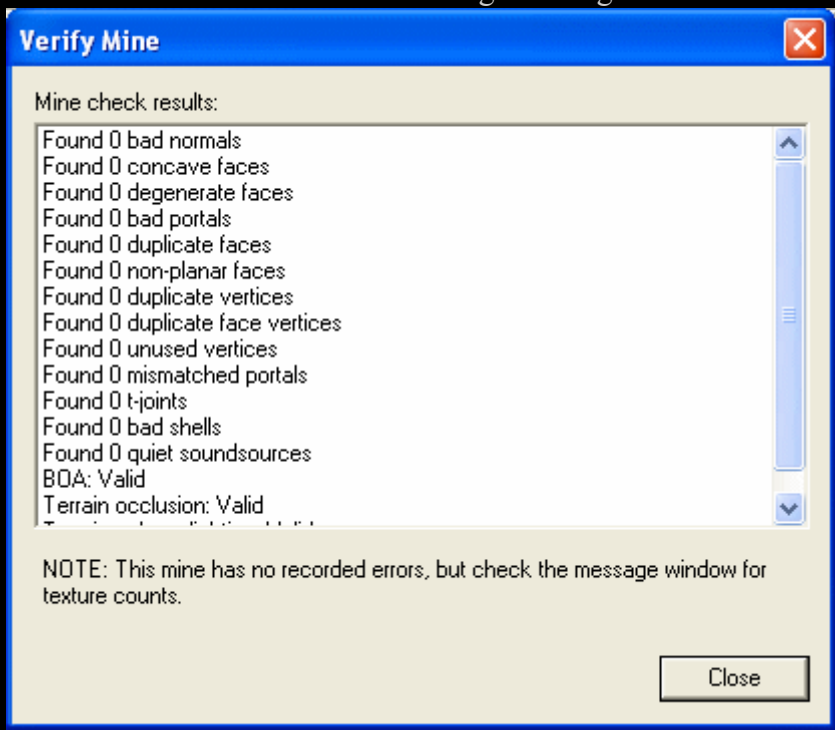


ist einfach. Wenn Du in der World View bist, enthält das File-Menü die Option Verify Mine, wenn Du in der Raumansicht bist enthält es die Option Verify Room.

Sie machen grundsätzlich dasselbe. Allerdings enthält der Verify Mine-Dialog mehr Informationen und deswegen werden wir uns darauf konzentrieren.

Den Output von Verify Mine verstehen

Wenn du Verify auswählst, poppt ein kleines Fenster auf. Es wird einige wichtige Informationen über Dein Level anzeigen. Das folgende ist ein Idealszenario, wo alles perfekt ist. Du solltest darauf abzielen, daß Verify Mine dieses Bild liefert. Dies hier ist der Verify-Output von Reactor Gamma.



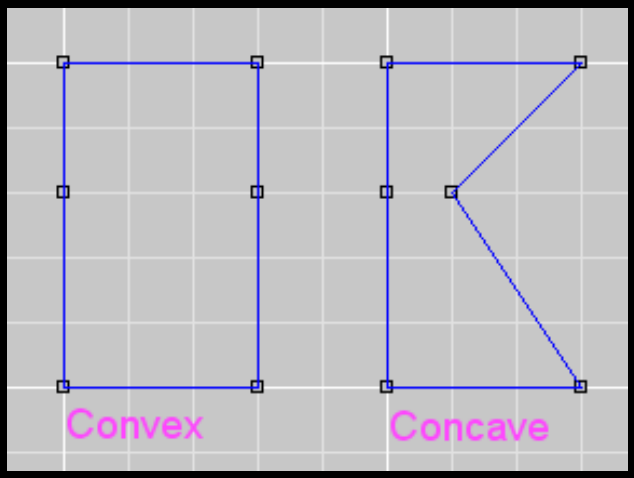
Zusätzlich wird spezifische Information über die Fehler und wo sie sind in die Zwischenablage kopiert. Du kannst das in Notepad oder irgenein anderes Textprogramm einfügen, um sie zu sehen. Oder, Du kannst die neue Message-Bar in D3Edit verwenden (später mehr). Laß uns herunterbrechen was jeder Fehler bedeutet.

Bad Normals

Eine Bad Normal ist eine Face-Normale, deren Richtung nicht oder nicht mehr bestimmbar ist. Dies kommt vor, wenn Faces zu Linien oder gar Punkten deformiert werden; Gerade oder Punkte haben keine Normalen, nur Flächen (nein Körper auch nicht, nur deren Oberflächen ☺)

Konkave Faces

In Descent 3 (und den meisten wenn nicht allen Level-Bau-Tools für diese Materie) müssen Faces immer konvex sein. Grundsätzlich denke von einem Face wie von einer Pizza. Sie ist rund, ganz und gut. Dann, wenn ich ein Stück der Pizza genommen habe, ist sie konkav. In anderen Worten, konvex bedeutet, daß jeder Punkt in einem Face jeden anderen Punkt erreichen kann, ohne die Begrenzung des Faces zu verlassen. Zum Beispiel:



Die Reparatur ist üblicherweise eine einfache Affäre. Du kannst entweder Split Face verwenden um das konkave Face zu zerschneiden, oder Du kannst das Face mit Triangulate Current Face in Dreiecke zerlegen. Das bringt einen anderen guten Aspekt – ein Dreieck kann niemals konkav sein. Jede Form danach, ein Rechteck, Fünfeck, Sechseck etc. kann das. So nebenbei, dieser Begriff von ‚konkav' und ‚konvex' ist auf das Leveldesign beschränkt. Sie sind Übersimplifizierungen der eigentlichen mathematischen Konzepte.

Degenerate Faces

Das ist ein Face, das Dimensionen eingebüßt hat. Mit anderen Worten, es ist keine 2d-Figur mehr sondern entweder eine eindimensionale Figur wie eine Gerade, oder eine nulldimensionale Figur wie ein (mathematischer) Punkt. Sie kommen am häufigsten vor, wenn Du Vertices nach einer Extrusion herumbewegst und Du mit ‚zusammenfallenden' Faces endest. Du kannst sie löschen um das Problem zu lösen, aber sie auszuwählen kann eine echte Schweinsarbeit werden, da sie – nach allem – nur mehr Linien oder Punkte sind. Du kannst die Faces in einem Raum mit **F** durchlaufen, oder du machst **Rechtsklick** in der World View und wählst Select Face By Number. Die Nummer bekommst Du in der Message-Bar (später mehr).

Bad Portals

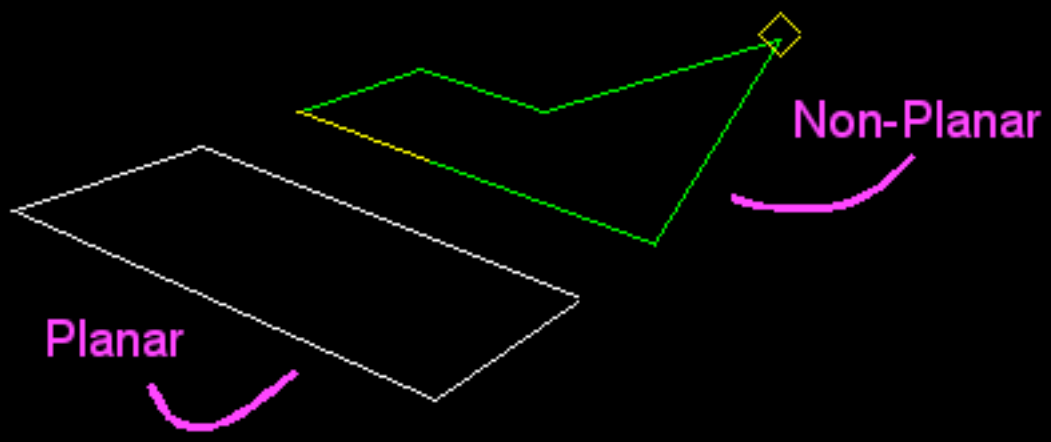
Wird üblicherweise dadurch verursacht, wenn die zwei Faces die das Portal ausmachen nicht exakt abgeglichen sind. In anderen Worten, ein Face hat vielleicht einen Extra-Vertex drin. Oder vielleicht sind sie nicht mehr abgeglichen, weil Du sie in der Raumansicht bewegt hast. Oder, schlimmster Fall, die Faces sind weit davon entfernt aneinanderzupassen . Sorgfältiges Planen wird das Geschehen dieses Fehlers verhindern. Wenn er passiert, wirst Du das Portal löschen wollen und das fehlerhafe Face oder Faces erneut erzeugen, sodaß das portal ordentlich gemacht werden kann.
(Manche Türen können diesen Fehler hervorrufen)

Duplicate Faces

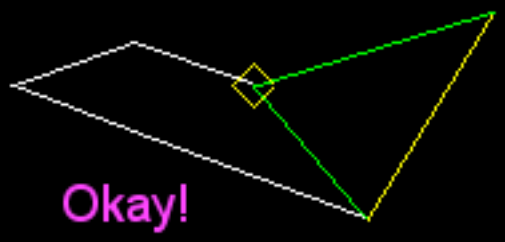
Dieser ist recht offensichtlich. Zwei Faces Existieren in der selben Ebene, teilen sich die selben Sets von Vertices und deren Normals schauen in die selbe Richtung. Oder in kürzeren Worten, Du hast ein Face auf einem anderen drauf. Für gewöhnlich sind sie offenkundig in den texturierten Ansichten, da Du die Texturen auf den fehlerhaften Faces einander ‚bekämpfen' sehen kannst, wenn du Dich in der Ansicht bewegst. Du kannst Doppelte Faces ‚sicher' löschen.

Non-Planare Faces

Der Begriff des nonplanaren Faces ist ähnlich demjenigen von konkaven Faces, nur betrifft dieser hier die Ebenheit, nicht die Form. Ein Beispiel:



Obwohl es ein bischen schwer zu erkennen ist da hier kein Gitter ist, von oben geseen sind die Vertices dieser beiden Formen in der selben Position.Trotzdem, von der Seite, einer der Punkte ist nach oben bewegt worden. In diesem Fall wurde die Planarität verletzt. Ein planares Face ist eins wo alle seine Vertices auf der selben Ebene liegen. Das heißt, ein planares Face von der Seite betrachtet sollte komplett flach sein. Viele nicht-planare Faces werden auch als konkave Faces gezählt. Die Reparatur ist wieder ähnlich, obwohl Du es einfacher finden wirst ein Face zu splitten, anstatt es zu triangulieren. Ein Beispiel:



Duplicate Vertices

Dieser ist selten aber kommt vor. Grundsätzlich, nimm an Du hast ein Dreieck aber anstatt daß es drei Vertices hat, hast Du vier - das heißt, ein Vertex sitzt auf einem anderen. Um das zu reparieren, kannst Du das betroffene Face auswählen und mit **V** durch die Verts durchlaufen. Wenn Du den Punkt erreichst wo zweimaliges Drücken von **V** scheinbar nicht zum nächsten Vertex weitergeht, dann weißt Du daß du den doppelten Face-Vertex gefunden hast. Verwende **Remove Vert** um sie zu eliminieren.

Unused Vertices

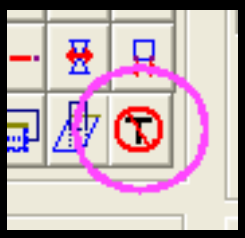
Sie sind nicht gefährlich, aber sie sind wie Sägestaub am Boden des Schneideraums - du wirst eventuell staubsaugen wollen. Glücklicherweise, dieser ist unter den am einfachsten zu beseitigenden Fehlern. Drücke einfach **Remove Extra Vertices**. Fertig.

Mismatched Portals

Sind wie Bad Portals, nur daß ich denke daß hier eine Betonung darauf liegt, ein Ersatz-Vertex oder Vertices zu haben. Das passiert im Allgemeinen wenn Du ein Portal machst und später die Faces an einer Seite des Portales veränderst (z.B. ein Vert auf einer Seite des Portalfaces hinzufügst, aber nicht auf der anderen). Diese sind beträchtlich einfacher zu reparieren. Das Portal neu zu formen, indem man das aktuelle Portal löscht und dann Join Rooms benutzt löst für gewöhnlich das Problem.

T-Joints...

...waren ein Alptraum zu reparieren. Glücklicherweise, die letzten Entwickler-Versionen von D3Edit (speziell 8J) haben ein Tool im Geometrie-Paneel das den T-Joint eliminiert, wenn Du das Face current machst, welches den T-Joint enthält.



Ein T-Joint ist typischerweise definiert als ein Ort im Level, wo ein Face der Raum-Shell nicht bündig zu einem Anderen ist, was ein Loch (eben ein 2d-‚Linien'-Loch) ins ‚Nichts' hinaus produziert. Oder, in gebräuchlicheren Worten, wenn ich mein Level mit Wasser fluten würde, wären T-Joints die Stellen wo das Wasser heraussickern würde. Visuell, das todsichere Zeichen eines T-Joints in einem Level ist eine dicker-als-übliche Linie im Wireframe-Mode. (konnte ich mit v039 nicht reproduzieren, aber man sieht im Wireframe Mode einen Unterschied zu normalen Edges) T-Joints sind ein üblicher Gegner in beinahe jeder Level-Bau-Software (speziell nervig in UnrealEd2). Beachte, daß Räume die das External Flag aktiviert haben immun gegenüber T-Joints sind.
Daher sind Außengebäude und -strukturen von T-Joints freigestellt und Du mußt Dich damit nicht herumärgern.

Bad Shells

Jeder Raum in D3Edit besteht aus zwei Haupt-Typen von Faces – Shell-Faces und Detail-Faces. Shellfaces haben typischerweise Normalen die nach innen zeigen und die äußere ‚Schale' des Raums definieren. Wenn Du Dich umblickst wo Du gerade sitzt (vermutlich in irgendeinem Raum), wären die Shell-Faces die Wände des Raumes - die Wände zeigen nach innen. Dann sind Detailfaces jene Faces, deren Normale typischerweise nach außen zeigen und im Raum existieren. Das sind Sachen wie Lichtaufhängungen, und der Monitor auf dem Du diesen Text gerade liest. Eine Bad Shell kommt vor, wenn die Shellfaces nicht ‚luftdicht' sind. Die üblichen Verdächtigen sind T-Joints oder doppelte Vertices. Gucke in die entsprechenden Stellen, um dem Abhilfe zu schaffen.

Quiet Soundsources

Soundsources sind eine Rarität in selbstgebauten Levels, aber sie können eine Menge an Atmosphäre bringen, wenn sie korrekt eingesetzt werden. Grundsätzlich sind sie unsichtbare Objekte in einem Level die einen von Dir spezifizierten Sound abgeben, welchen Du in ihren Einflußbereich fliegst. Letztenendes sollten Wasserfälle wie Wasserfälle klingen. Eine Soundsource ist allerdings zwecklos, wenn kein Sound mit ihr assoziiert ist. Eine Quiet Soundsource ist so eine. Sie kann gelöscht werden, oder Du ordnest ihr einen Sound zu.

BOA

Die ist entweder `Valid` oder `Not Valid`. Die BOA Vis Table ist eine Notwendigkeit für D3-Levels. Sie sagt dem Spiel wie Räume in den Speicher zu laden sind, basierend auf dem Aufenthaltsort des Spielers, optimiert dadurch die Level-Performance weil nicht der gesamte Level auf einmal geladen werden muß. Es gibt aber auch andere Vorteile. BOA Vis wird automatisch berechnet, wenn Du die Beleuchtung machst, wenn D3Edit eine Änderung in der Levelgeometrie bemerkt oder wenn eine BOA erst noch berechnet werden muß. Du kannst Das BOA-Berechnungstool ohne die Levelbeleuchtung starten, indem du den Button **Vis** im oberen Bereich deines Schirms drückst. Du mußt keine neue BOA berechnen, außer Du veränderst die Levelgeometrie in irgendeiner Weise. Das Spiel regt sich dann wegen einem BOA-Fehler auf, wenn es den Level lädt, also hast Du keine Entschuldigung dafür es zu vergessen.

Terrain Occlusion

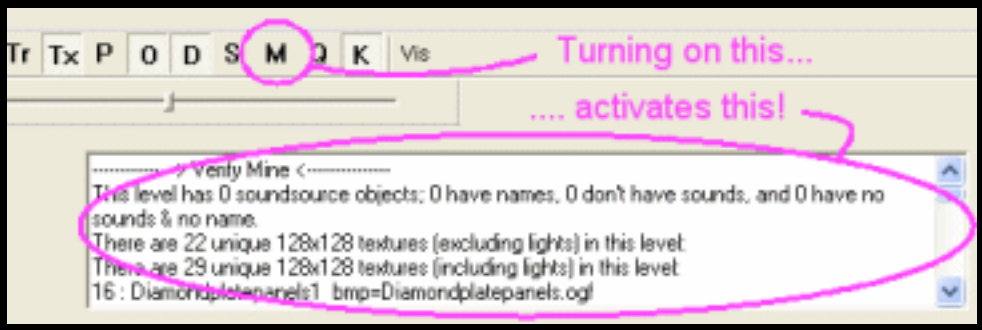
Entweder `Valid` oder `Not Valid`.Dieser Fehler ist irrelevant, außer Du kannst den Outdoor-Bereich in Deinem Level sehen. Wenn Dein Level vollständig Indoor ist und Du niemals auf das Terrain hinausziehst, brauchst Du Dir darum keine Sorgen zu machen. Wenn Du das Terrain-Paneel öffnest (**Tr**), drücke Calculate Terrain Occlusion. Dieser Prozeß kann (wird) einige Minuten dauern. Ich bin mir nicht ganz sicher was Terrain Occlusion ist – obwohl es ohne Zweifel etwas mit der Optimierung der Performance zu tun hat, wenn man draußen ist.

Terrain Volume Lighting

Lies das als „Hast Du die Beleuchtung fürs Terrain berechnet?“. Wieder ist es entweder `Valid` oder `Not Valid`. Ebenso ist dies wieder irrelevant wenn das Terrain nicht genutzt wird oder nicht vom Level aus gesehen werden kann. Gehe in den Beleuchtungsdialog und wende **Dynamic Light** und **Light it!** an. Beachte, daß Du nur ein paar Iterationen zur Lichtberechnung des Terrains brauchst (ich mache 500 aber nur weil ich das kann). Konvergenz ist nicht zwingend.

Die Messagebar verwenden um Fehler zu identifizieren

Dank der Kooperation zwischen Nirvana und OtherOne haben die neuesten Versionen von D3Edit (besonders 8J) die Message Bar. Drücken von **M** läßt ein Fenster im unteren Bereich Deines D3Edit-Fensters erscheinen.



Wenn Du Dein tapferes Verify Mine mit der Message Bar erweiterst, wirst Du unverwundbar sein. die Message Bar zeigt an, was Verify Mine bisher immer in die Zwischenablage kopiert hat. Hierbei mußt Du nicht mehr zwischen zwei Applikationen hin und her springen, um die Fehler zu lesen. die Message Bar sagt Dir genau wo jeder Fehler ist, normalerweise als erstes den Raum dann das Face, dann die Kante oder das Vertex. Lern das! Es wird Dir das Leben wesentlich erleichtern.

055 - Verifiziere Deine Mine

WillyP

Du brauchst hierzu die neueste Version von D3Edit (v39). Du brauchst auch Dein bereits existierendes, Über-L33t, aber leicht kaputtes Level. Und einen Zaubertrank, gemacht aus drei Teilen Ausdauer und drei Teilen Fleiß ☺

Beachte: Vert, Verts, Vertex, Vertices... alles dasselbe.

www.descentbb.net
Englisch-sprachiges Descent-Forum
www.descentforum.de/forum/
Deutsch-sprachiges Descent-Forum

beide haben eine Abteilung für Levelentwickler.

Eine Fehlerfreie Mine Bauen

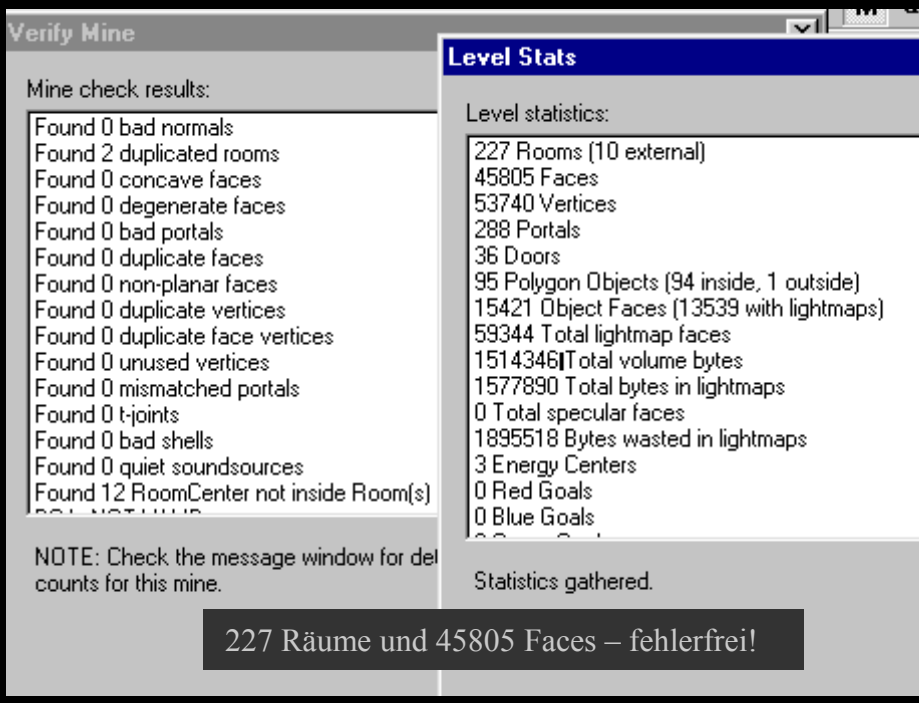
Es ist nicht soooooo schwer... wirklich!

Warum sich das antun?

Einfachgenommmen, eine dürftig konstruierte Mine wird sich nicht gut spielen. Sich verschiebende Texturen und Faces, verschwindende Faces, Spieler außerhalb des Levels, das sind einige Dinge die Du möglicherweise in so einem Level erfährst. Aber um die Wahrheit zu sagen, ein paar Fehler bringen Dir Deine Mine nicht um. Die Outrage-Levels haben einen Haufen Fehler, T-Joints, Bad Shells etc....
Für gewöhnlich sind diese Probleme einfach zu finden und zu beseitigen, aber es braucht Geduld und Beharrlichkeit! Es gibt keinen Grund nicht eine fehlerfreie Mine zu haben, Wenn Du auf Fehler triffst für die Du scheinbar keine Lösung findest, frag nach Hilfe (Kasten oben rechts).

Woher kommen Fehler?

Versuch's wie Du's willst, manche Fehler sind unvermeidbar. Die meisten Funktionen die wir verwenden um Räume und Level zu bauen können Fehler erzeugen, sogar die einfachsten Operationen wie etwa das Bewegen einiger Verts, können eine Vielzahl an Fehlern produzieren.



227 Räume und 45805 Faces – fehlerfrei!

Wann sollten diese Fehler repariert werden?

So bald wie möglich! Es ist niemals eine gute Idee Fehler zu Fehlern hinzuzufügen. Die meisten Fehler können beseitigt werden sobald sie auftauchen. Wenn Du einen Raum separat von Deiner Mine erzeugt hast, verifiziere ihn bevor Du ihn anhängst. Wenn Du vertrauter mit den Prozessen bist, wirst Du davon eine Ahnung bekommen welche Operationen wahrscheinlich Fehler produzieren und welche nicht.

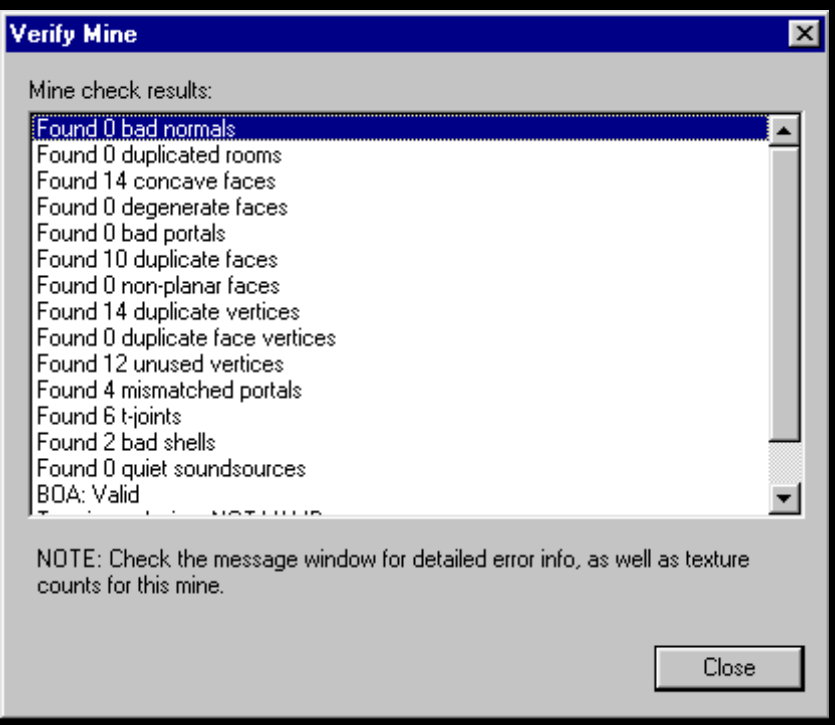
Kann jeder Fehler beseitigt werden?

Ich glaube schon. Aber es sollte erwähnt werden, daß einige ‚Fehler' die Verify berichtet nicht wirklich Fehler sind. Zum Beispiel, daß bei externen Räumen berichtet wird, daß das Raumzentrum nicht im Raum ist. Und wenn Du einen Raum auf dem Terrain machst, dann binde all dessen Faces an einen externen Raum, Verify wird das als doppelten Raum berichten.
Wenn eine Mine gebaut wird, läßt es sich nicht vermeiden Fehler einzuführen. Aber wir *können* und *müssen* diese entfernen!

Eine kleine Mine mit typischen Fehlern

Der erste Schritt ist es Verify laufen zu lassen. Öffne Deine Mine, und in der World View wähle Verify Mine aus dem File-Menü. Oder wähle Verify Room, wenn Du an einem Raum arbeitest.

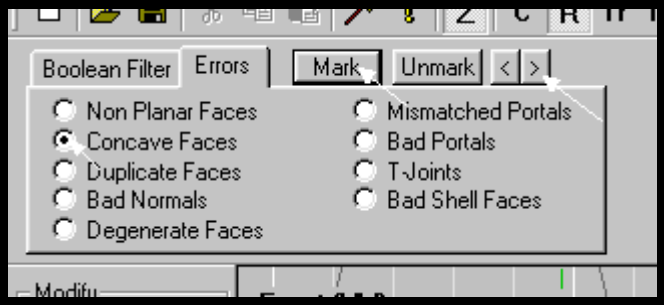
Verify berichtet



Hier ist Verify's Bericht. Es sind hier keine Bad Normals oder Duplicate Rooms, aber wir werden sie trotzdem behandeln. Drücke den Button M. Dies öffnet die Message Bar. In diesem Fenster ist zusätzliche Info von Verify. Es sagt uns welcher Raum Face und Vertex den Fehler hat. Rechtsklicke in der World View, dann wähle Select Room by Number, und gib die Raum-Nummer aus der Message Bar ein. Klicke OK um zu diesem Raum zu gelangen. Dann kannst Du Dein fehlerhaftes Face oder Portal auch per Nummer auswählen, und beide Kameras, in World View und und Current Room View, zoomen zum betroffenen Face. Dieses herzige Tool (Danke, Atan!) funktioniert auch für Objekte.

Nun laß uns besprechen, was die Fehler sind und wie man sie repariert.

Aber zuerst ein anderes gutes Tool...



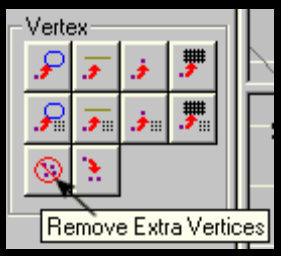
Dieses hier ist unter dem Button K versteckt. Du mußt in einer Raumansicht sein. Wähle die Errors-Karte und wähle den Fehler auf dessen Vernichtung Du abzielst. Dann drücke Mark. Ist das kool? Es wird noch bessr, siehst Du die kleinen Buttons < und >? Drücke > und das erste markierte Face wird zum current Face! Glatt! Drück ihn nochmal! Aah! Schlagt mich!... Äh, ich meine laß uns weitermachen...

Gängige Prozeduren

Ein paar Tastaturkürzel

Um zum nächsten Raum weiterzugehen drücke **R**, **Shift-R** für den vorigen. Geht in der Room oder World View. Um Faces zu durchlaufen, in der Room View, verwende **F** oder **Shift-F**. Durchlaufe die Kanten des current Face mit **E** oder **Shift-E**. Durchlaufe die Verts des current Face mit **V** oder **Shift-V**. Verwende **P** und **Shift-P** für Portale.

Remove Extra Verts

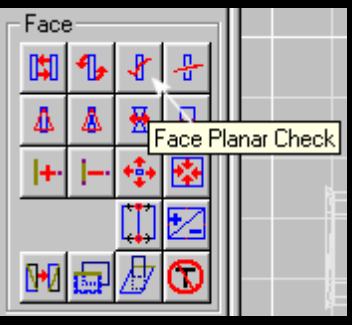


Mehr als ein Vert kann denselben Platz okkupieren. Aber wenn diese zu Shellfaces gehören, würde das als Loch in einer Shell betrachtet werden. Es ist einfach genug, Extra-Verts zu beseitigen, klicke einfach den Button **Remove Extra Verts**. Viele Reparaturen können Extra-Verts hinterlassen. Und Extra-Verts zu entfernen kann auch alles sein, was gebraucht wird um einige Dinge zu reparieren. Wenn also Verify sagt daß da Extra-Verts sind, entferne die als Erstes.

Laß Verify nochmals laufen

Nachdem Du Reparaturen gemacht hast, solltest Du Deinen Raum nochmals verifizieren, und wenn du mit dem Raum fertig bist, verifiziere die gesamte Mine. Um ein Auge auf deinen Fortschritt zu halten und die Message Bar zu updaten.

Non-Planar Faces



Stell Dir ein Blatt Papier auf Deinem Schreibtisch vor. Alle vier Ecken sind auf derselben Ebene. Hebe eine Ecke hoch, und das Papier wird gebogen. Es wird ‚non-planar' Descent kann nicht-planare Faces nicht behandeln, und wird unerwartete Ergebnisse zeigen. Nun stelle Dir eine diagonale Falte im Papier vor, so daß drei Ecken am Schreibtisch bleiben während eine angehoben wird. Das Papier wurde in zwei Sektionen aufgeteilt, jede ist planar. In unserem Level können wir ein Face in kleinere aufsplitten, die planar sind. Ein Face das nur drei Verts hat wird immer Planar sein.

Hast Du die non-planaren Faces markiert, kannst Du **Face Planar Check** verwenden um die Faces zu splitten. Das ist der schnelle und einfache Weg, aber Du wirst die neuen Faces re-texturieren müssen. Du kannst Faces auch manuell splitten. Markiere zwei Verts in einem Face. Der nächste Button rechts ist **Split Current Face**. Verwende das, um die Art und Weise zu kontrollieren wie Faces gesplittet werden. Zum Beispiel, um Symmetrie zu wahren.

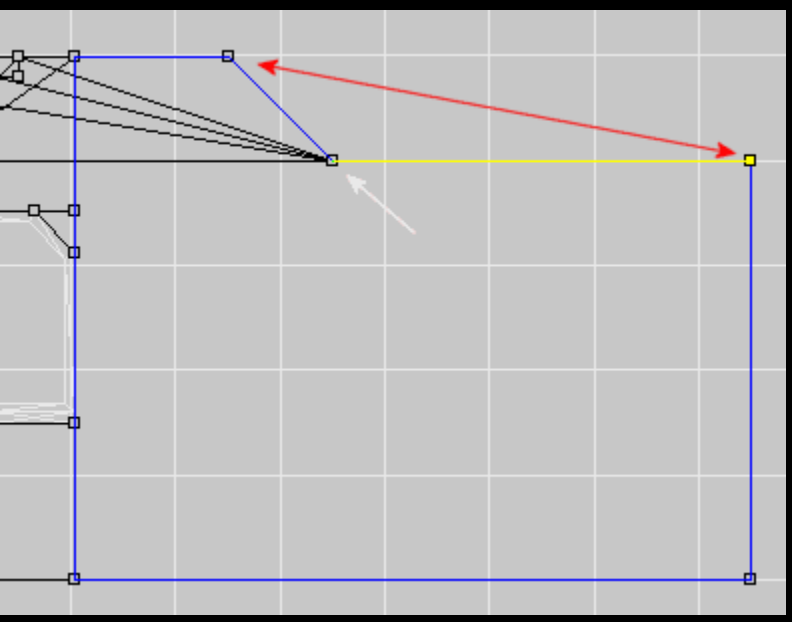
Ein anderer Weg um wieder ein planares Face zu bekommen ist, dessen Verts an das Gitter zu bringen, dann sie auf die selbe Gitterlinie zu bewegen. Verwende **Snap Marked To Grid** nachdem Du die Verts markiert hast, oder Du kannst auch Snap **Current Vert To Grid** verwenden. Diese Buttons finden sich in der Vertex-Sektion

Noch ein anderer Weg um non-planare Faces zu korrigieren ist sie zu löschzen und neue Faces zu bauen. Das ist manchmal der leichteste Weg um eine spezifische Geometrie zu bekommen.

Konkave Faces

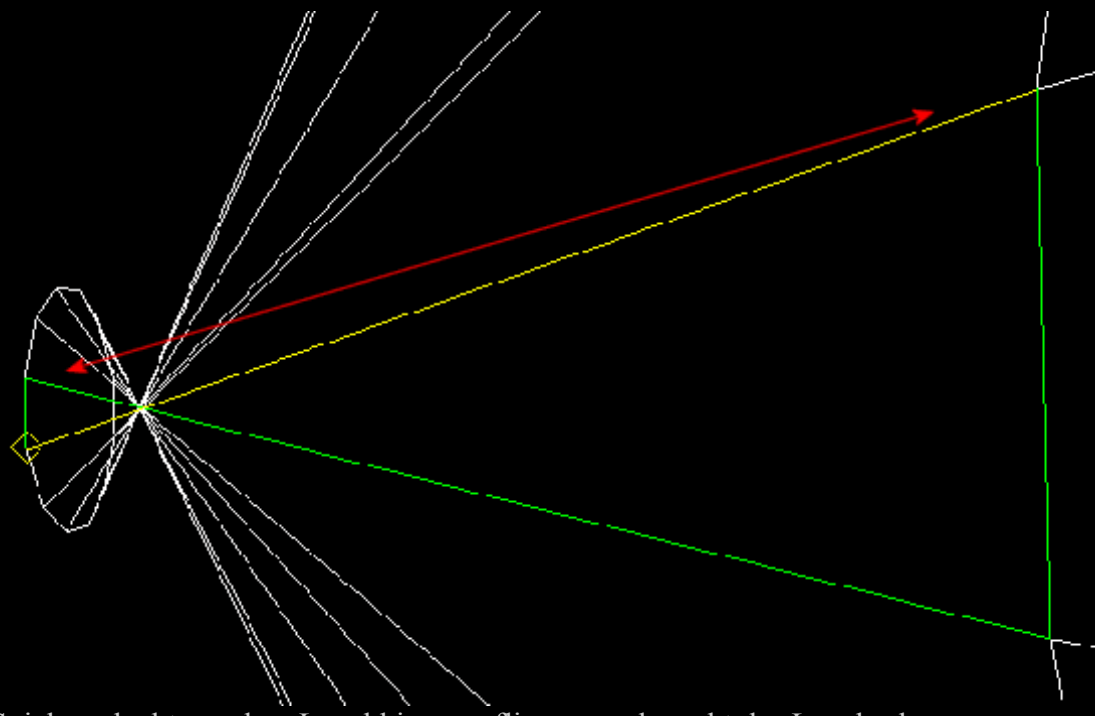
Beachte, Du kannst den Toleranzlevel für Konkavität im Einstellungsdialog Deines Editors setzen. Eine niedrigerer Nummer könnte mehr Faces detektieren. Bei der niedrigsten Einstellung, 0.0010, magst Du ein wenig mehr Arbeit zu machen haben, aber Du wirst mit einer besseren Shell enden.

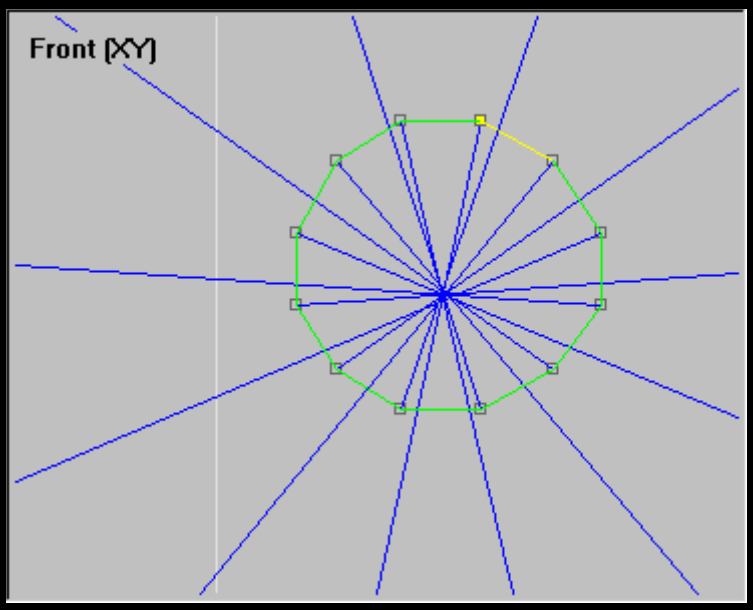
Wenn Du eine Linie zwischen irgendwelchen zwei Vertices auf einem Face ziehen kannst und diese außerhalb des Faces ist, hast Du ein konkaves Face. Der beste Weg damit umzugehen ist das Face zu splitten. Verwende das Mark-Tool um das konkave Face zum Current zu machen. Schau in die Message Bar, um zu sehen welches Vert die Konkavität verursacht falls Du Dir nicht sicher bist. Markiere dieses Vert und ein Anderes (aber nicht das Nächste), und drücke Split Current Face. Ein Face das nur drei Vertices hat wird niemals konkav sein. Manche Konkavitäten sind schwer zu sehen, aber wenn Du diese Prozedur anwendest kannst Du einfach alle Konkavität aus Deinem Level entfernen. Wie bei non-planar, Du magst es manchmal vorziehen das Face zu löschen und Neue zu machen.



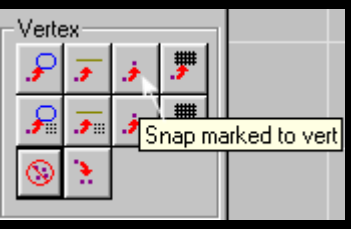
Hier links ist eine Ansicht eines Faces, das konkav ist. Die Rote Linie die ich zwischen zwei Verts eingezeichnet habe ist ganz klar außerhalb des Faces. Der weiße Pfeil zeigt zum Vert, das konkav ist. Ich *könnte* das Vert einfach nach oben bewegen, aber das würde die Architektur des Raumes ändern. Und es könnte mehr Fehler verursachen. Es ist einfach genug, das konkave Vert zu markieren und ein Anderes, um das Face zu splitten.

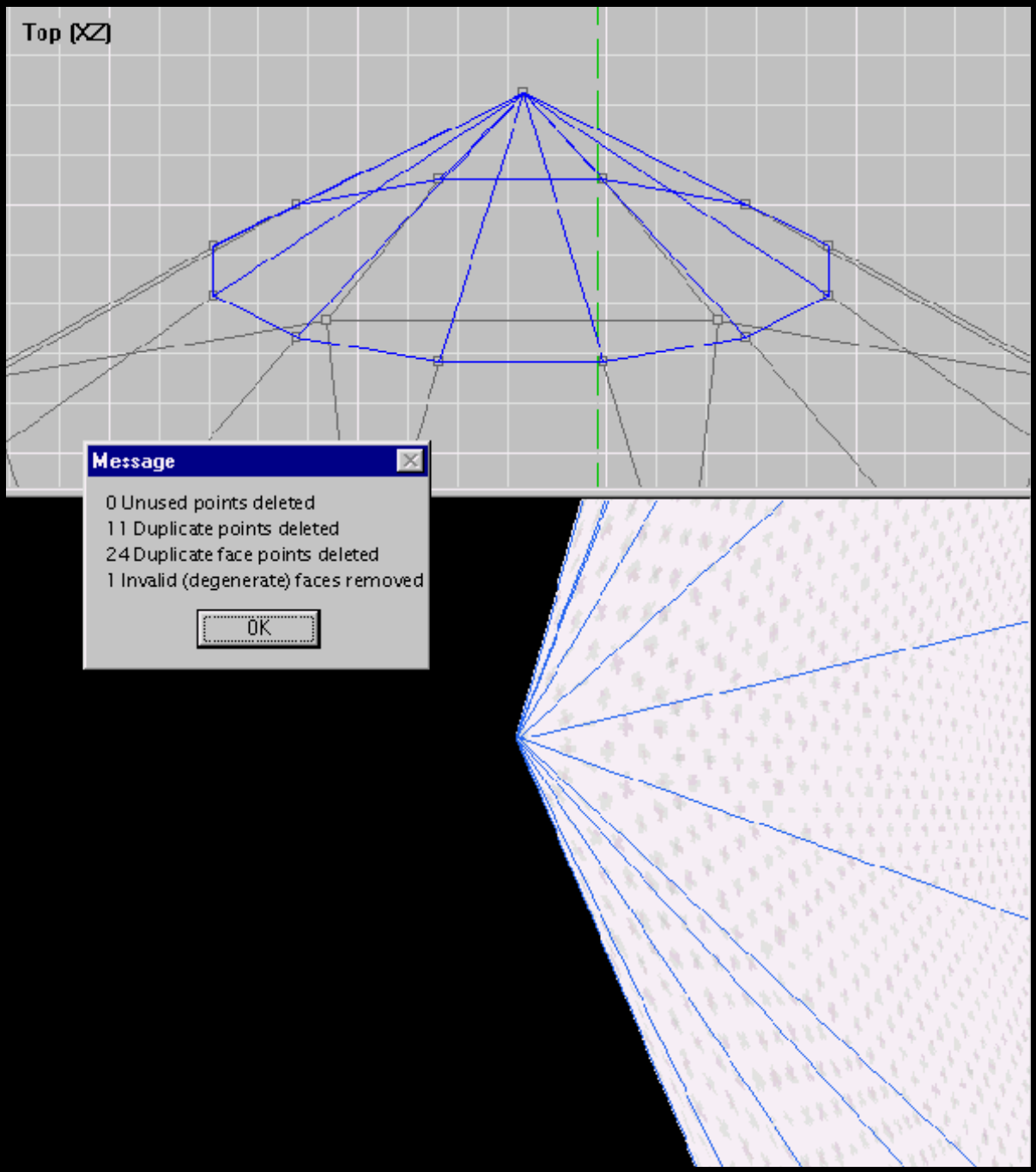
Hier ist ein anderes Beispiel eines konkaven Faces: Dieses hat vier Verts. Die current Edge kreuzt die gegenüber liegende. Alle vier Verts sind auf der selben Ebene, also ist das Face planar. Ich habe eine rote Linie zwischen zwei Verts eingezeichnet. Dies ist eine einzigartige Situation, in der das Face in sich selbst verdreht ist. Hier sind eigentlich zwölf Faces im selben Dilemma gezeigt. Dieser Fehler ist so ernst daß er es dem Spieler erlaubt aus dem Level hinauszufliegen, und macht das Level sehr laggy. Da die Shell nicht ordentlich definiert ist, gibt es keinen Weg zu wissen wann man den Spieler stoppen soll. Ich habs nicht versucht, aber ich glaube, wenn ich versucht hätte alle zwölf Faces einfach zu splitten wie weiter oben beschrieben, würde ich ein Durcheinander aus konfusen Faces erhalten und möglicherweise andere Fehler. Aber ich habe eine andere Idee.





In dieser Ansicht können wir sehen daß hier ein zwölf-seitiges Face (wäre das nicht ein dutzend-eckiges Face?) an unseren Problemfaces hängt. Im Vertex Mode werde ich mit dem Lasso alle seine zwölf Verts markieren. Stelle sicher, daß eins von ihnen current ist (klicke auf das Face um es current zu machen falls es das nicht ist). Ich drücke **Snap Marked To Vert**. Zwölf Verts, alle auf dem Platz von einem. Und die Fläche des zwölf-seitige Face wurde zur Größe eines Punktes reduziert (ein Degenerate Face). Die Kanten der konkaven Faces, welche in Gemeinschaft mit denen des zwölf-seitigen Faces sind wurden in ihrer Länge auch null. Sie alle verschwinden und werden zu einem Vert, wenn wir den **Remove Extra Verts** drücken.





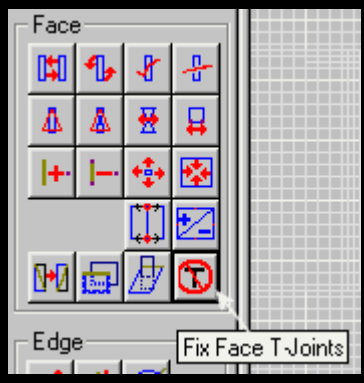
Duplicate Faces, Bad Normals, Degenerate Faces

Wenn zwei identische Faces den selben Platz okkupieren, sind sie Duplicate Faces. Markiere sie mit dem Mark-Tool und lösche sie. Mission abgeschlossen. Böse (Bad) Normals sind keine guten Normals. Degenerate Faces waren gute Faces, aber sie haben harte Zeiten erlebt; es gibt für beide wenig bis keine Hoffnung. Du kannst sie packen schicken, mit der alten Mark-Delete-Rebuild-Routine. Manchmal funktioniert auch, ein markiertes Vert auf ein anderes zu snappen; dann geht auch **Remove Extra Vertices**, obwohl es sein kann, daß du diese als erstes erwischt hast. Obwohl, nicht gerade ein kleiner Katerschluck!

Mismatched Portals, Bad Portals

Meistens verursacht, indem man ein Face oder Vert bewegt, das Teil eines Portals ist. Ein anderer Grund ist, das Anhängen eines Raumes an ein Face das an einem Portal anliegt; wenn dieses Face gesplittet wird bekommt das existierende Portal ein Vert hinzugefügt. Dies macht das Portal kaputt. Du kannst versuchen das Portal zu löschen, alle Faces zu verbinden die durch es gesplittet wurden, dann Join Rooms anzuwenden um ein neues Portal zu machen. Oder, Du kannst einen der Räume löschen (zuerst als Raum-Datei speichern!), die Faces wieder vereinigen, dann den Raum erneut platzieren. In Zukunft, wenn Du ein Portal in ein Face einfügen willst daß bereits eins hat, kannst Du versuchen dieses Face derart zu splitten, daß es das existierende Portal nicht beeinträchtigt, dann das neue Portal hinzufügen. Oder noch besser, Baue den Raum mit einer eingebauten Isolation.
(vergleiche Sirians Tuts)

T-Joints



Dieser ist relativ einfach. Ein T-Joint ist eine Verbindung zwischen zwei Faces. Ein Face hat ein Vert in der geteilten Kante, das andere nicht. Es gibt einen Button um T-Joints aus dem current Face zu entfernen. Jetzt solltest Du wissen, wie man das fehlerhafte Face findet und markiert.
Du hast nur ein Vert zur Kante hinzugefügt, der eins abging. Wenn das Face daß Du repariert hast ein Portal ist, hast du auch das Portal zerstört. Schaue in den Absatz über Portale. Viele der Türen die in D3 inkludiert sind haben Fehler, inklusive T-Joints. Wenn Du die Tür löscht und erneut installierst, hast Du dasselbe Problem. Du kannst versuchen das Portal zu löschen, den T-Joint zu reparieren, dann die Räume erneut verbinden.

Wichtig: Lösche niemals eine Tür mit Delete Room. Lösche eine Tür IMMER über das Door-Paneel!

Bad Shell Faces

Diese können die am trickreichsten zu reparierenden sein. Oft kommen sie als Ergebnis von andereren Fehlern und werden repariert, wenn man die anderen Fehler beseitigt. Manchmal resultieren sie aus der Reparatur anderer Fehler. Wenn Du alle anderen Fehler repariert (und **Remove Extra Verts** gemacht) hast aber noch immer einen Bad-Shell-Fehler hast... gut, viel Glück, Du bist auf Dich allein gestellt... nene, nur ein Scherz... Aber... Bist Du Dir sicher daß du alle anderen Fehler gefixt hast? Stelle sicher, daß Du Verify Mine nocheinmal laufenläßt um Gewißheit zu haben. Markiere die Faces unter Verwendung der **Mark**-Bar, und begutachte das fragliche Gebiet sorgfältig. Indem Du die Verts der Kanten durchläufst, die in der Message Bar identifiziert wurden, magst Du eine winzige Lücke finden, die nicht wirklich sichtbar ist... entweder snappe die Verts zusammen oder mache ein Neues Face.

BOA und Beleuchtung

Wenn Du zufriedenstellend alle Fehler entfernt hast, wirst Du die BOA erneut berechnen und die Beleuchtung wieder ausführen müssen. Wenn die BOA nicht gültig ist, wird sie bei Ausführung der Beleuchtung berechtnet.

Abschnitt E – Terrain

...AND THERE IS LIGHT AT THE END OF THE TUNNEL !

(O-Ton Interplay/Outrage, aus dem Promo-Video)

Es ist dringend empfohlen, die Einbindung von Terrain mehrmals zu üben, zumindest solange bis man darin fluent ist. Wird die Verlinkung zwischen Mine und Außenwelt nicht ordentlich gemacht, kann man davon ausgehen daß der Level nicht richtig funktionieren wird.

056	**Terrain-Implementierung**	Erklärung, wie man Terrain macht und bearbeitet	Kyouryuu	214
057	**Außenwelt erstellen**	Ein Schritt-Für-Schritt-Durchlauf, wie man ein Terrain erstellt und einbindet	Fischlein	217
058	**Außenwelt – Update**	Mine und Außenwelt mit dem neuen D3Edit verbinden	Ragil Ral	223
059	**Spielerstart in der Außenwelt**	Verschiedene Möglichkeiten, den Spieler am Terrain starten zu lassen	Ragil Ral	225

Toc

056 - Terrain-Implementierung

Kyouryuu

Einleitung

Eines der Schlüsselfeatures die Descent 3 über seine Vorgänger hinaus anbot, war die Hinzufügung von Außengebieten. Dieser Teil der Fusion-Engine arbeitet unterschiedlich als der Indoor-Teil, den Du höchstwahrscheinlich gewohnt bist zu verwenden. Dieses Tutorial wird Dich Durch die Erstellung einer Terrain-Höhenkarte und ihre Implementation in D3Edit geleiten.

Schnellreferenz

Für die unter Euch, welche bereits wissen wie man mit Terrain arbeitet, hier ist eine kurze Auflistung der Kontrollen, die verwendet werden um mit Terrain zu arbeiten.

Ctrl-T – Schaltet das Terrain in der World View ein/aus.
Linksklick auf Terrainzelle – Selektiert die Terrain-Zelle.
Ctrl-Alt-Shift-Linksklick auf Terrain – ‚Flutfüllung', die alle Terraintexturen der selben Textur zur current Textur macht.
Shift-. (Punkt) – Rotiert die Textur der current Terrain-Zelle 90° im Uhrzeigersinn
Shift-, (Komma) – Rotiert die Textur der current Terrain-Zelle 90° entgegen dem Uhrzeigersinn
Shift-doppelter Linksklick auf Satellit – Ändert die Satelliten-Textur.
Shift-Linksklick auf Skydome – Ändert die Skydome-Textur zur current Textur.
Shift-Linksklick auf Terrainzelle – Setzt die Textur der nächsten Terrainzelle zur current Texture.

Eine Höhenkarte erzeugen

Terrain in Descent 3 wird unter Verwendung einer Höhenkarte definiert, welche mit jedem beliebigen Malprogramm erzeugt werden kann. Grundsätzlich, die Art und Weise wie es funktioniert ist, daß Du ein Graustufenbild erzeugst, das 256 Pixel im Quadrat mißt. Reinweiß (255, 255, 255 in Rot, Grün, Blau) repräsentiert die Maximalhöhe, Vollschwarz (0, 0, 0) die Minimalhöhe. Dazwischen hast Du Deine Hügel und Kliffe. Das Spieler-Schiff kann nicht über Reinweiß fliegen, da dies außerhalb der unsichtbaren Barriere liegt, die auf dem Terrain liegt.
Um ein gutes Terrain zu machen, wirst Du eine Mischung aus grauen und schwarzen Gebieten haben wollen, wo Du willst Daß der Spieler durchfliegen kann, und überall anders Weiß. Bestimmt halte aber die Grenzen des Terrains Weiß, weil sonst der Spieler in das ‚Nichts' außerhalb des Terrains sehen könnte. Ich mag es, den normalen Pinsel zu verwenden, um die Hauptpfade in Schwarz zu zeichnen. Dann habe ich etwas Grau zur Auskleidung dieser Pfade. Schließlich verwende ich die Smudge- und Push-Tools, um die Kanten der Farben miteinander zu verwischen, vielleicht auch noch einen Gaußschen Weichzeichner drüberlaufen lassen. Dies scheint recht natürliches Terrain zu produzieren. Wenn Du fertig bist, speichere Dein Terrain als 256-Farben-Graustufen-`.pcx`-Datei. Stelle sicher, wenn Du die Option hast, daß es eine Version 5 PCX-Datei wird.

Die Höhenkarte in D3Edit importieren

Um das klarzumachen, laß mich auf einen Punkt hinweisen. D3Edit wird dir nicht erlauben, die Form des Terrains an und für sich zu manipulieren. Das einzige, was D3Edit von Natur aus kann, ist das Terrain zu texturieren und zu beleuchten. Alles Andere, daß physisch geändert werden muß, muß über Dein Malprogramm behandelt werden und dann in den Editor zurückimportiert werden. Wenn Du eine `.pcx` re-importierst, verlierst Du Deine Texturinformationen nicht.
Das gesagt, nun kannst Du D3Edit öffnen und ein Level laden, und in die World View gehen. An diesem Punkt schalte Das Terrain-Paneel mit Tr ein, falls Du das noch nicht gemacht hast. Klicke auf den Button Import PCX und navigiere zum Ordner, wo Du Deine `.pcx`-Datei drin hast und öffne sie. Nach einer kurzen Weile sollte Dir D3Edit das Terrain in der Ansicht zeigen. Falls Du es nicht sehen kannst, **Ctrl-T** schaltet das Terrain ein/aus. Im Wireframe Mode wird das Terrain durch ein Gitter aus Punkten repräsentiert. In den texturierten Modes, wenig überraschend, siehst Du das Terrain mit seinen Texturen.

Den Skydome und die Satelliten bearbeiten

Du wirst bemerken, wenn Du die texturierte Ansicht in der World View hast, daß da eine gigantische, halbrunde Form ist, die über dein Terrain reicht. Das ist, natürlich, der Skydome. Dieser enthält auch die Satelliten-Elemente. Diese wiederum sind die kleinen Dekorationen, die Du im Himmel siehst, wie die Sonne, Planeten, Monde und so weiter. Für die meisten Absichten und Zwecke wirst Du zumindest einen Satelliten haben wollen, der eine Lichtquelle ist und der Licht auf das Terrain wirft. Das Problem ist, da sind nicht sehr viele Satelliten-Texturen, die an und für sich Lichtwerte haben. Du kannst eine komplette Auswahl im Texturenselektor finden, unter Terrain. Ich finde, daß MercSun gut als eine Basis-Sonne dient. Um sie anzuwenden, suche als erstes im Skydome nach Deinem Satelliten (leichter gesagt als getan – Du mußt Dich vielleicht über den Skydome bewegen bevor Du ihn sehen kannst). Standardmäßig ist der Satellit eine eher häßliche blaue und pupurne Wand-Textur. unter Drücken der **Ctrl**-Taste auf den Satelliten **doppelzuklicken** wird seine Textur zu dem ändern, was auch immer die Current Texture ist.
Ähnlicherweise kannst Du auch den Skydome editieren, indem Du **Shift** drückst und darauf **linksklickst**.
Optional kannst Du auch mit anderen Einstellungen in der Terrain-Bar herumspielen. Stars beispielsweise platziert kleine Sterne im Himmel, die verschwimmen wenn Du Dich bewegst. Das ist großartig, wenn Dein Skydome nur schwarzer Raum ist, eher als Wolken zu haben oder andere atmosphärische Features. Nach allem, Sterne unter den Wolken sind einfach kein realistischer Anblick. Ein anderer erwähnenswerter Parameter ist das Rot. Deg.-Feld. Was es macht ist die Himmels-Textur graduell um den Skydome herumzurotieren. Eine Einstellung von .50 ist langsam und natürlich. Natürlich, wenn du deinen Himmel rotieren lassen willst wie einen Whirlpool, sei mein Gast.

Dein Terrain texturieren

Ganz wie Deine Höhenkarte besteht Dein Terrain aus einem Gitter von 256 auf 256 Zellen. Das gibt ein Total von 65 536 Zellen insgesamt. Dankbarerweise mußt Du nicht dasitzen und jedes einzelne anmalen. Der hauptsächliche Weg wie Du Terrain texturierst ist daß Du als Erstes die Grund-Textur applizierst, und dann die anderen Texturen sporadisch hinzufügst. Zum Beispiel, wenn dein Terrain ein Schnee-Thema hat, wirst Du als erstes eine grundlegende Schnee-Textur auflegen, und dann andere Typen von Schnee-Texturen da und dort ausgestreut haben um die Monotonität zu durchbrechen. Du texturierst eine Terrainzelle einfach so, wie Du ein Minen-Face texturieren würdest. Wie eingangs erwähnt, **Ctrl-Alt-Shift-Linksklick** auf das Terrain ist wie eine ‚Flutfüllung'. Alle anhängenden Terraintexturen mit der selben Textur werden in die current Texture geändert. Da das Terrain standardmäßig komplett mit einer häßlichen purpurnen Textur belegt ist, ist es einfach das gesamte Terrain anfangs ‚flutzufüllen'.
Manche Terrain-Textur-Schemen, wie die Canyons und speziell das Piccu Station Sand/Gras/Erde-Set, enthalten Texturen die Du drehen mußt. Auf dem Terrain kannst Du texturen nur in 90°-Winkeln rotieren, und es kann ein paar Versuche brauchen, bevor Du lernst welche Textur so rotiert daß sie nahtlos mit der Nächsten anschließt. Rotation wird gemacht indem man als erstes die Terrainzelle auswählt, die Du rotieren willst indem man rechts draufklickt und dann **Shift-.** oder **Shift-,** (Punkt bzw. Komma) drückt, um sie im bzw. entgegen dem Uhrzeigersinn zu drehen.

Terrain Occlusion und Beleuctung

Das ist der einfache Part, obwohl er lange dauern kann. Für die finale Version Deines Levels wirst Du die Terrain-Occlusion berechnen wollen und das Terrain auch ausleuchten. Für das Erstere gehe einfach ins Terrain-Paneel und klicke auf Calculate Terrain Occlusion. Das mag ein Weilchen dauern aber, außer Du ersetzt die PCX-Höhenkarte, mußt Du es nicht nochmal machen. Danach kannst du den Beleuchtungsdialog aufrufen (Window->Lighting) Dann, wenn Du das Terrain beleuchtest, ist Iterations der einzige relevante Faktor. Setze diese Zahl größer als 100 (2000 ist ganz ordentlich für die Finalversion eines Levels). Dann, in der Terrain-Sektion des Beleuchtungsdialogs, klicke auf Light it! Wenn er Dich fragt ob Du Dynamic Lighting danach berechnen willst, laß ihn das machen. Dann lehne Dich zurück und warte.

Du magst vielleicht dumm auf das Fortschrittsfenster glotzen, daß aufgeht –480% bis Konvergenz erreicht ist? Ja, gut – Die Wahrheit ist, daß Konvergenz nicht notwendig ist wenn Du das Terrain behandelst. Es ist nur wichtig wenn du die Mine beleuchtest. Deswegen kommen wir mit so wenig Iterationen davon.

Das Innen mit dem Außen verbinden

Laß uns für einen Momen annehmen, daß Du eine Mine hast und Du diese an die Outdoor-Gebiete anhängen willst. Als erstes mußt Du ein Bauwerk für die Mine erzeugen, daß auf dem Terrain sichtbar ist. Denke davon wie von einer Metapher – Du hast ein Standard-Bürogebäude, daß Dein ‚Level' ist. Die Korridore und Büros entsprechen dem Inneren Deiner Mine. Aber von außerhalb des Gebäudes siehst Du sie nicht notwendigerweise, wegen der ‚Shell' die auf der Außenseite ist. In diesem Schritt mußt Du diese Shell Bauen.

Die Außen-Shell des Gebäudes ist recht simpel. Es ist ein Standard-ORF-Raum, bei dem alle Normals nach außen geflippt wurden (ein ‚umgestülpter' Raum). Ein beachtenswerter Punkt – halte die Basis Deines Bauwerks relativ breit. Du wirst späte sehen warum. Wenn Du ihn in Dein Level setzt, mußt Du in die Room Properties für diesen Raum und External Flag anticken. Indem Du Room View und World View zusammen verwendest, kannst Du Dein Bauwerk recht einfach über das Terrain bewegen. Voila – Gebäude auf der Oberfläche.

Zuletzt mußt Du Löcher in das Terrain stanzen. Wenn das Terrain vom Ort des Spielers aus sichtbar ist und der Spieler in der Mine ist, sieht er das Terrain wie es mitten durch die Mine schneidet, wo diese auf das Terrain trifft. Um diese Blockade zu beseitigen, wähle die einzelnen Terrain-Zellen durch Links-Klick darauf aus und verwende dann **Toggle Vis.** im Terrain-Paneel. Nun siehst Du, warum ich Dich dazu angehalten habe, die Gebäudebasis breit zu halten. Weil Du ein Quadrat auf einmal aus dem Terrain schneidest, ist das ein eher unpräziser Prozeß. Schneide ein wenig zuwenig weg, und Du siehst das Terrain in der Mine. Schneide zuviel weg, und Du hast Löcher rund um Dein Gebäude, wenn Du es von außen betrachtest.

Objekte und das Terrain

Dieser Teil ist ein wenig obskur. Wenn du ein Objekt auf dem Terrain platzieren willst, wähle das Objekt daß Du hinzufügen willst. Im Objekt-Paneel wähle den Terrain-Radiobutton an. Dann wähle die Zelle wo Du das Objekt platzieren willst und klicke auf den Insert-Button während Du in der World View bist. Die meisten der Objekte werden ordentlich auf diese Weise eingesetzt. Einige davon, wie das InvisiblePowerup, werden abhängig von Deiner Kamerapositon eingesetzt, was ein wenig ungeschickt ist. Wenn Du das InvisiblePowerup platzieren willst, füge erst ein anderes Objekt ein und konvertiere das dann in ein InvisiblePowerup.

Wenn Du Objekte auf dem Terrain bewegen willst, wähle als erstes das Objekt durch Rechtsklick an. Dann verwende folgende Kontrollen:

Alt-X – Sperrt die Bewegung auf die X-Achse (links und rechts, Standard)
Alt-Y – Sperrt die Bewegung auf die Y-Achse (rauf und runter, Standard)
Alt-X – Sperrt die Bewegung auf die Z-Achse (vor und zurück, Standard)
Alt-CursorRauf – Bewegt Positiv auf der aktuellen Achse
Alt-CursorRunter – Bewegt Negativ auf der aktuellen Achse
Alt-CursorLinks – Rotiert im Uhrzeigersinn auf der aktuellen Achse
Alt-CursorRechts – Rotiert entgegen dem Uhrzeigersinn auf der aktuellen Achse

Wenn Du in der World View **Rechtsklick**st, kannst Du auch die Inkremente für die Bewegung des Objektes setzen und sehen welche Achse gerade aktuell ist.

File: 057-beisp_ausenw.zip

057 - Außenwelt erstellen

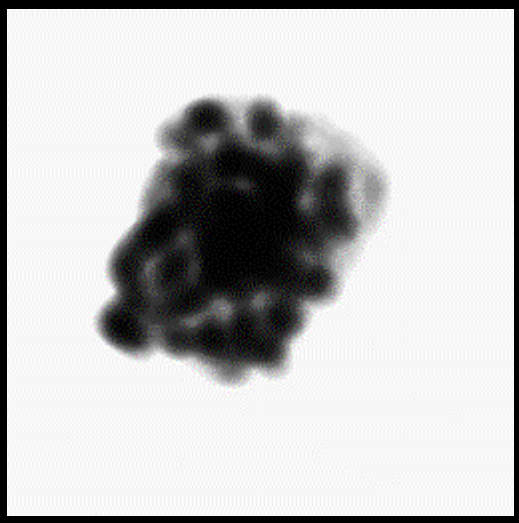
Fischlein

(überarbeitet)

Um eine Aussenwelt zu erstellen, benötigt Ihr die D3Edit Version 1.0 oder höher!

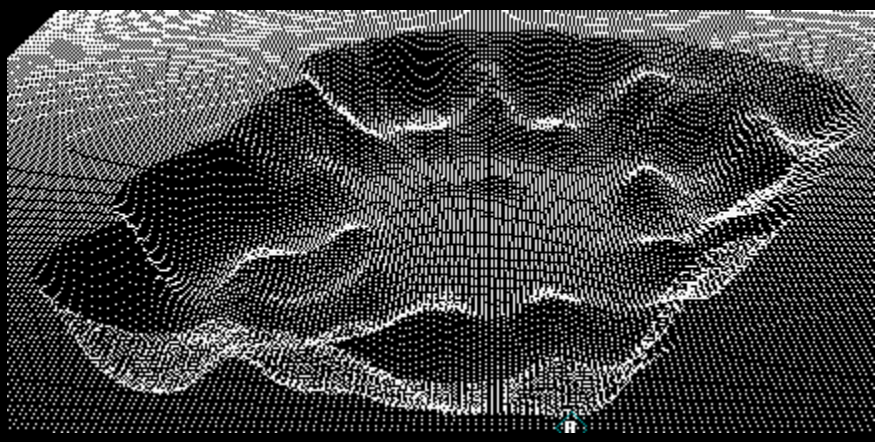
Es ist eigentlich nicht schwer eine Aussenwelt in Eueren Level einzubauen. Ihr braucht ein Grafikprogramm, das das Format **PCX** unterstützt.

Erst einmal etwas zum Verständnis. Ihr erstellt ein Zweidimensionales Bild, jetzt malt Ihr einen Fleck auf das Blatt (mit Schwarzer Farbe), nun habt Ihr die Farben weiss und schwarz wobei schwarz das Tal ergibt und weiss den Berg. D3Edit berechnet die Aussenwelt nach den Farbabstufungen.



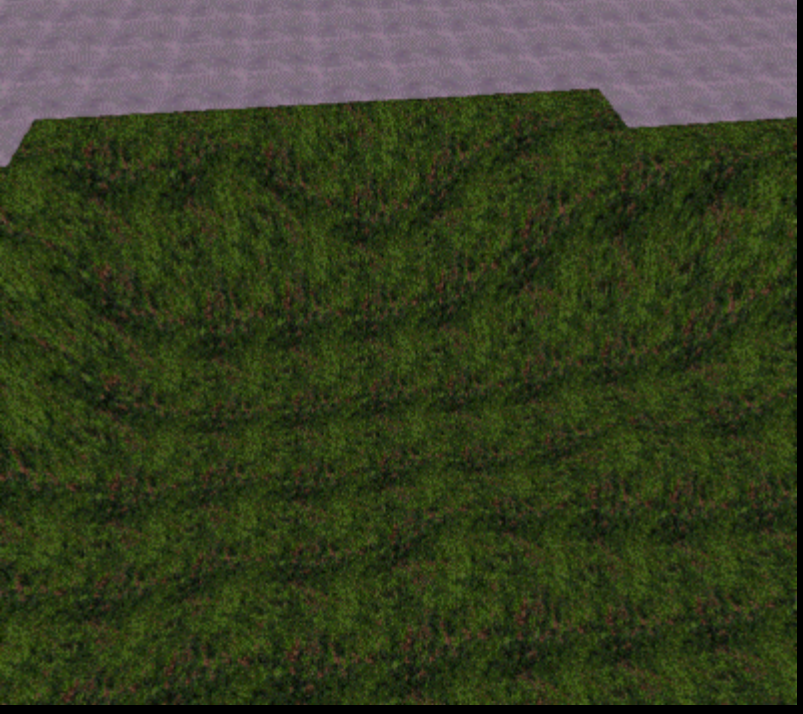
Ich habe einmal ein Beispiel vorbereitet. Erstellt eine PCX - Datei in der Grösse 256 x 256 Pixel in Graustufen (links).

Jetzt müsst Ihr noch das Bild in Euren Level importieren und das geht wie folgt. Öffne D3Edit, lade Deine Level. Jetzt klicke auf die Schaltfläche **Tr** das Terrain-Menü öffnet sich. Klicke auf die Schaltfläche **Import PCX** und importiere Deine Datei. Das Ergebnis sieht etwa wie im Bild unten aus. (D3Edit – World View)



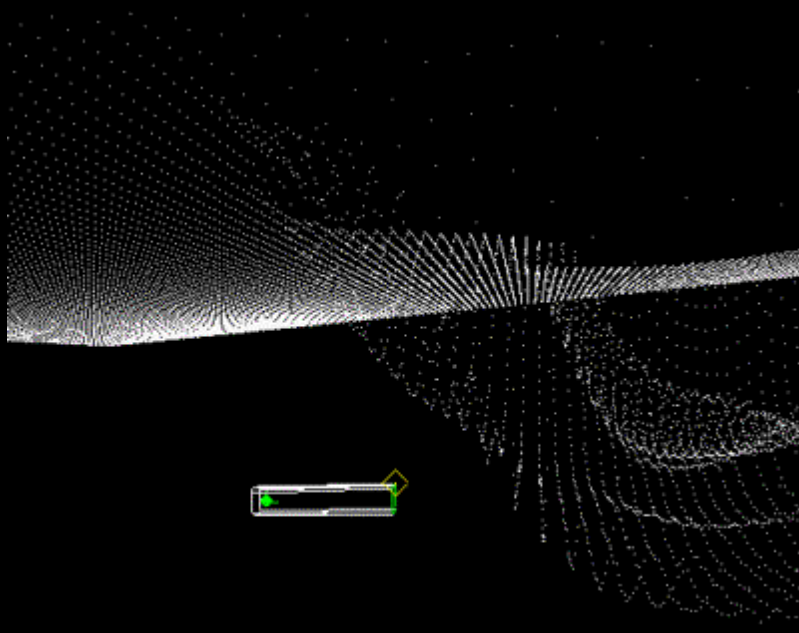
Nun müsst Ihr noch die Aussenwelt texturieren und das geht so. Zunächst klickt mit der rechten Maus-Taste in die World View Ansicht. In den darauffolgenden Menü wählt Ihr Textured with Outline. Jetzt seht Ihr die "Landschaft" mit den Standard-Texturen. (Lasst Euch nicht von der Darstellung irritieren) Ihr könnt nun auch mit **Z** auf den "Z-Buffer" Modus umschalten, man kann dadurch eine bessere Darstellung bekommen, braucht aber einiges an Rechenpower. Das müsst Ihr aber selbst ausprobieren was besser ist.

Weiter mit dem Texturieren. Ihr brauch nur das "Tal" und die "Hänge" zu texturieren, eventuell noch die oberen Kanten. Der Rest, den Ihr nicht benötigt wird sowieso "Abgeschnitten" doch dazu später. Wählt zunächst eine Textur aus der **Textur Bar (Terrain Textures)**. Wie das geht müsstet Ihr eigentlich wissen, wenn nicht schaut weiter vorne nach. Ich wähle bei meinem Beispiel die Textur **Dan_Moss1**. Haltet nun die **Umschalt** Taste gedrückt und klickt in das "Tal", jetzt habt Ihr das erste Stück texturiert. Wiederholt es solange, bis Eure Aussenwelt (Tal und Hänge) komplett texturiert ist. Bei mir sieht das dann wie hier unten links aus.



Wenn Ihr das erledigt habt, klickt Ihr auf die Schaltfläche **Calculate Terrain Occlusion**. Im nachfolgenden Dialogfeld werdet Ihr noch einmal gefragt, ob Ihr es berechnen wollt, auf **Yes** klicken. Nun hat es den Anschein, das der Editor abgestürzt ist oder sich aufgehängt hat. Keine Panik er arbeitet noch! Wenn Ihr ein Tool habt, das die Prozessorauslastung anzeigt, werdet Ihr sehen das D3Edit noch arbeitet. Wenn D3Edit mit der Berechnung fertig ist, bekommt Ihr ein Dialogfeld daß Euch das bestätigt. Man wird ebenfalls zum Speichern aufgefordert. Jetzt schauen wir uns noch einmal unsere Aussenwelt an und sehen, das alles was nicht benötigt wurde abgeschnitten ist.

So, die Aussenwelt ist nun fertig. Aber wie können wir vom Inneren der Level nach aussen fliegen? Um Euch das zu erklären, habe ich den Default Room etwas in die Länge gezogen und ihn an einen "Hang" platziert, wobei die **Rainbow Texture** nahe am Hang ist. (siehe Bild rechts) Ich habe das wie folgt gemacht. Ich habe in der Ansicht **Current Room View**, im Modus Vertex (**Strg+R**), alle Vertex mit **M** markiert und in der Ebene **Side (ZY)** (Grid 100), den Raum mit Hilfe der Tasten **2**, **4**, **6** und **8** an die richtige Position gestellt. Jetzt müssen wir noch den Player Startpunkt nach holen, hierzu wechseln wir in den Objekt Modus (**Strg+G**) und öffnen die **Objekt Bar** und seht nach, ob der Player Startpunkt Current ist, das seht Ihr in der Objekt Bar, wo dann etwas von "PLAYER 0" steht. Nun wieder in die Ebene Slide (ZY), mit den oben genannten Tasten den Player Startpunkt wieder in den Raum positionieren. Kontrolliert das in dem Ihr in der World View Ansicht nachschaut.

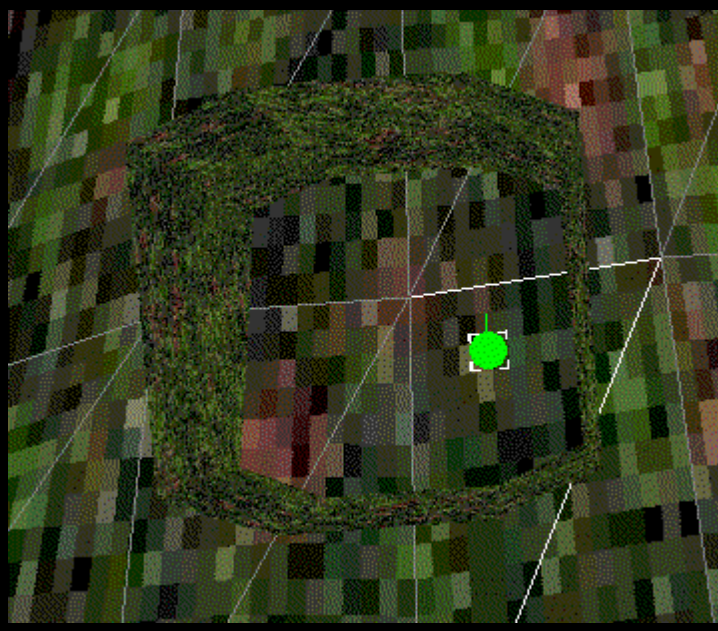


Jetzt brauchen wir noch einen **External Room**. Wechselt ins Room Menü und wählt Add Room at Current Face oder betätigt die Taste **Einfg**, beachtet aber, daß Ihr das Face ,das in Richtung Aussenwelt zeigt, angeklickt habt.

Der Neue Raum ist jetzt dran, zieht ihn soweit in die Länge, daß er in die Aussenwelt ragt. Texturiert ihn nach Euren Geschmack.

Anschliessend klickt das Portal, das zwischen den beiden Räumen liegt, an und wählt im Room-Menü Delete Current Portal. Jetzt klickt Ihr im ersten Raum auf die **Rainbow Texture** und wählt im Room Menü Link Room to New External Room. Der Editor fragt dann ob er einen neuen Raum machen soll, klickt hier nein. Jetzt müssen wir noch den zweiten Raum als **External** kennzeichnen, dazu klicken wir ihn im World View den Raum an und wählen im Menü Window.../Room Properties. Setzt das Häkchen so wie im Bild rechts.

Danach joint die Räume.
Jetzt müssen wir noch das Face vom zweiten Raum auf der Außenseite löschen (der Ausgang). Dazu wechseln wir in die Current Room Ansicht, mit **Strg+F** in den Facemodus, wählen das Face, markieren es mit der **Space**-Taste und löschen es mit **Entf**. Nun haben wir scheinbar den Ausgang, doch es ist ja noch kein Loch in dem "Hang". Man könnte zwar nach draußen fliegen, doch nach innen würden wir nicht mehr kommen.

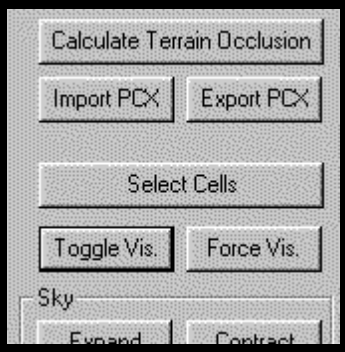


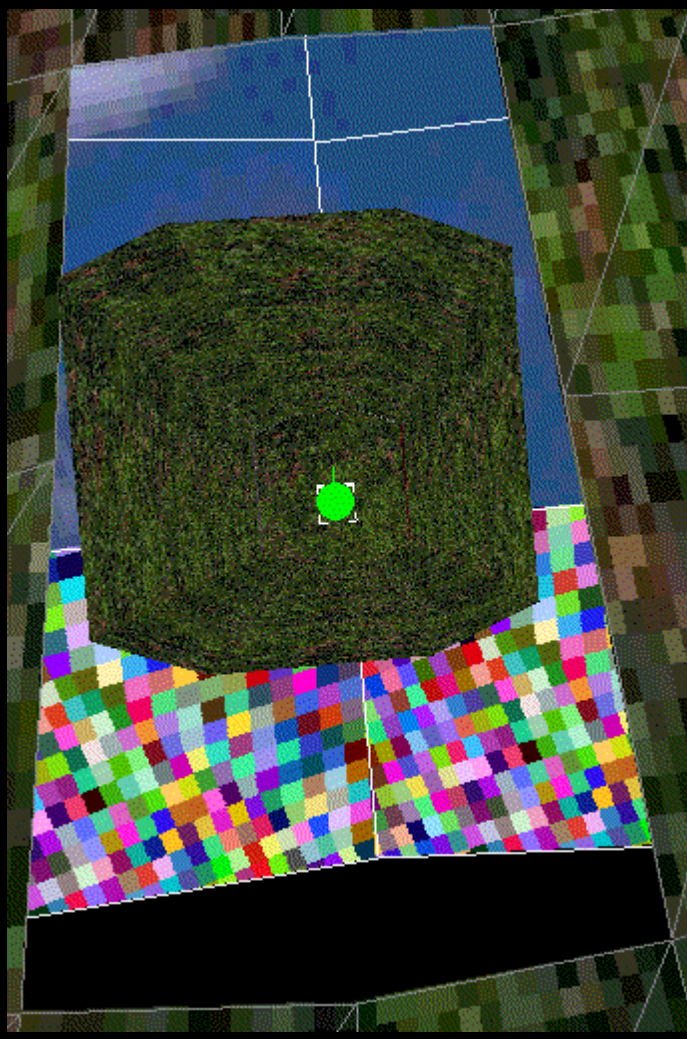
Schaltet mit der Taste **Z** den Z-Buffer ein, den benötigen wir jetzt. Geht in die World View Ansicht und positioniert Euer Blickfeld nahe am Ausgang wie im Bild rechts. Da seht Ihr bereits das schon ein "**Cell**" selektiert (angeklickt) ist.

Nachdem wir die Cell angeklickt haben, sieht man auch daß unten in der Statuszeile etwas angezeigt wird, im Bild hier unten zu sehen.

In diesem Bild sehen wir daß Cell 20095 angeklickt ist.

So, jetzt klickt auf die Schaltfläche **Toggle Vis**. daraufhin wird das angeklickte Cell gelöscht. Anmerkung, wenn Ihr Cells wieder einfügen wollt, müsst Ihr die Cell - Nummer wissen, deswegen schaut immer auf die Statuszeile und notiert Euch diese Nummern! Wiederholt die Schritte bis die Öffnung nicht mehr blockiert wird und es aussieht wie im Bild rechts.

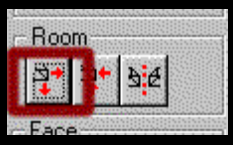




Das wäre geschafft, nun müssen wir noch etwas um den Ausgang bauen, damit man nicht aus dem Level fliegen kann. Ich löse dieses Problem wie folgt:

Geht in die Current Room Ansicht, markiert alle Faces (mit der Taste **M**) und wählt im Menü Edit.../Copy. Danach wählt Ihr das Menü File.../New und erstellt einen neuen Raum, New Room. Nachdem das erledigt ist, geht wieder in das Menü Edit und wählt Paste. Jetzt vergrössern wir den Raum, dazu rufen wir das Room - Menü mit der Taste **R**. (das Terrain Menü könnt Ihr schließen)

Nun klickt Ihr auf die Schaltfläche die im kleinen Bild Rot markiert ist.



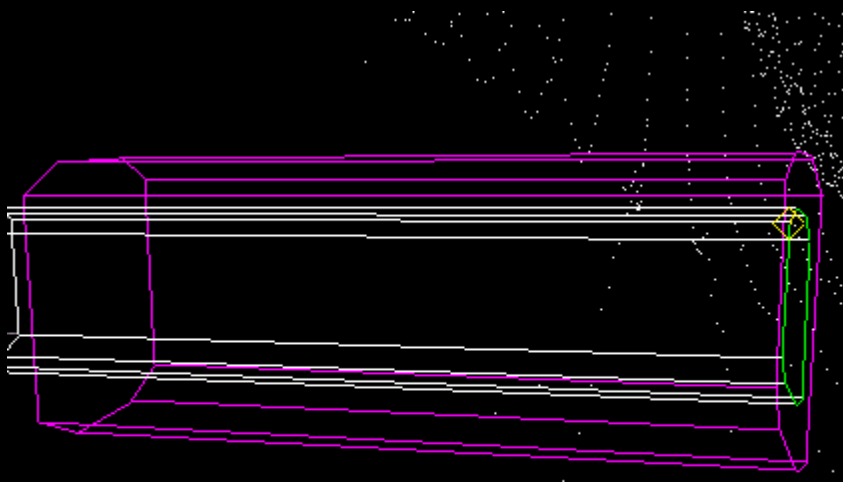
Klickt ca.10 mal darauf.

Nachdem das gemacht ist speichert den Raum, z.B..: "`temp1.orf`", schliesst den Raum und öffnet ihn wieder. Ich weiß nicht warum, aber es geht nur so. Nun müssen wir den Raum noch etwas bearbeiten. Der Raum braucht für unser Vorhaben nicht so lang zu sein. Zunächst machen wir ihn um die Hälfte kürzer. Markiert auf einer Seite die Vertex und macht ihn dann mit Hilfe der Tasten **4** oder **6** kürzer. **Bevor Ihr weitermacht schaut nach ob alle Faces vorhanden sind, auch die äusseren !!!** (sonst bauen wir uns ein "Bad Shell" ins Level)

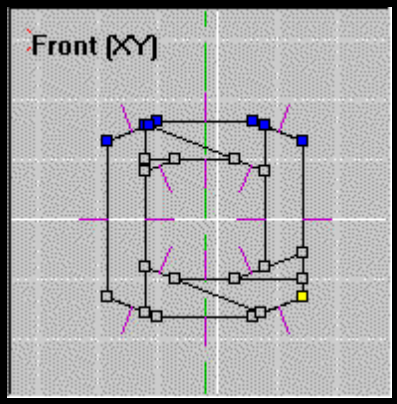
Jetzt wechseln wir in den Face-Modus (**Strg+F**). Markiert alle Faces (Taste **M**). Im **Room** Menü gibt es eine Funktion **Flip Faces**, (**N**) darauf klicken und Ihr bekommt die Bestätigung in einer Dialogbox. Nun schliessen wir den Raum und wechseln in den zweiten Level Room (der Externe). In diesem Raum hatten wir vorhin das äußere Face gelöscht, dieses brauchen wir nun wieder. Wechselt in den Vertex Modus und markiert die Vertices wo das Face fehlt, Ihr erkennt es an den kleinen roten Strichen, die immer in der Mitte eines Faces sind. Danach wieder zurück in den Face Modus (**Strg+F**) und die Taste **Einfg** drücken. Jetzt müsst Ihr schauen in welcher Richtung das Face steht, es muß nach außen zeigen.

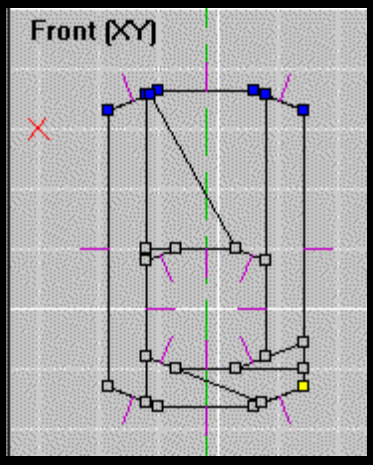
Speichert den Raum und schließt ihn.

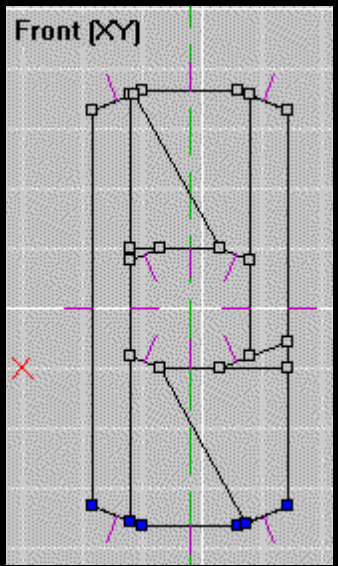
Wechselt in die Ansicht World View und seht nach, ob das äussere Face grün umrandet ist. Ist das nicht der Fall, dann klickt es an. Jetzt den Raum `temp1.orf` öffnen, hier müssen wir das äußere Face markieren (anklicken und **Space** drücken).
Nun wieder zurück in die World View Ansicht, ins Menü Room.../Place Room at Current Room. Eure World View müsste jetzt aussehen wie im Bild rechts.

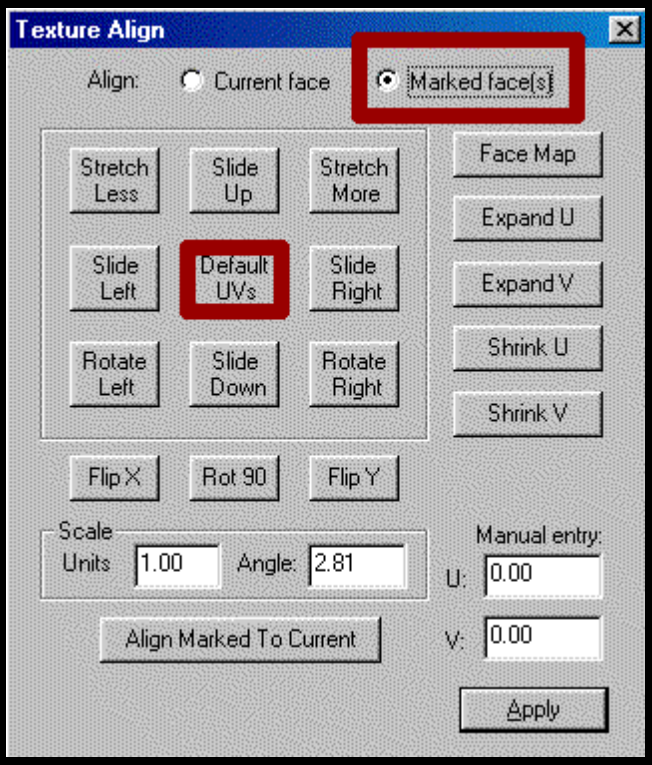


Machen wir weiter, geht wieder in das Menü Room und klickt auf Attach Room ,der Raum ist nun eingefügt. Bevor wir etwas anders machen, drückt die Taste **M** um den Raum zu markieren, dann klicken wir irgendein Face des angefügten Raumes an und gehen wieder in das Room Menü und wählen dort Combine Rooms. Ok, jetzt schließen wir den Raum `temp1.orf`. Wechselt nun mit **Strg+Tab** in die Room View Ansicht und markiert die oberen Vertex. (Bild unten links) Wir müssen ja noch die Stellen verdecken, die wir vorhin gelöscht haben. Im Bild unten mittre habe ich die markierten Vertex um 20 Einheiten nach oben verschoben (Grid 10) dann mit der Taste **U** die Vertices demarkiert und dasselbe an der unteren Seite gemacht (Bild unten rechts). Bevor wir weitermachen, löschen wir noch das hintere Face von dem angefügten und dem zweiten Raum. Weil man sonst nicht mehr hinein fliegen kann.



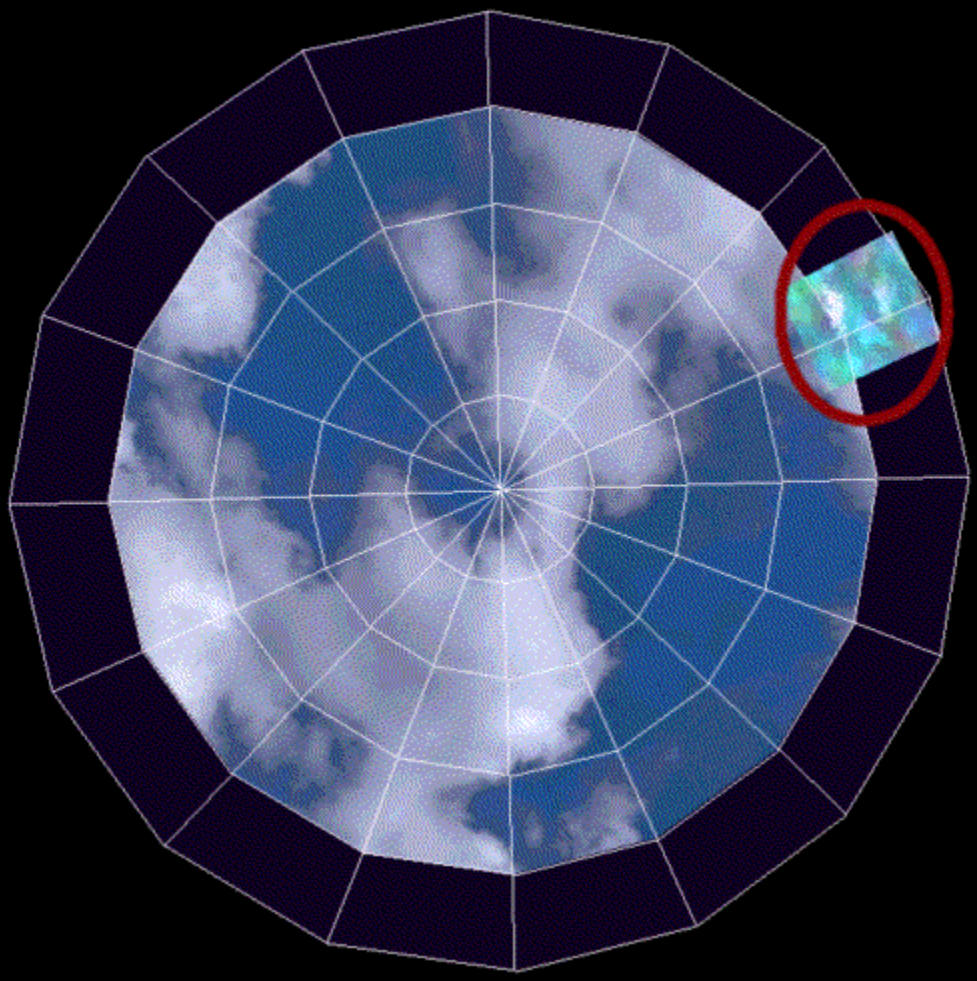




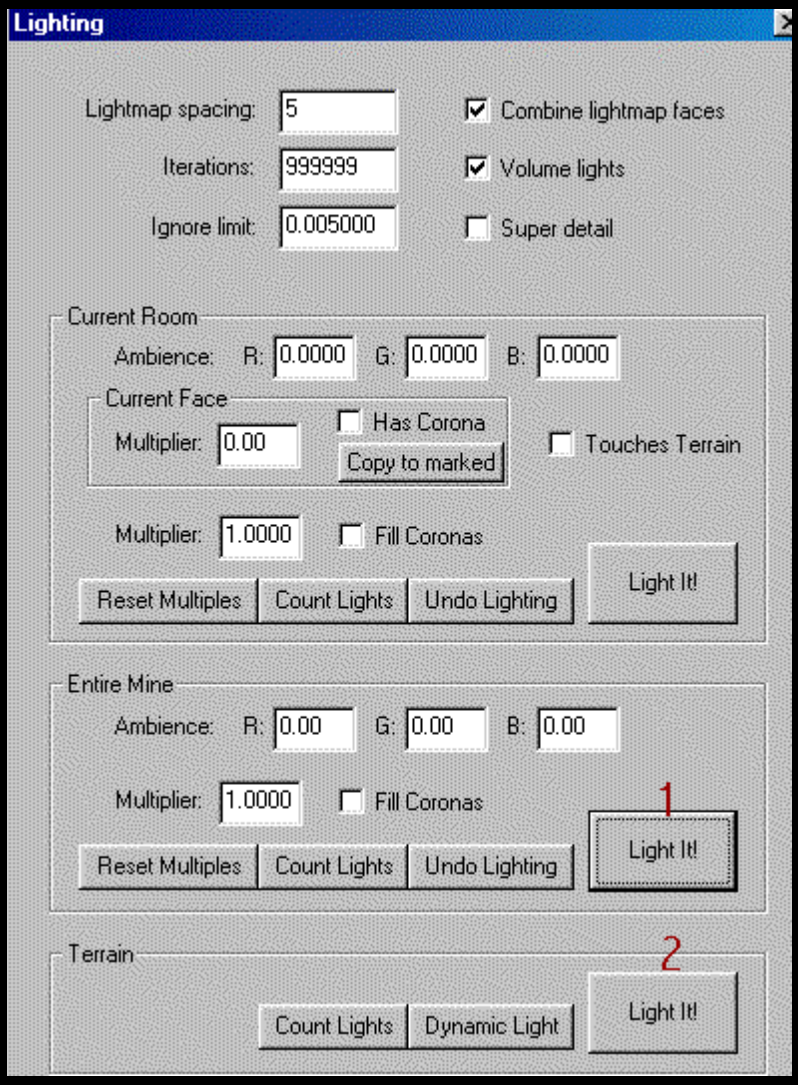


Wir demarkieren die Vertices nach dieser Aktion und wechseln mit **Strg+F** in den Face Modus, markieren mit **M** alle Faces und klicken in der Texture Bar auf **Align...**. Wie im Bild links zu sehen ist, habe ich ganz oben Marked face(s) angeklickt und dann auf die Schaltfläche **Default UVs** geklickt. (wegen der Textur-Darstellung)

So, nun brauchen wir noch einen Himmel und eine Sonne. Soll ja auch gut aussehen! Bewegt Eure Position (Kamera) soweit zurück, so das man den Himmel und die Sonnenposition erkennen kann. Die Sonneposition ist das rotumrandete Viereck. Um da nun eine Sonne zu platzieren, müssen wir in der Texture Bar eine Sonne suchen. Ich habe die Texture Sun1 gewählt (im Bild nicht zu sehen). Nun haltet die Umschalt - Taste gedrückt und klickt auf das Viereck. Den Himmel werde ich so lassen, aber wenn Ihr eine andere Textur am Himmel haben wollt, müsst Ihr Euch eine auswählen, die **Umschalt**-Taste gedrückt halten und mehrmals in den Himmel klicken.



Zu guterletzt müssen wir noch die Lichter berechnen. Wechselt zm Menü Window und wählt Lighting. Füllt alles so aus wie ich es im Bild rechts gemacht habe. Klickt dann auf die Schaltfläche die im Bild mit 1 beschriftet ist, das nächste Dialogfeld mit Ja beantworten und das Licht in der Mine wird berechnet. Jetzt klickt in der Box Terrain auf **Count Lights** (Anzahl der Lichter ermitteln) und noch einmal auf die Schaltfläche 2 klicken und fertig.



Hinweis: Die Terrain-Lichtberechnung kann sehr lange dauern, wenn Ihr bei Iterations **999999 stehen laßt !!!**

Wenn Ihr das auch geschafft habt, speichert den Level und rein in den MN3 Packer, fliegt einmal durch den Level und schaut es Euch an.

Das war ziemlich umfangreich, aber ich finde es gibt zu wenige Level mit Aussenwelt und wollte Euch von Anfang bis zum Ende erklären.

Viel Spass beim Level Bauen. :-)

File: 058-TestTerrain.zip

058 - Außenwelt – Update

Ragil Ral

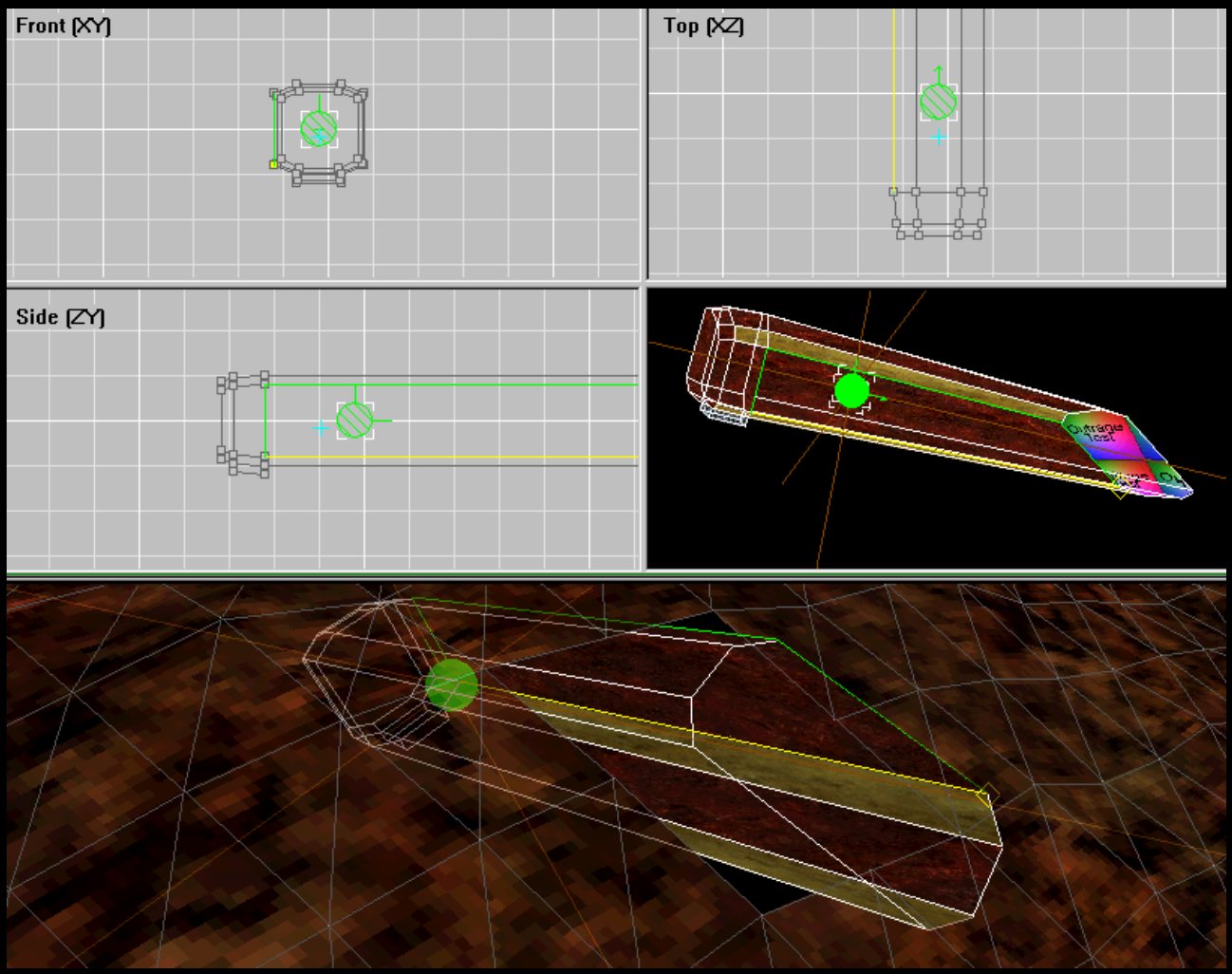
In der derzeitigen Version von D3Edit (1.1 [Beta 10 | Atan P.0_1_39(p), bei früheren weiß ich es nicht) gibt es noch eine andere Möglichkeit, die Verbindung zwischen Terrain und Mine zu realisieren.

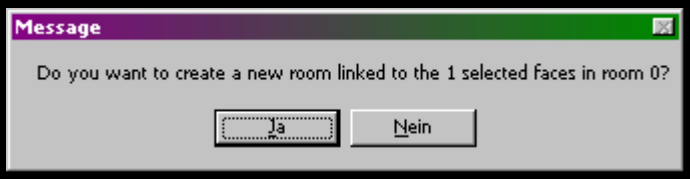
Erst noch ein paar Worte zum Terrain.

Der Spieler kann noch Terrain überfliegen, wenn in der Höhenkarte 97% Weiß vorliegt. Weißere Gebiete sind zu hoch. Ich hab dazu eine Testdatei mit `.d3l` erzeugt, die Ihr in der `.zip` findet.
Zum Anderen, wenn der Grauwertunterschied zwischen zwei benachbarten Pixeln mehr als ein paar Prozent beträgt, führt dies zu unschönen Spikes im Terrain; man kann dies natürlich aber auch als gestalterisches Element verwenden. Die Terrainzellen werden so angelegt, daß jede Terrainzelle aus vier Pixeln erstellt wird, daher wird ein einzelner Pixel, dessen Grauwert stark von der Umgebung abweicht, eben als Spike am Terrain sichtbar. Des weiteren solltet Ihr in den Editor Settings Use Terrain LOD anhaken sodaß man alle Terrainzellen sieht, andernfalls werden Zellen die auf der selben Ebene liegen zusammengefaßt, beim sichtbar/unsichtbar schalten aber wieder getrennt, was das markieren sehr erschwert. Außerdem, haltet euch von den Terraingrenzen fern. Kommt man denen mit Räumen oder Objekten zu nahe, passieren merkwürdige, seltsame Sachen...

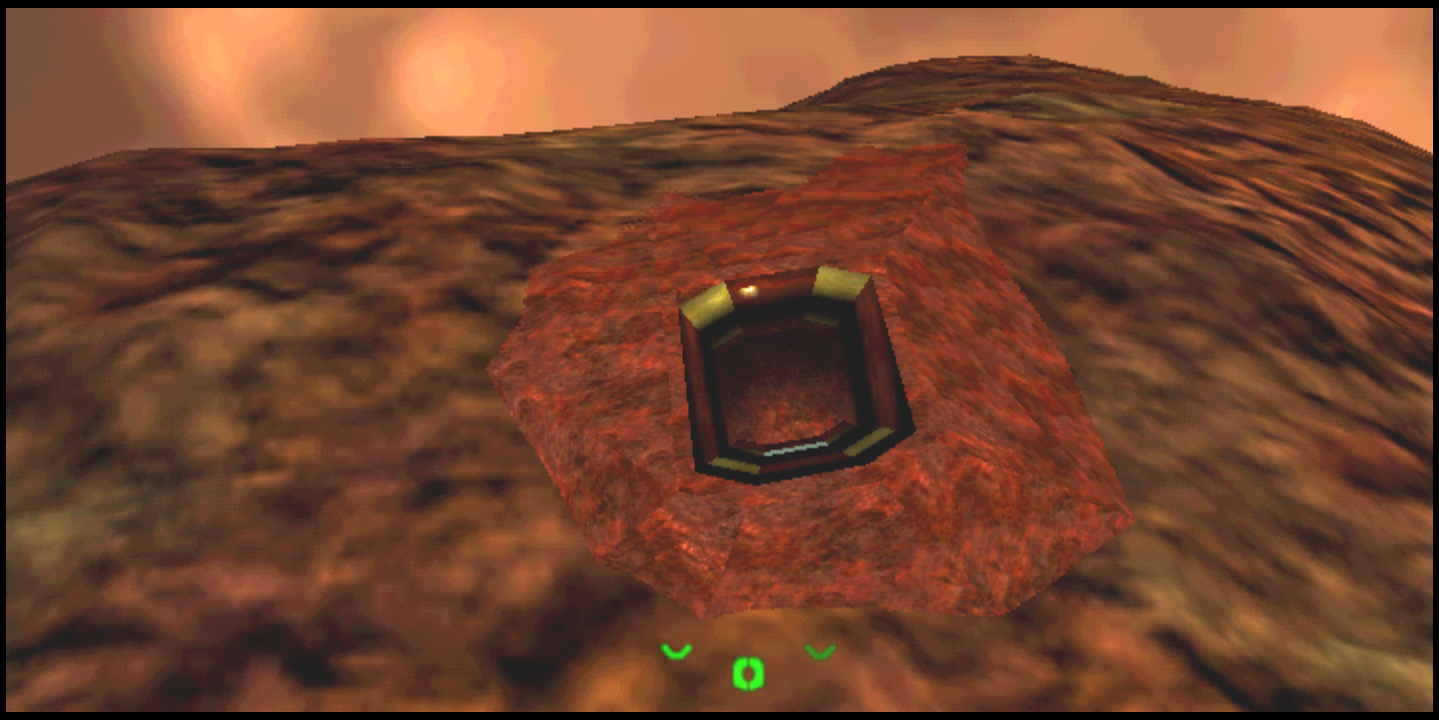
Terrain-Anbindung neu

Im vorgeganenen Tut habt Ihr einen Raum angehängt, diesen als extern gekennzeichnet, Faces gelöscht, etc. etc.; der neue Editor bietet hier nunmehr einen bequemeren Weg an. Baut Euch also ein Terrain, und erzeugt einen Raum sodaß er durchs Terrain schaut. Nicht vergessen, die Seite wo man dann rausfliegen soll, MUSS die Textur ‚Rainbow Texture' haben (die erste unter ‚Terrain').
Der Screenie zeigt einen Default-Raum, den ich an eine appropiate Stelle bewegt und in die Länge gezogen habe. Danach habe ich ihn mit der ‚Cut Room'-Funktion beschnitten (&) und die entsprechenden Terrainzellen ausgeschaltet.





Macht jetzt daß Face current, aus dem man rausfliegen soll. Dann, in der Weltansicht, Geht auf Room->Link Room to new External Room, dann bekommt ihr ein Dialogfeld, daß ihr hier mit **Ja** bestätigt. Im vorigen Tutorial mußtet Ihr mit Nein bestätigen, da Ihr ja bereits einen Externen Raum angelegt hattet.
D3Edit hängt jetzt einen Raum an, der nur aus einem Face besteht und als ‚external' gekennzeichnet ist. Um nun die Abdeckung für die Lücken im Terrain zu machen, geht folgendermaßen vor:
Erzeugt Euch einen passenden Außenraum, etwa mit Fischleins Methode von vorhin. Das Face, über welches die Abdeckung an den neuen Externen Raum angehängt werden soll, muß die Normal nach außen haben.
Dann geht mit **R** zum externen Raum, der zur Zeit nur aus einem Einzigen Face besteht, dessen Normal nach außen zeigt. Da es ein Portal ist, kann man es nicht flippen, also fügt ein zweites Face ein, dessen Normal muß dann nach innen zeigen. Daran wird die Abdeckung angehängt. Nun joint die Räume ganz normal wie sonst auch mit Room->Place Room at Current Room und Room->Attach Room. Zupft die Abdeckung noch so zurecht, daß die Lücken ganz verdeckt sind, voila, fertig.
Texturieren, die Texturen ausrichten, Lichter berechnen... etc. etc.



059 - Spielerstart in der Außenwelt

Ragil Ral

Es gibt mehrere Möglichkeiten den Spielerstart in die Außenwelt zu verlegen, aber leider keinen Königsweg.
Die Ausgangslage ist die: man kann den Player0 nicht direkt in die Außenwelt setzen, man kann ihn auch nicht dorthin verschieben. Es ist ebenso nicht möglich, den Player auf einen Außenraum zu verlegen, also einen Raum mit Flag 'External' - die Grafikengine flippt dann total aus.

Folgende Methoden sind bekannt und erprobt:

Zum einen einen internen Raum ans Terrain anlinken
'Unsichtbarer Raum': Voll ans Terrain gelinkter Innenraum, sodaß dieser nicht mehr sichtbar ist.
'Garage': Innenraum, größtenteils ans Terrain gelinkt.

diese beiden Methoden haben den Nachteil, daß Licht und Sound von der 'Innen'-Engine berechnet werden; man muß hier also derart anpassen, daß kein oder nur wenig Unterschied zu sehen ist. Hat etwa der Startraum kein Licht, wird der Pyro dort komplett schwarz; in reinen Single-Player-Missionen eher kein Problem sieht aber nicht so gut aus bei COOP-Missionen, oder wenn man ein Intro hat.

'Methode Thomas': Ein Level extrahieren, welches den Player0 draußen hat; dies kommt überwiegend nur in den Original-Missionen vor und hat daher eine rechtliche Komponente.
'Methode Boogie': den Player0 einfach nach draußen beamen.

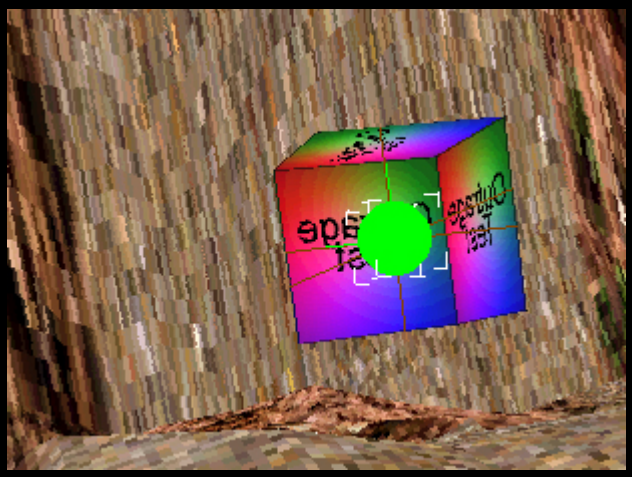
Die Praxis der Methoden sei hier nun beschrieben.

Internen Raum ans Terrain linken

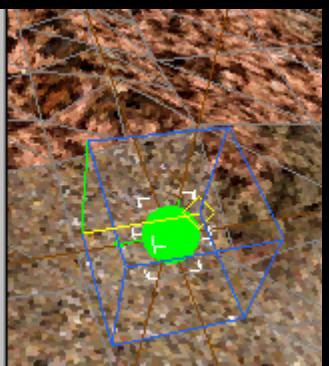
Es ist der einfachste Workaround.

A) Unsichtbarer Raum

Beginnt entweder einfach mit einem New->New Level-Default Room und importiert Euer Terrain-`pcx`, oder falls Ihr bereits ein Level habt, setzt einen Raum auf die Höhe des Terrains. Nehmt eine einfache Form, Würfel oder ein Tetraeder; dort setzt Ihr den Player0 hinein. Gebt dem Raum auf jedem Face die Rainbow-Textur, und markiert alle Faces (rechts). Von Dark kam der Tip dem Raum zu benennen, um Soundprobleme zu vermeiden.
Nun der Trick: In der World View geht nach Room->Link Room to New External Room, Der Editor fragt Euch dann ob Ihr die markierten Faces nach außen verlinken wollt, was Ihr mit ja bestätigt.
Der Raum wird nun unsichtbar, da alle seine Faces in ein Portal zur Außenwelt umgewandelt werden. Danach geht Ihr in die Room View, öffnet den Beleuchtungsdialog und



setzt den Haken bei 'touches Terrain'.



Ingame sieht das dann so aus (Raum mit `outlinem` eingeblendet):

Wie Ihr hier sehen könnt, stimmen die Lichtverhältnisse nicht.

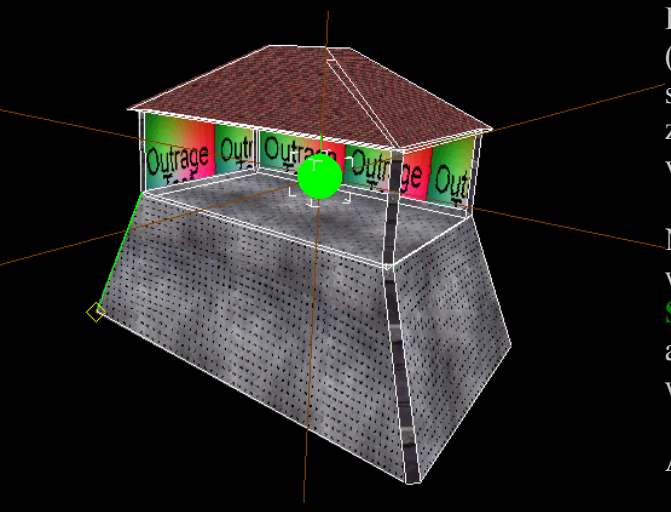
Player im Startraum

Player außerhalb des Startraumes

Nun ist experimentieren angesagt, um das Licht anzupassen. Eine weitere Sache ist, wenn Ihr Wetter auf dem Terrain habt findet dieses im unsichtbaren Startraum nicht statt; auch das wirkt ein wenig komisch; man könnte den Raum aber stark verkleinern sodaß das fehlende Wetter nicht auffällt.
Wenn Ihr im Editor nachguckt, werdet Ihr sehen daß da jetzt zwei Räume sind; zum einen der Würfel mit dem wir begannen, und der Link-Raum.

B) Garage

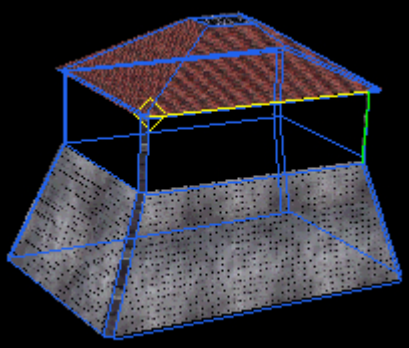
Wenn Ihr mehr Arbeit investiert, ist dies ein guter Workaround. Ihr baut einfach einen Verschlag oder sowas in der Art, verlinkt den in die Außenwelt und deckt ihn mit einem Externen Raum ab.



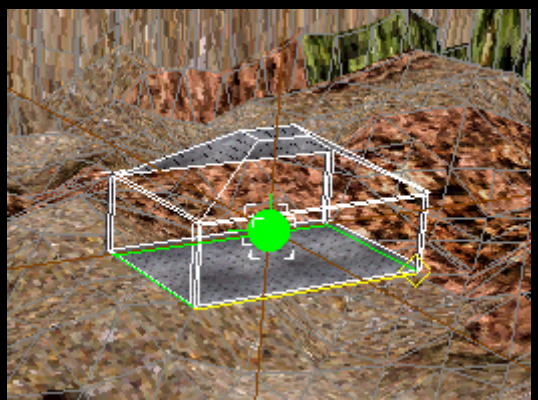
Baut das komplette Gebäude zuerst, Innen- wie Außenteil (links). Gebt den Faces die nach 'Außen' verlinkt werden sollen wieder die Rainbow Texture und verlinkt diese wie zuvor mit dem Terrain. Spieler in den Raum setzen nicht vergessen.

Nun macht folgendes: markiert alle Faces des Raumes, welche Außen liegen sollen. Schneidet diese zunächst mit **Strg-X** aus. dann geht mit **R** in den Link-Raum und fügt die ausgeschnittenen Faces mit **Strg-Shift-V** an Ort und Stelle wieder ein.

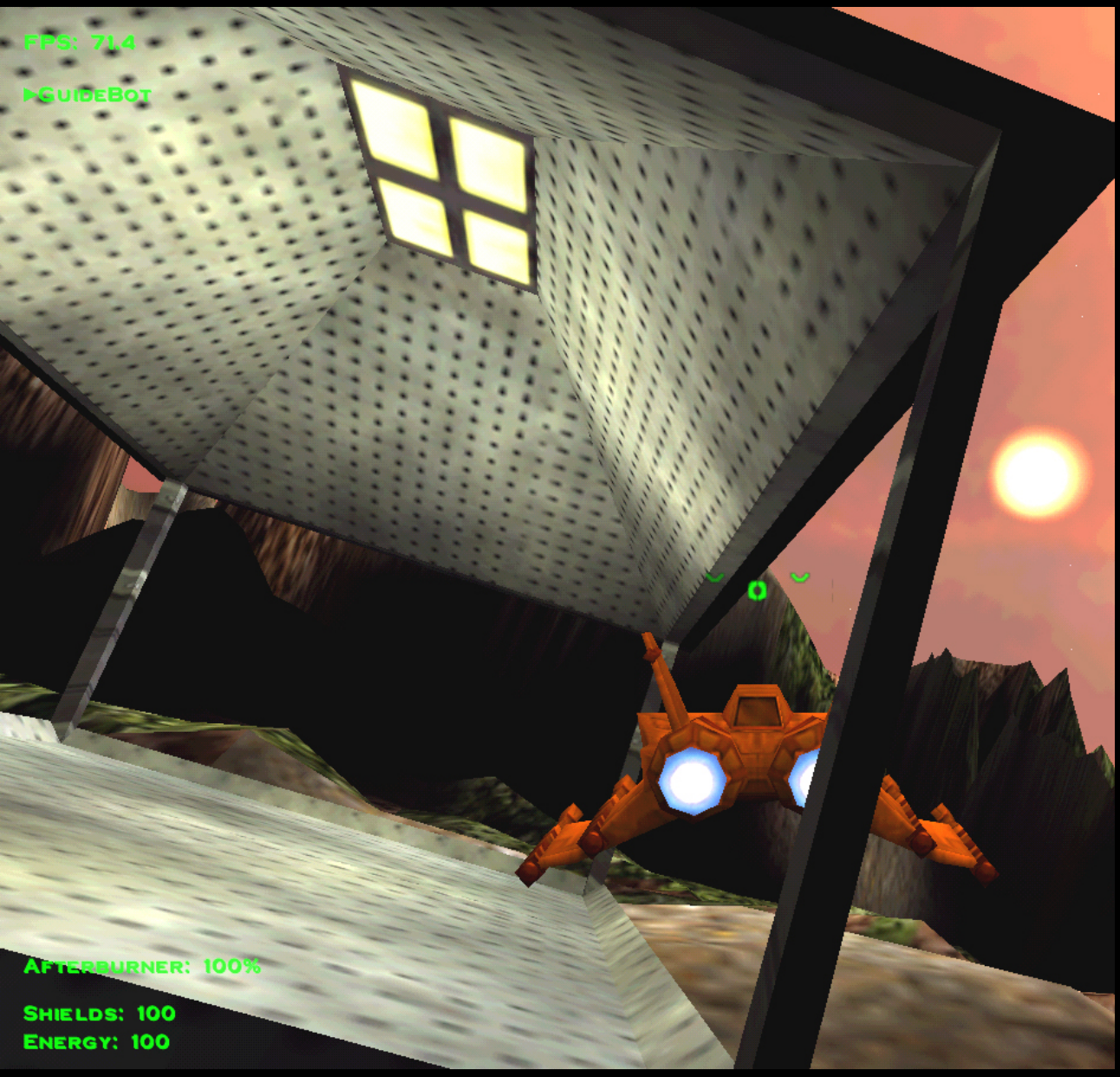
Ab da geht es weiter wie zuvor.



Nun solltet Ihr in etwa sowas haben, Links der Außenraum, rechts der verbliebene Innenraum.



Was bleibt, sind die unterschiedlichen Lichtverhältnisse, doch sieht das ganze schon etwas 'echter' aus:



Und daß es unter einem Dach nicht schneit oder regnet ist auch eher ein-leuchtend. Weiters habt Ihr hier die Freiheit die Sache 'luftiger' oder mehr geschlos-sen zu bauen.

Methode nach Thomas

Die gefällt mir persönlich am besten, da sich der Player0 hier wirklich am Terrain befindet, aber wie eingangs erwähnt gibt es hier vielleicht rechtliche Probleme.
Es läuft so daß Ihr ein Originallevel hernehmt, alles rauslöscht (Räume, Objekte, etc.) sodaß nur Terrain, ein Innen-Raum und der Player 0 enthalten ist.
In diesem Beispiel verwende ich `level15.d3l` aus den Retribution-Missionen, grundsätzlich ist das aber egal. Die Originalmissionen könnt Ihr mit Quickedit aus den d3-hogs extrahieren.

Ausräumen des Levels

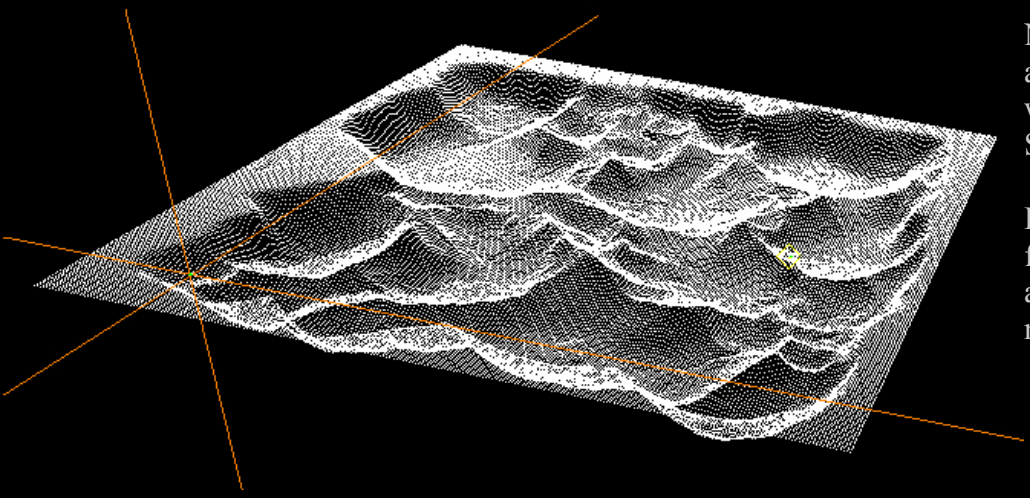
Zunächst entfernt alle Räume. Das geht am einfachsten indem Ihr sie in der World View selektiert und ausschneidet; so spart Ihr Euch die Abfrage die jedesmal kommt wenn Ihr die Räume einfach löscht. Klickt also irgendeinen Raum an und geht nach Edit->Select all Rooms attached to Current Room. Danach geht in den Raum 0 und deselektiert ihn (**Space**); ein Interner Raum muß im Level verbleiben. Dann löscht Ihr alle Portale des Raumes 0. Anschließend markiert in einem an den Raum #0 angrenzenden Raum ein Face mit **M**. Dann sagt Ihr einfach Edit->Cut selected Rooms oder **Strg-X**. Der Vorteil dieser Methode ist, daß die in den Räumen enthaltenen Objekte auch gleich mit

entfernt werden, und es gibt keine Probleme mit Türräumen. Verbliebene Räume werft Ihr dann mit **Del** raus.
Das ist nur ein Weg, probiert ruhig ein wenig herum.

Nun bleiben noch Objekte auf dem Terrain über; geht in den OI und klickt dort die Objekte an, löschen müßt Ihr diese jedoch über den Delete-Button im Objekte-Paneel, und zwar alle einzeln... Laßt nur den Player0 über; im OI seht Ihr unter Room Nr die Zelle eingetragen über welcher sich der Player0 befindet. Spätestens jetzt solltet Ihr das Level unter einem neuen Namen abspeichern.

Nun geht nach Window->Goals und schmeißt auch diese alle raus.

Danach importiert euer eigenes Terrain-pcx.



Nun habt Ihr einen komplett ausgeräumten Level, in welchen Ihr Eure eigenen Sachen hineinbauen könnt.

Diesen könnt Ihr dann als Basis für künftige Levels verwenden; also müßt Ihr diese Prozedur nur einmal machen.

Methode nach Boogie,

er hat das mit Teleportation gelöst; zunächst startet man ganz normal in einem Indoor-Raum. Der enthält ein Objekt, welches den Spieler per DALLAS auf ein korrespondierendes Objekt am Terrain beamt. Im Startraum sorgt Wind dafür, daß der Spieler auf das Objekt geschoben wird, woraufhin er unmittelbar aufs Terrain gelangt. Wie man den Spieler teleportiert - dies erfordert Scripting - werdet Ihr unter 093 - Teleporter lernen. Wind wiederum ist eine Eigenschaft des Raums und kann ohne Skripting unter Window->Room Properties gesetzt werden.

Ein gewisser Nachteil der Methode besteht darin, daß sich Skripten nicht immer mit COOP-Multiplayer verträgt.

Abschnitt F – Texturen

Alles was man im Spiel sieht bzw. sehen kann, ist mit Texturen ‚bespannt'. Manchmal findet man jedoch in den Standardtexturen von Descent 3 nicht genau das was man braucht bzw. will. Custom Texuren zu verwenden ist zwar nicht schwierig, doch lest Euch dazu auch die Sektion Nr 117 und 'Custom-Objekte in D3Edit sichtbar haben' im Anhang, durch.

060	**Eigene Texturen**	Einbinden eigener Texturen mit dem Gamtool und dem OGFTool	[DCG]Roadrunner	230
061	**Custom Texturen Anwenden**	Das ganze etwas tiefergehender.	Schplurg	232
062	**Custom Texturen: Tipps**	Tipps zur Texturenbeschaffung & Erstellung	Schplurg	236
063	**Texturen kachelbar machen**	Wie bekomm ich's bloß zur Fliese?	Ragil Ral	239
064	**Teiltransparente Texturen**	Deren Erzeugung und Einbindung	DigiJo	242
065	**Animierte Texturen - Screenshots**	Ingame(!)-Screenshots der Animierten Texturen	(LL)Dark	247
Fehler: Referenz nicht gefunden	**Fehler: Referenz nicht gefunden**	Erklärt, wie man mit Animake bewegte Texturen erstellt	Ragil Ral	Fehler: Referenz nicht gefunden
067	**Texturen @ 256x256**	Großtexturen und deren Anwendung	Ragil Ral	255

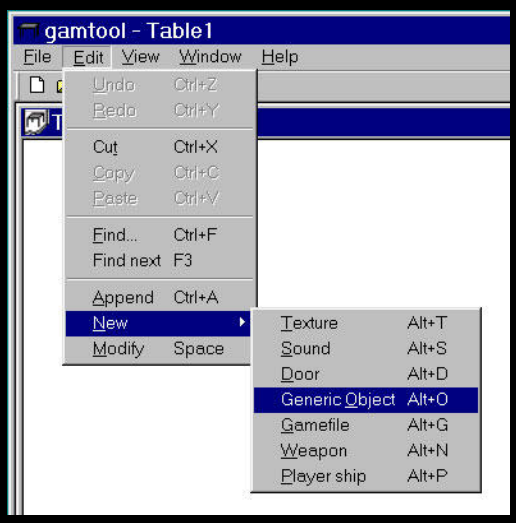
Toc

060 - Eigene Texturen

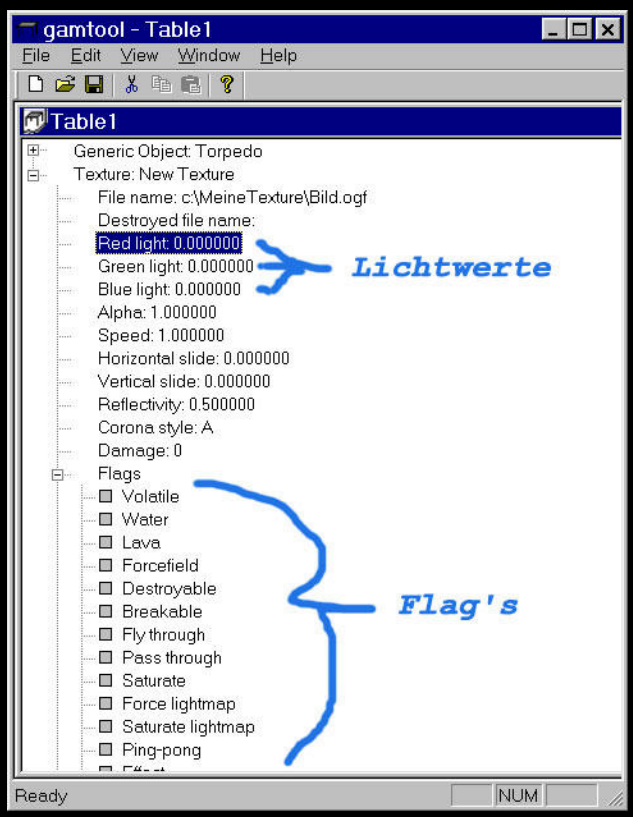
[DCG]Roadrunner

Gehe wie folgt vor: Erstelle eine Texture mit Deinem Grafikprogramm (Photoshop usw.). Sie sollte 128*128 Pixel oder 64*64 Pixel mit einer Auflösung von 72 Pixel haben! Speicher das ganze als TGA-Texture! Öffne nun den OGFtool und Konvertiere das TGA-Bild in eine OGF Texture.

Nun öffne den GamTool und wähle neue Texture!



Öffne die Texture zum bearbeiten (Bild links).

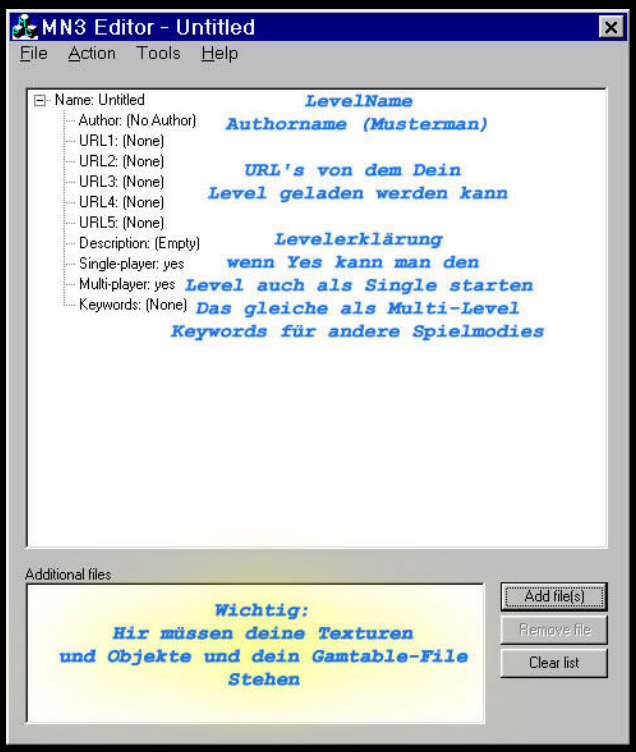


Hier kann man jetzt einige Einstellungen zur Texture vornehmen (Bild rechts) :
Als erstes, trage den Namen mit **Pfadangabe** **(maximal 34 Zeichen sind zulässig !!!)** ein . Den Pfad brauchst Du nur angeben, wenn Deine eigenen Texturen nicht im D3-Pfad abgelegt sind !!! Du kannst jetzt der Texture einen Lichtschein hinzufügen. Spiele mit den Farben um den passenden Schein für Deine Textur zu bekommen .z.B.: Ein Wert um 10 in allen drei Spalten ergibt ein weißes grelles Licht . Stehen alle Werte auf 0 ist kein Lichtschein zu sehen. OK wähle nun den Flag . Dies bedeutet, das Du beispielsweise die Texture unter dieser Gruppe suchen musst . Speicher das ganze und schon bist Du fertig . Der Gamtable-File muß den gleichen Namen (auch Groß/Kleinschreibung) haben wie Dein Level MN3.

Starte nun den LevelEditor (keine Datei öffnen!) und klicke im Menü auf Data. Wähle den Tablefile Manager. Als erstes steht der Base Tablefile (Nicht ändern) . In der zweiten Zeile kannst Du den Gamtable File eintragen . **File Browse** usw... Nun sollten Deine Texturen im Editor zu sehen sein und man kann sie vergeben

Tip: Lege einen Ordner an, in dem Du die Texturen und den GamTable-File reinpackst.

Starte jetzt den MN3-Editor.



Gehe so wie sonst vor und trage alle notwendigen Sachen ein! Nun kommt das Wichtigste : Wähle unten den Button 'Add Files. Hier müssen alle neuen Texturen, der Gamtable-file stehen
Der Name vom Gamtable-file muss der gleiche sein wie der Levelname der MN3 !!!
Nun speichern und das war's schon.

[Rip]DwnUndr hat mir noch eine andere Art gezeigt um eigene Texturen ins eigene Level zu bringen. Nur Du kannst keine Lichteffekte zur Textur hinzufügen. Er überschreibt im Prinzip vorhandene Texturen aus der Hog Datei . Seine Level kennt Ihr alle, "Worldsapart,Canyon Siege,Canyon Aftermath". Er öffnet den HOG- File und sucht sich eine Texture aus, die er in seinem Level nicht verwendet. Dann sucht er die Größe der Orginaltextur (Pixel ZB 128*128) und erstellt seine eigene, in der Größe (muss den gleichen Namen haben). Nun fügst Du Deine Texture in den MN3 Editor genauso wie den Gamtable ein (Ohne ein Gamtable-File). Auspacken kannst Du die HOG-File mit dem HOG Viewer Um die Grösse festzustellen .

Viel Spass !
[DCG]Roadrunner

061 - Custom Texturen Anwenden

Schplurg

Prerequisiten:

-Neben dem Spiel und D3Edit (ver 1 oder höher)
-OGF-Tool und MN3-Edit kommen mit dem Editor mit
-GamTool oder etwas vergleichbares um .gam-Dateien zu erzeugen
-Ein gutes Malprogramm um deine Texturen zu erzeugen (Photoshop, Paint Shop Pro...)

Machen Wir's

Die Textur die wir verwenden werden ist 128x128 Pixel groß mit einer Farbtiefe von 24 bit. Du kannst Deine eigene machen, oder meine verwenden (`.zip`). Wenn Du sie selbst machst, stelle sicher daß es eine 128x128 `.tga`-Datei ist. Ein `.tga` brauchen wir für das OGFTool, um es in eine verwendbare Textur zu verwandeln (`.ogf`). Rechts ein Bild meiner Textur. Ich glaube da D3 wirklich eine bessere Holztextur braucht wäre die hier praktisch für Euch.

Ok, wir haben ein 128x128 24 bit `.tga`-Bild. Starte nun das OGF-Tool entweder im D3Edit über Tools->OGF Tool (mußt Du gegebenenfalls erst konfigurieren) oder über den MN3-Packager über Tools->OGF Converter.

Im Bild links siehst Du, daß ich die Textur `woodply128.tga` ausgewählt habe. Ich gebe meinen Bildern gern detaillierte Namen... und ja, ich habe einige Three Stooges-Texturen ☺

Stelle sicher, daß die Output Settings auf ‚Outrage Graphics File' (`.ogf`) eingestellt sind. Durch Ändern der Dateitypen-Auswahl kanns Du von `.tga` nach `.ogf` oder umgekehrt konvertieren. `.oga`-Texturen (animierte) werden auch unterstützt. Stelle gegebenenfalls noch den Ausgabe-Folder ein.

Wenn alles so eingestellt ist wie Du willst, drücke den Button **Start**. Obwohl Du keine entsprechende Nachricht bekommst, hast Du gerade eine `.ogf`-Textur erzeugt. Navigiere zu deinem Ausgabe-Ordner und finde die woodply128.ogf. Klicke sie einmal an, und sie taucht im kleinen Voransichtfenster auf. Wenn Du sie siehst hast Du's richtig gemacht ☺ Nun müssen wir das Teil nur noch ins Level bekommen!

Um die Textur in einem Level zu verwenden und sie in D3Edit sehen zu können, müssen wir ein paar Dateien mehr erzeugen.

Die GAM Datei

Die .gam-Datei sagt Descent3 daß eine eigene Textur verwendet wird. Du kannst auch eigene Sounds und Objekte zum Level hinzufügen. Sie müssen alle in der .gam gelistet sein, oder D3 wird sie nicht verwenden.

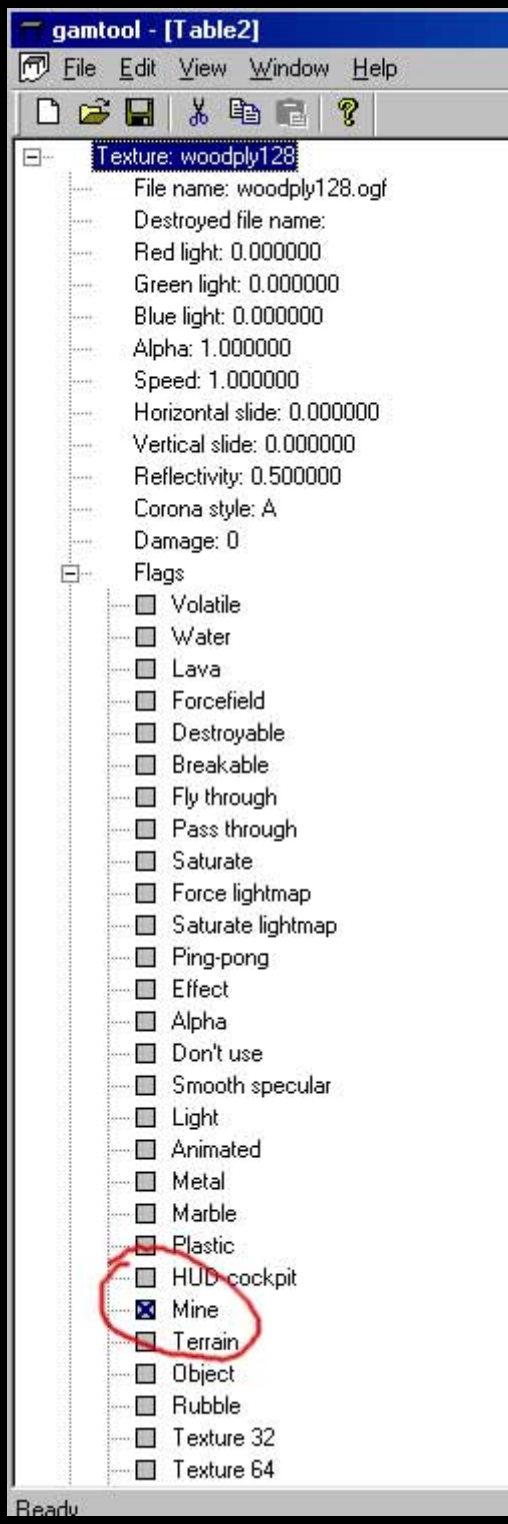
Öffne GAMTool. Wenn nicht bereits ein neues Fenster offen ist, gehe nach File->New. unter dem Menü Edit wähle New Texture (beachte die anderen Auswahlmöglichkeiten wenn gerade dort bist!) Dann doppelklicke den neuen Eintrag sodaß Du tippen kannst und ersetze ‚`New Texture`' mit `woodply128` oder etwas aussagekräftigem. Drücke **Enter** um die Änderung abzuschließen.

Dann klicke das kleine ‚+'-Zeichen und klappe das Listing aus (Bild rechts). Dort wo File name steht schreibe `woodply128.ogf` hin oder etwas beschreibendes. Stelle sicher daß Du es exakt wie den Dateinamen schreibst und die Extension anhängst... hier wird die Textur vom Spiel aufgerufen. Die Änderung wieder mit **Enter** abschließen.

Nun suche das + vor Flags und explandiere das Listing. Stelle sicher, daß die Box Mine angekreuzt ist, und alle anderen nicht. Das sagt D3Edit daß die Textur in der Texturenbibliothek unter der Sektion Mine angezeigt wird. Das ist die Default-Auswahl.

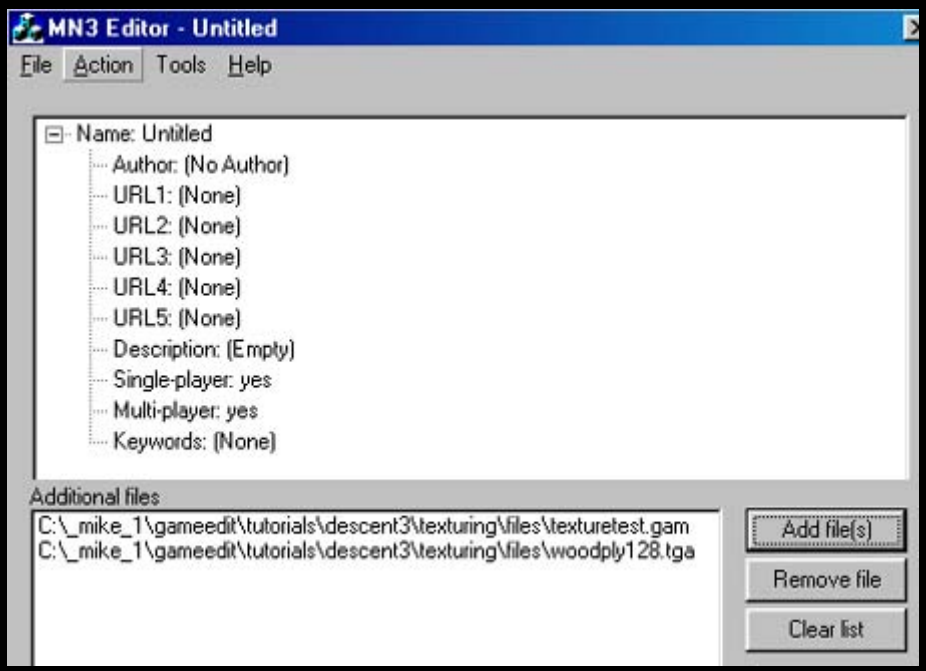
Du kannst in diesem Bild sehen, daß Du viele Optionen hast, wie Fly through und Pass through. Dies läßt die Spieler durch die Textur durchfliegen und -schießen. Andere Einstellungen sind nich so einfach und erfordern andere Einstellungen um zu funktionieren.

Speichere Deine `.gam`-Datei unter `texturetest.gam` auf Deine Festplatte. Die `.gam` MUSS exakt denselben Dateinamen haben wie Deine `.mn3`! Wir haben noch keine, aber werden bald.
Nächster Schritt...



Die MN3-Datei

Sie ist normalerweise die Datei, die Du als fertiges, spielbares Level verwendest. Aber das MN3-Format wird auch von D3Edit verwendet, sodaß eigene Texturen im Editor sichtbar sind.
Öffne den MN3 Editor.



Verwende den Button Add File(s) und finde Deine `texturetest.gam` und die `woodply128.ogf`.

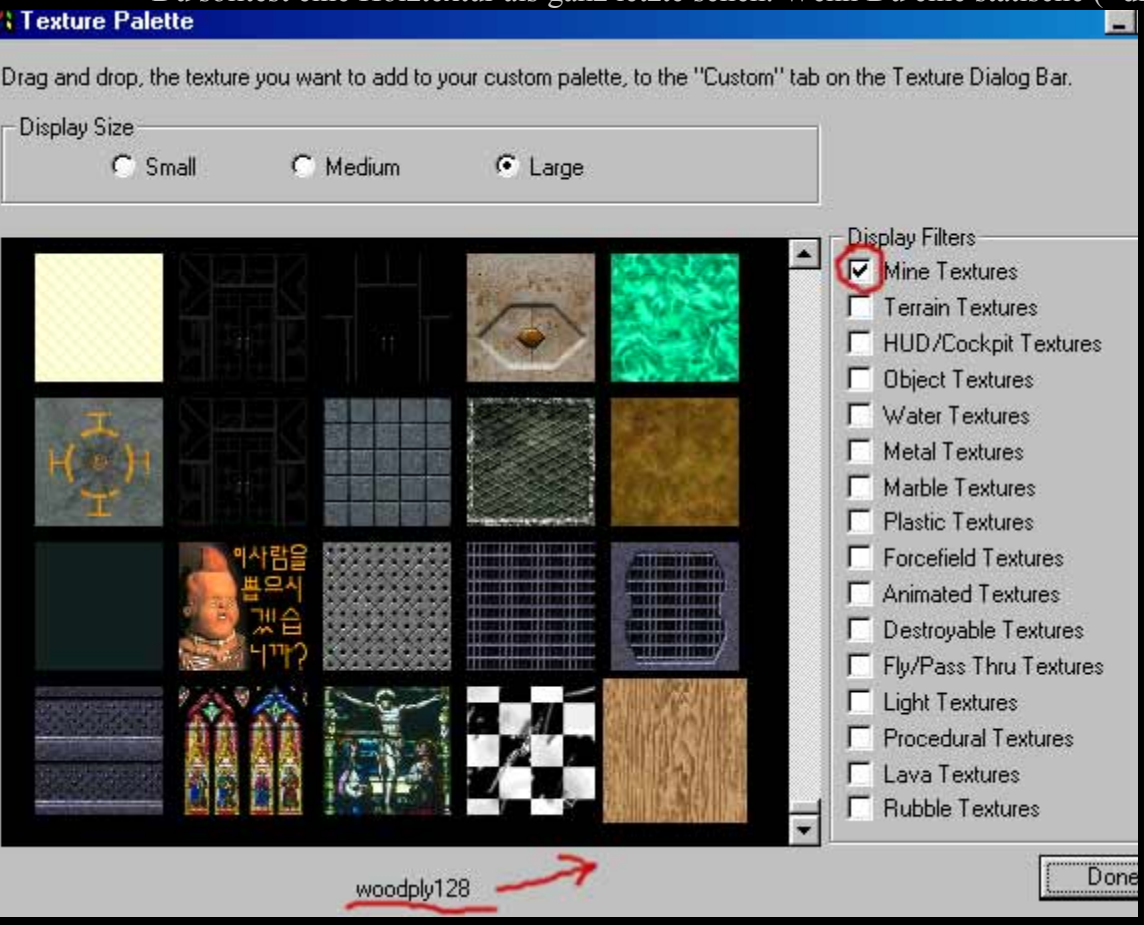
Sonst brauchst Du hier nichts weiter tun. Speichere die .mn3 als texturetest.mn3 (gleicher Name wie die .gam-File.)
Jetzt wirds Zeit unsere Textur zu verwenden!

Die Textur verwenden

Öffne D3Edit. Mach noch nichts sonst! Um unsere Texturen im Editor zu sehen, müssen wir D3Edit sagen wo sie sind. Deswegen haben wir die `.gam` und die `.mn3` erzeugt.

Gehe ins Menü `Data` und wähle Tablefile Manager (unsere `.gam` ist eine Tablefile). Da sind drei Reihen Buttons. Klicke den Button File Browse in der Reihe „Mission Tablefile (Optional)“. Finde Deine `.gam` und bestätige mit **OK**.
Dann gehe nochmal auf Data und wähle die zweite Auswahl, Configure a Mission Hog. Klicke den Button Change und suche deine .mn3-Datei. Mit OK bestätigen. Ein Dialog poppt auf, einfach mit OK klicken weiter.

Nun öffne D3Edit's Textur-Paneel, wähle Mine Textures und scrolle ganz bis zum Ende der Liste. Du solltest eine Holztextur als ganz letzte sehen. Wenn Du eine statische (=die mit dem Farbrauschen) Textur siehst, hast Du was falsch gemacht. Wenn Du weder eine statische noch die Holztextur siehst, manchmal ist die neue Textur in der nächsten Reihe weiter unten, obwohl Du bis zum Listenende gescrollt hast. Wähle eine andere Display Size und sie wird auftauchen. Das ist ein ‚scrolling'-Bug in manchen Versionen von D3Edit.



Du kannst nun Deine neue Textur wie jede andere verwenden!

Ein Schritt noch!

Wenn Du dein `.d3l` baust und die Texturen im Spiel ausprobieren willst, mußt du die `.gam`-File und die `.ogf`-Texturen in die finale `.mn3` laden. Das wird mit dem Button **Add Files** im MN3 Editor gemacht. Dann füge Dein `.d3l` normal hinzu und speichere Deine MN3 mit dem selben Namen wie die Gamdatei!

Tu die `.mn3` in Deinen `D3\missions`-Ordner, spiel Dein Level und check Deine neue Textur aus ☺ Hier ist ein Screenie der Textur in meinem Level ‚Mad Science'.

Wichtig:

- Du mußt die .gam und .mn3-Dateien JEDESMAL in D3Edit laden, wenn Du Dein Level bearbeitest. Wenn nicht, und Du speicherst den Level, werden alle Texturen zur statischen und müssen erneut aufgebracht werden.
- Die `.gam`-Datei MUSS denselben Namen wie Deine `.mn3` haben!
- Texturen können 32x32, 64x64, 128x128 und 256x256 groß sein und bis 32 bit Farben haben.
- Die Verwendung von vielen großen 256x256er Texturen kann Grafikkarten runterziehen. Es ist beim Entscheiden der Texturengröße wichtig, die Größe der Fläche zu bedenken wo die Textur draufkommen soll.

Hab Spaß am Texturieren und benutze Deine Vorstellungskraft. Vielleicht verwendest Du nie wieder Standard-D3-Texturen!

062 - Custom Texturen: Tipps

Schplurg

Ich habe hier ein wenig gekürzt. Originalversion unter: http://gameedit.net/custom/GameEdit_OG/tutorials/descent3/texturing/texturing_tips.htm

Einführung

Ich habe viel übers Texturieren gelernt, während ich 'Mad Science' gebaut habe. Obwohl ich kein Texturenmeister bin, denke ich daß ich genug wertvolle Informationen habe um eine oder zwei Webseiten zu füllen. Für mich ist das Machen von Texturen vielleicht der erfreulichste Teil beim Levelbauen. Es ist ein wichtiger Aspekt, da alles was Du siehst texturiert ist. Das innere einer Scheune kann man wie eine Kirche aussehen lassen, wenn die Texturierung gut gemacht wurde.

Um Texturen zu machen, brauchst Du ein gutes Malprogramm (Photoshop, Paint Shop Pro...). Du solltest ein ordentliches Verständnis deines Malprogramms haben, speziell wie man Layers, Klonwerkzeuge, Filter und Effekte verwendet, und Deine Vorstellungskraft.

Wo sind diese Texturen?

Woher bekommen wir sie? Angenommen, Du hast eine Idee was Du suchst, gibt es viele Methoden um zu dem zu kommen was Du brauchst. Manchmal kannst Du die komplette Textur in Photoshop designen. Allerdings sind manche Texturen ein wenig zu schwierig für eine durchschnittliche Person, um sie einfach mit der Maus zu zeichnen. Ich bin einer von Diesen. Für jene mit einem weniger als spektakulärem künstlerischen Talent gibt es alternative Methoden.

Wir brauchen etwas Grundmaterial um anzufangen. Als Texturenkünstler wirst Du entdecken, daß das Internet dein bester Kumpel ist! Du kannst mit ein wenig Geduld und einer Suchmaschine jede Art von Bild die Du brauchst finden. In meinem Level ‚Mad Science', daß in einem Haus spielt, wollte ich etwas fetzige Tapete für das Wohnzimmer. Ich hatte eine gute Idee was ich wollte, aber glaubte nicht daß ich es durch Zeichnen echt aussehen lassen könnte. Ich dachte daran, mit einer Digitalkamera Fotos von Tapeten zu machen (eine andere gültige Option), aber ich war zu ungeduldig mir eine Kamera zu borgen, ganz zu schweigen davon, das richtige Design in irgendjemandes Haus zu finden!

Also ging ich auf Google's Suchmaschine und gab „wallpaper images catalog" oder sowas ähnliches ein. Und da kamen die Webseiten der Tapetenhändler. Es brauchte ein wenig Zeit eine Seite zu finden die einen guten Online-Katalog ihrer Designs hatte, aber letztlich habe ich eine gute gefunden. Tatsächlich, auf einer Seite, habe ich tausende von netten .jgp's von Tapeten aus der ganzen Welt gefunden, in allen Farben und Mustern die ich mir vorstellen kann.

Hier sind ein paar die ich downgeloaded habe... ich habe dutzende heruntergeladen, so daß ich eine gute Auswahl habe, und für zukünftige Verwendung. Wenn Du eine gute Quelle findest, lade herunter was Du bekommen kannst!

Schließlich verwendete ich die Textur links für mein Level. Ich optimierte sie ein wenig. Ich spiegelte sie, und machte sie dann kachelbar, was nicht zu schwer war bei diesem speziellen Muster. Später dann entschied ich daß ich einen wirklich ausgefallenen Teppich für diesen Raum haben will. Ab ins Internet!

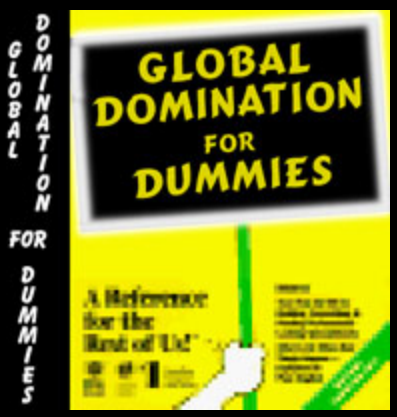
Ich denke, ich habe Nummer zwei oder ein ähnliches verwendet. Ich glaube, sie brauchte keine zusätzliche Arbeit um kachelbar zu sein. Hier und jetzt, glaub's mir später, es gibt für ALLES eine Website! Schau genau und Du wirst was finden mit einer Site, die genau das hat was Du brauchst, inklusive Game Editing Tutorials :\

Das sind keine Texturen, die Du in Descent3 inkludiert finden wirst, zumindest das kann man sagen! Es gibt ein paar Texturen-Webseiten die gratis Texturen anbieten welche bereit zur Verwendung sind, aber ich ziehe es vor mein eigenes Zeug zu machen, außerdem sind viele dieser Texturen zum kotzen oder werden von Programmen ausgespuckt, die kaleidoskopische Zufallstexturen erzeugen. DENNOCH stehe ich nicht darüber, eine existierende Textur aus einer Bibliothek als Basis für eine neue zu verwenden, oder sie einfach so unbearbeitet zu verwenden. Ich habe irgendwo ein paar nette Holztexturen gefunden, die kachelbar gemacht werden mußten (mit Photoshops Klonwerkzeug) aber außer dem habe ich keine Änderungen gemacht. Übrigens, wenn Du jemandes Anderen Arbeit verwendest, solltest Du Texturen verwenden die gratis angeboten werden oder dem Original-Autor die Credits geben, wenn er Dir die Verwendung erlaubt. Natürlich kannst Du sie trotzdem verwenden, es liegt an Dir und deinem Gewissen. Wenn Du planst Deine Arbeit zu verkaufen, zocke niemanden ab, das kann teuer werden.

Viele Bilder die ich als Basis-Bilder verwende sind einfach Fotos aus dem Internet, die ich später nach Bedarf bearbeite. Die vorigen Beispiele sind Samples aus einem Online-Katalog. Ich bezweifle, daß es ihnen was ausmacht wenn ich sie verwende (jedenfalls sage ich das). Sowieso, zur Hölle mit Juristenzeugs! Wir sind auf einer Mission (und ich mache sowieso keinen Cent, also verklagt mich)!

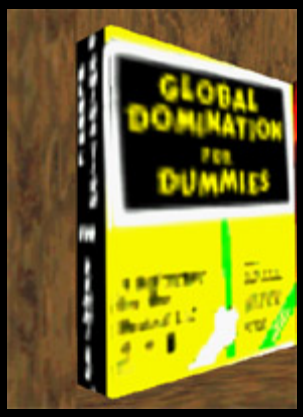
Ich habe einige Texturen von Fotos designt, welche ich mit einer Digitalkamera aufgenommen habe. Ich komme in einem zukünftigen Tutorial dazu, aber es war spaßig auf ländlichen Straßen herumzufahren und Fotos von alten Scheunen und Hütten zu machen und die Blicke in den Gesichtern der Leute zu sehen, wenn sie mich Nahaufnahmen von ihrem Zeugs machen sahen (meistens von der Sicherheit in meinem Truck aus was, ich bin mir sicher, noch verdächtiger aussah!) Was würde passieren, wenn sie D3 spielen und ihre alte Scheune in meinem Level sehen? Auch sind Fotos aus dem Web leichter zu bekommen.

Ich habe ein kleines Bild eines „Programming for Dummies"-Buches genommen und es leicht für das Regal in ‚Mad Scientist' verändert:



Unser verrückter Wissenschaftler hat in der Tat große Ziele! Die eigentliche Textur die ich im Level verwendet habe ist nur 64x64, also war Detail nicht mein Hauptanliegen. Ich wollte den Titel leserlich und das Buch wiedererkennbar und nahe am Original. Das Buchfoto daß ich heruntergeladen habe, hatte überwiegend verschwommenen Text wie Du im Bild links sehen kannst. Ich habe eine neue ‚Schiefertafel' für das Cover gemacht und meinen eigenen Text eingefügt. Die Front war nahe genug am Original und noch immer lesbar.

Eine Sache die zu beachten ist, ist daß ich diese einzelne Textur für beides, Deckel und Rücken, verwendet habe. Das verursacht Deiner Grafikkarte weniger Arbeit, da sie nicht zwei separate Texturen laden muß um das Buch zu rendern. Das Bild rechts wurde ingame aufgenommen.



In einem anderen Raum ist eine Donkey Kong Arcademaschine. Ich fand Fotos der jeweiligen Aufkleber des originalen Arcadespiels im Web, und hackte und klebte sie in nur zwei separate Texturen zusammen. Dies schließt die Seite des Gehäuses, das obere Titelpaneel, das Paneel wo die Kontrollen sind, den Münzschlitzbereich und die einfach blauen Bereiche die den Rest der Maschine bedecken ein. Anstatt einen Haufen kleiner Texturen zu verwenden, habe ich ein paar größere verwendet und dadurch ein definierteres Äußeres.
Hier links ist eine dieser Texturen. Ich zeichnete den Lautsprecher und das „25“-Ding, der Rest ist Vintage-Kong! Diese beinhaltet die Vorderfront des Gehäuses, das Kontrollpaneel und den Lautsprecher sowie den Münzschlitz.

Rechts ist das fertige Produkt (außer dem Joystick). Diese Maschine ist viele Male größer als das Schiff, daß Du im Game fliegst. Die Textur mußte gestreckt werden, obwohl ich die größtmögliche Größe verwendet habe. Daher habe ich sowenige wie möglich verwendet. Nicht schlecht oder?`

063 - Texturen kachelbar machen

Ragil Ral

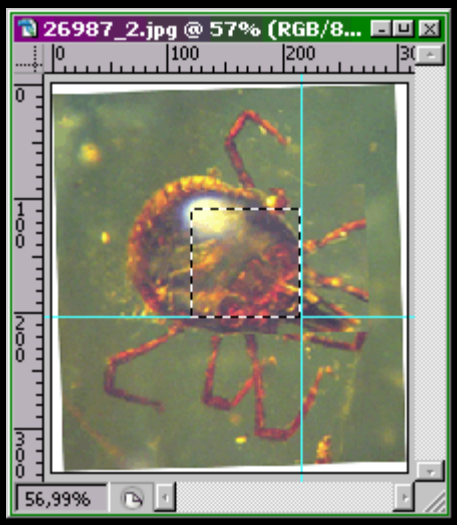
Gehört eigentlich nicht hierher, doch wer viel mit Texturen herumtut kommt nicht umhin sich zu fragen wie man am schnellsten eine Grafik oder einen Teil davon in eine Descent-Taugliche Textur bekommt. Oft braucht man halt nicht *DIE* Textur, sondern halt nur etwas das nach Fels aussieht, nach Sand oder Metall oder so, was man über eine größere Menge an Faces bzw. Terrainzellen drüberspannen kann um ein größeres Gebiet zu texturieren, ohne daß es zu monoton aussieht.

Formate und Einbinden wurden ja gerade eben behandelt, Schplurg behandelte das Gewinnen und Erstellen von Texturen; ein Problem ist aber die Kachelung: wie kriegt man das rasch gebacken?
Hier ist eine schnelle, um nicht zu sagen billige Methode dafür, allerdings ist sie leider nicht für alles geeignet.

1. Ausgangsgrafik wählen

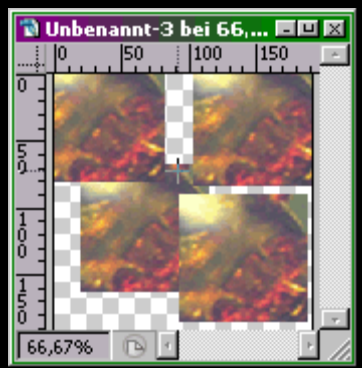
Sicher hast Du auch schon mal irgendwo im Netz Bilder oder Grafiken gesehen, wo Du Dir dachtest 'wär' geil als Textur'. Ich habe hier das Bild einer Urzeit-Zecke in Bernstein gefunden, deren metallischer Schimmer und Reflexionspunkt mich das sofort denken ließ. Wär doch irgendwie passend für einen Bot in D3, oder?

2. Formatieren

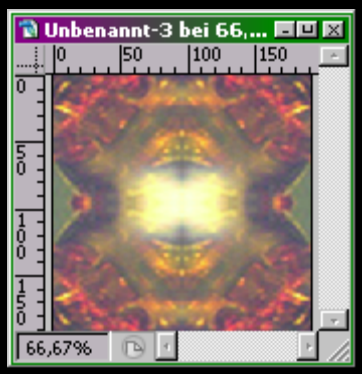


Texturen sind in D3 von quadratischer Natur. Das heißt, entweder hast Du eine Textur die an allen Seiten kachelbar ist, oder ein Viertel davon, was dann größere Flächen zu texturieren erlaubt, aber nur unter Ausrichtung seiner selbst kachelbar ist (Etwa wie die Bullseye-Texturen unter Mine). Such Dir also den Teil den Du als Textur willst (außer Dir geht es ohnehin um das gesamte Bild) und bringe ihn in ein quadratisches Format. Jetzt mußt Du entscheiden: Vollkachel oder Viertel? (Man kann natürlich auch größere Texturengruppen machen) Achte dann darauf, daß die Auflösung und Größe (in Pixel) der Grafik (bzw. den Teil um den es Dir geht) nicht unter der Texturengröße liegt, da Du sie sonst vergrößern mußt - was je nach Bearbeitungsprogramm sehr unschöne Artefakte in der Grafik erzeugen kann. Bei `.jpg`'s als Ausgangsmaterial achte unbedingt auf die Qualität; die jpeg-Komprimierung verursacht selbst schon solche Artefakte, wenn sie falsch angewandt wurde. Öffne also die Grafik mit einem Grafikbearbeitungsprogramm Deiner Wahl. Eventuell noch vorbearbeiten, so daß sie Deinen Vorstellungen entspricht. Kopiere dann den gewünschten Teil aus dem Bild heraus; günstig ist es wenn man einstellen Kann daß die Auswahl quatratisch ist, wie etwa im Photoshop. Geht aber meist auch, wenn man Während des Markierens **Shift** drückt.

Viertelkachel: in diesem Fall einfach in eine neue Datei einfügen, deren Pixelmaße dem kopierten Teil entsprechen. Beim Texturieren in D3Edit wirst Du dann in den Buttons **Flip X**, **Rot 90** und **Flip Y** unter Align schnell neue Freunde finden 😁



Kachel:
Aurch hier fügst Du den kopierten Teil in eine neue Datei ein, aber achte darauf daß der Arbeitsbereich genau doppelt so groß ist wie der eingefügte Teil. Zwecks angenehmeren Arbeiten: nicht alle Grafikprogramme erlauben es, Grafiken auch außerhalb des Arbeitsbereiches zu manipulieren - in diesem Falle nimm einfach etwa die vierfache Größe, so hast Du Platz zum 'manövrieren'.



Den Teil den Du in D3Edit mit den Buttons unter Align erledigst, machst Du jetzt hier: füge den kopierten Teil insgesamt viermal ein. Spiegle bzw. rotiere die jeweiligen Stücke derart, daß sie sich aneinanderfügen lassen und setze sie nahtlos zusammen. Mit Rotation alleine funkt das nicht. Mit spiegeln bist Du am besten dran!
Nun hab ich meine Textur. Ich werd die als Light und Forcefield verwenden.

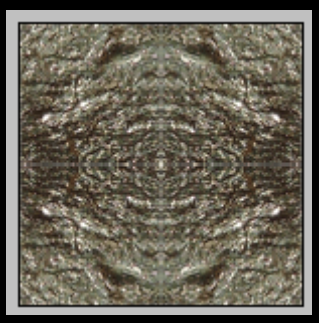
Für den Fall, daß Du später draufkommst daß Du was anders haben willst, speichere die Arbeitsdateien in jedem Falle ab, denn nachher löschen ist einfacher als schimpfen weil man's zu früh gelöscht hat und dann nochmal machen darf ☺

Falls Dein Grafikprogramm mit Ebenen arbeitet, reduziere nun alle Ebenen auf eine. Das Bild dann auf D3-taugliche Größe herunterskalieren (ich rate wirklich dringend davon ab irgendwas raufzuskalieren, das sieht in der Regel echt wirklich grottig aus), mit D3-tauglichen Werten als `.tga` speichern, mit dem OGF Tool ins `.ogf` umwandeln, in die `.gam` einfügen, eventuell Flags setzen, feddich.

Begucke Dir die Textur immer ingame! Etwas, was außerhalb gut aussieht muß nicht notwendigerweise auch ingame gut aussehen. Eventuell mußt Du mit den Farbwerten, Alpha-Kanal oder sonstwas herumspielen bis es paßt, von Lichtwerten mal ganz abgesehen.

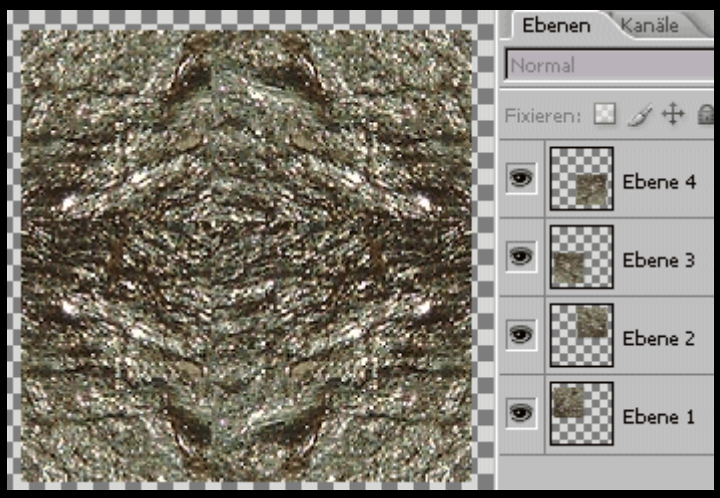
Theorie: Durch das Einsetzen der Viertel und deren korrekte Ausrichtung/Positionierung ergibt es sich wegen der Spiegelsymmetrie - die eine Eigenschaft dieses (current ☺) Universums ist - ganz easy, daß die Kanten jeweils zueinanderpassen - das Ding wird damit kachelbar. Das geht mit wirklich allen Grafiken, da die Textur ja eine in-sich-selbst-Kopie ist und die Ränder daher immer passen werden. Schlimmstenfalls, wenn Du ein ungeeignetes Motiv gewählt hast, kann man halt die 'Fliesenfugen' sehen, aber die Symmetrie wird immer erhalten bleiben. Es leben die mathematischen Naturgesetze, hehe 😜.

Kachelbar ohne Symmetrien



Ein Problem ist, daß diese Methode stark auffällige Symmetrien entstehen läßt(rechts). Um das zu beseitigen stelle Dich schon mal auf Arbeit ein...

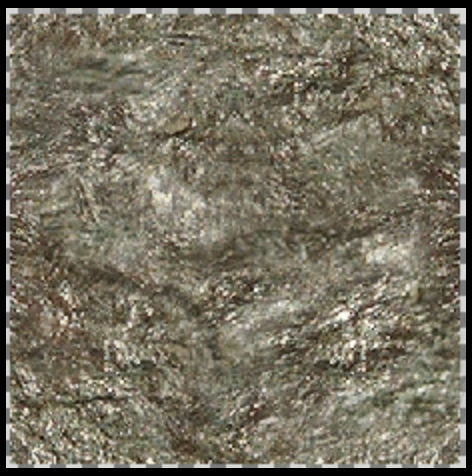
Abhilfe:



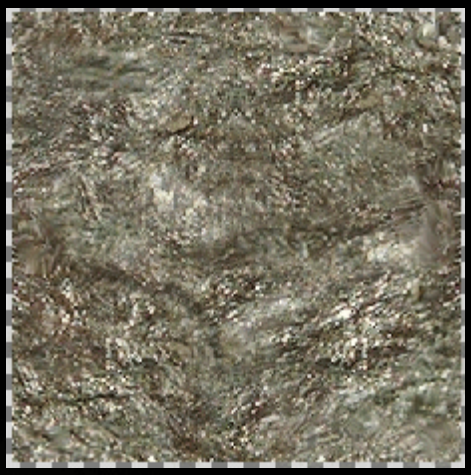
Schiebt die Einzelteile erstmal ineinander, achtet aber darauf daß der Grundumfang Quadratisch bleibt (links):

Nun wirds ein wenig schwieriger:
Greift euch ein Löschwerkzeug mit weicher Kante, und radiert Teile am Rand der jeweiligen Viertel aus. Achtet darauf den Kanten nicht zu nahe zu kommen, da sonst die Kachelung leidet, und achtet auf die Ebene, in der ihr gerade seid.
Dadurch brecht Ihr die Symmetrie ein wenig und die harten Kanten verschwinden (rechts).

Als nächstes auf eine Ebene reduzieren.



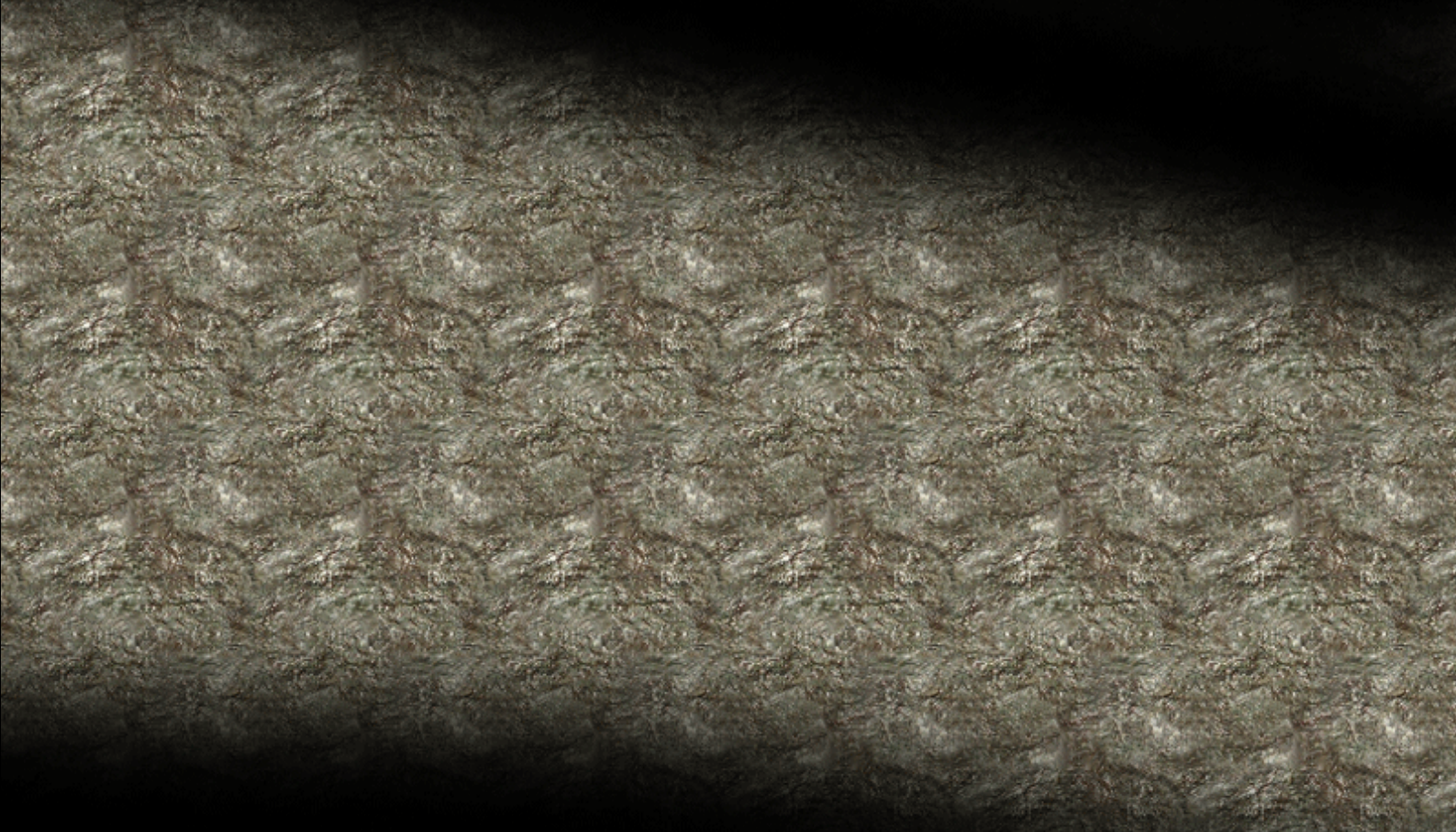
Dann holt euch einen Klonstempel und verändert die Teile, die doppelt erscheinen. Falls Euer Malprogramm das erlaubt, holt euch die Quelle aus dem Bild, von dem Ihr den Texturenteil herhabt. Ihr werdet merken, je näher ihr den Rändern kommt, umso schwieriger wird es (links). Ihr werdet vielleicht fast das ganze Bild übermalen müssen.



Beim Kacheln würde jetzt nur noch an den Stoßrändern Symmetrie auftauchen. Um die noch loszuwerden macht folgendes:
Nehmt Euch den Wischfinger (oder ein ähnliches Werkzeug) und bearbeitet nun noch jeweils einen oberen und seitlichen Rand, sodaß sich auch die noch unterscheiden (rechts):

Falls mal etwas unscharf wird ist das nicht soo tragisch, das Bild wird ohnehin auf Texturengröße zusammenskaliert.

Jetzt Trick 17: Skaliert das Bild auf 12**9** (also einen Pixel mehr) herunter. Hernach beschneidet es jeweils an einem Rand (oben oder unten und links oder rechts) um einen Pixel; dadurch könnt ihr vermeiden, daß die Randpixel doppelt dargestellt werden und die Größe beträgt wieder die amtlichen 128x128 Pixel.



Die Textur wird ja wiederholt, also gehts (leider) nicht ganz
ohne Symmetrien. Dagegen hilft wiederum, die Textur eher strukturlos zu designen.
Seid ebenfalls kreativ beim Bearbeiten, also versucht immer wieder mal andere Einstellungen der Werkzeuge bzw überhaupt andere Werkzeuge.

Kaum zu glauben, wieviel Text man mit sowas erzeugen kann :o)

Einstweilen Euch ein T-Jointfreies Bauen!

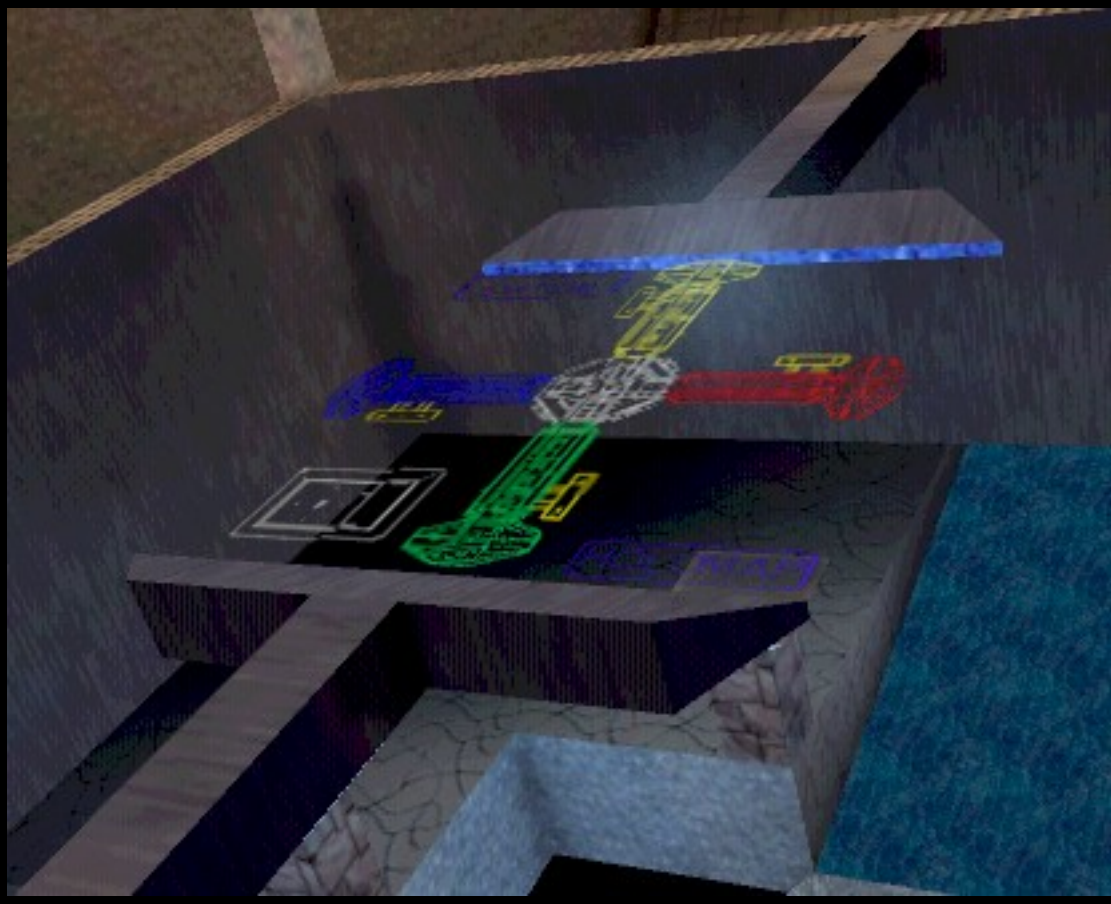
064 - Teiltransparente Texturen

DigiJo

Diese Tutorial handelt speziell teiltransparente Texturen in Descent 3 Levels ab. Für Basisinformationen über custom Textures in Descent 3 gibt es hier bei Fischlein schon reichlich Infos, darum will Ich mich hier mal auschließlich der Teiltransparenz widmen:
Zu erst mal zum Grundsätzlichen, was sind Teiltransparente Texturen eigentlich? Teiltransparente Texturen können einige nette Effekte erzielen z.B. Pflanzen (Buschwerk, Baumblätter) , Zäune, Fenster etc und viele mehr, das bedeutet das Teile dieser Texturen durchsichtig sind und den Blick auf dahinter liegende Teile des Level ermöglichen. Im Prinzip ist es das selbe wie mit transparenten GIF-Files, die z.B. in eine Website eingebunden den Blick auf eine Hintergrundgrafik ermöglichen. Der Vorteil dieser Methode liegt eindeutig im Einsparen von Faces / Polygonen, da z.b das erstellen eines Gartenzauns oder Baumes mit Polygonen / Faces einige 100 Polygone erfordern würde, mit teiltransparenten Texturen ist ein ähnlicher Effekt mit vieleicht nur 10 oder 15 Faces möglich, auch wenn es nicht so gut aussieht.
Ein Nachteil dieser Methode sei auch erwähnt, so wie es scheint funktioniert diese Methode aufgrund eines Fehlers in Descent 3 nur auf neueren Grafikkarten, wenn diese im OpenGL-Mode betrieben werden. Benutzer älterer Voodoo-Karten mit Glide oder Nutzer des D3D-Modes sehen statt des transparenten Textur-teils eine schwarze Fläche. Dieser Fehler scheint nur aufzutreten, wenn die transparenten Texturen auf Strukturfaces gelegt werden, auf Objekten oder (z.B als Mond wie der IO-Mond) am Terrainhimmel funktionieren sie überall.
Meiner Meinng nach kann man diese Texturen trozdem gut verwenden da fast alle Descent 3 Spieler neuere Grafikkarten im OGL-Mode verwenden. (D3D) in Descent3 ist echt Müll ☺)

Beispiele

Hier haben wir mal ein Beispiel für Teiltransparente Texturen, diese Holomap aus Monolith-Station besteht allerdings wegen der höheren Auflösung aus 8 Einzeltexturen (4 auf der Vorderseite und 4 gespiegelt auf der Rückseite) und hat zusätzlich noch eine Gesamt-Transparenz, dazu später mehr.



Noch ein schönes Beispiel, diesmal ein Baum aus Tower of Isengard. Hier kann man schön die "Löcher" im Blätterwald erkennen, diese Bäume haben pro Stück nur 16 Faces und schonen die Framerate...

Basics

Jetzt mal zum technischen Teil, Descent3 kennt grundsätzlich 2 Texturarten, das `.ogf`-Format und das `.oaf` Format. OGF wird für Standard-Texturen verwendent und bedeutet "Outrage Grafic Format", OAF wird für animierte Texturen verwendet und bedeutet "Outrage animated Grafic Format" oder ähnlich. OAF´s sind nichts anderes als hintereinander geschaltete OGF´s. Beide Formate unterstützen Teiltranzparenz. Die Grafikdaten in diesen Grafikformaten sind wiederum komprimiert mit der sogenannten ARGB-Kompression, Descent3 kennt 2 Versionen, die 1555 ARGB und die 4444 ARGB. Die Buchstabenkürzel ARGB stehen hierbei für:

- A=Alphakanal
- R=Rotkanal
- G=Grünkanal
- B=Blaukanal

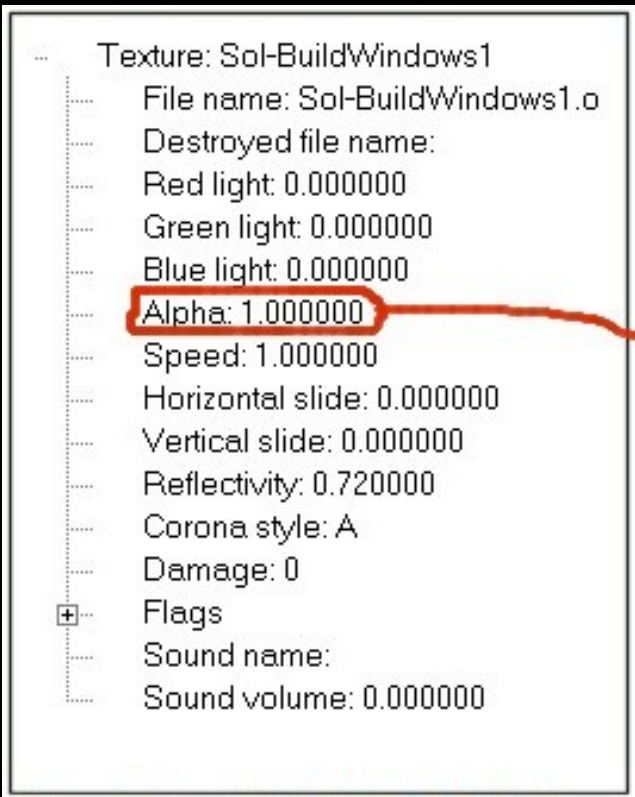
die Zahlen davor stehen für die jeweilige Bit-Tiefe der einzelnen Kanäle:

- 1555=1 Bit Alpha, 5 Bit Rot, 5 Bit Grün, 5 Bit Blau
- 4444=4 (3) Bit Alpha, 4 Bit Rot, 4 Bit Grün, 4 Bit Blau

Interresant ist hier der Alphakanal, weil der festlegt, welches Pixel transparent erscheint und welches Pixel nicht transparent ist. Bei der 1555 ARGB Kompression gibt es nur 1 Bit für den Alphakanal, das bedeutet das man jedes Pixel entweder voll transparent schalten kann oder voll sichtbar. Mit der 4444 ARGB-Kompression hab ich noch nicht experimentiert, sie legt aber eine Transparenz für jeden Farbkanal an. Das bedeutet das z.B. von einem dahinter liegenden Face oder Objekt nur der Rot-Anteil oder nur der Blau-Anteil durchscheint.

Gesamttransparenz

Auf die Gesamttransparenz von Texturen in Descent3 will ich nur kurz eingehen, zusätzlich zur Teiltransparenz über den Alphakanal in der Textur selber kennt Descent3 auch noch eine einstellbare Gesamttransparenz für jede Textur über den Gametable-Eintrag "`ALPHA`" der entsprechenden Textur, egal ob diese Textur nun einen Alphakanal hat oder nicht. Der Alphawert in der Gametable ist einstellbar von 0.0 bis 1.0 wobei 1.0 voll sichtbar bedeutet und bei einen Wert von 0.0 die Textur unsichtbar ist. Bei einem Wert von 0.5 ist sie halb durchlässig wie z.B. bei einer Glasmalerei. Dieser Wert gilt dann grundsätzlich für alle Pixel der Textur.



An der rot markierten Stelle in der Gametable wird der Alphawert für die Gesamttransparenz der Textur eingestellt, das geht bei jeder Textur und gilt grundsätzlich für alle Pixel der Textur.

Erstellen teiltransparenter Texturen

Zum erstellen teiltransparenter Texturen braucht man grundsätzlich 2 Dinge: ein Grafikprogramm das Alphakanäle und das "Tagged Image File Format" (TGA) unterstüzt, und ein Program zum konvertieren der Textur nach OGF (oder OAF). Descent3 liefert da das OGF-Tool (ogfool.exe) mit, viel besser ist aber das D3 Imagetool Ver.1.03 von Supersheep. Das gibts natürlich auch bei Fischlein. Für die Bearbeitung der Grafiken nehme ich meistens das Grafikprogramm Paintshop Pro 7 (PSP7), ist einfach genial das Proggy.

Erstmal braucht man ein entsprechendes Motiv aus dem Internet (gibt ja millionen von Vorlagen überall zu finden) oder zeichnet eine eigene Textur (entsprechende Begabung vorrausgesetzt ☺).

Die Grafik wird auf Texturgröße zurechtgeschnitten und/oder resized, sie sollte dann quadratisch sein (Höhe=Breite).

Brauchbare Formate (in Pixel):

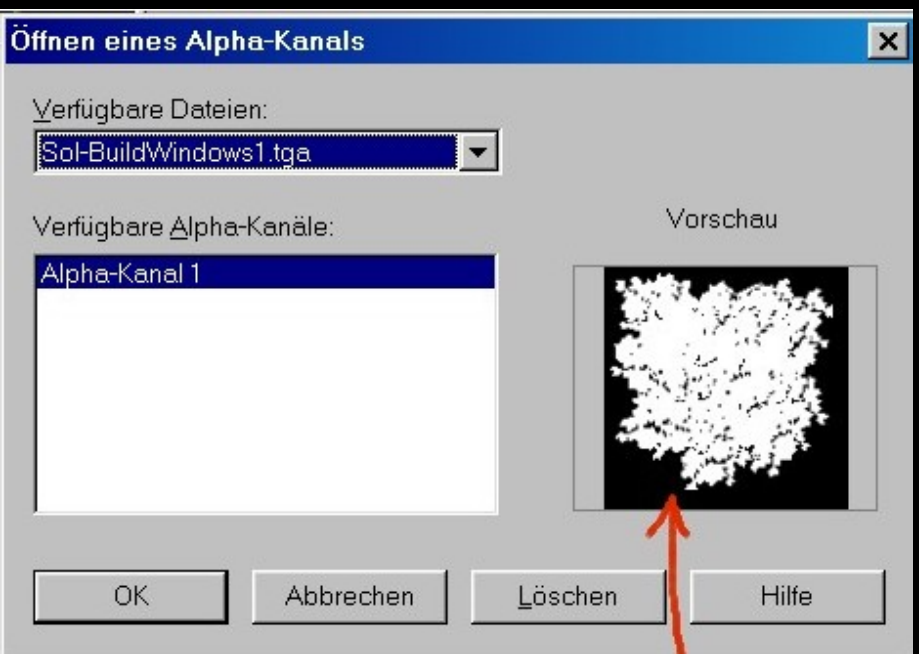
- 32x32
- 64x64
- 128x128
- 256x256

Andere Formate sind meines Wissens auch möglich aber nicht zu empfehlen. Kurz zum TGA Format, diese Format ist optimal für das erstellen von Texturen geeignet da es von allen Konverter-Programmen unterstüzt wird und genau das bietet was man für eine Textur braucht, bis zu 32bit Farbtiefe und einen 8 Bit Alphakanal.

Das rechts hier ist die Vorlage, die ich für das Blätterwerk in den "Tower of Isengard" Bäumen verwendet habe. Diese Textur ist schon zurechtgeschnitten auf 128x128 Pixel und hat einen schwarzen Hintergrund.

Nun soll ja später im Descent3-Level alles was jetzt schwarz ist transparent sein, dafür brauchen wir einen Alphakanal. Zu diesem Zweck habe ich in PSP7 eine Maske erstellt in der jedes schwarze Pixel enthalten ist und diese Maske als Alphakanal gespeichert.

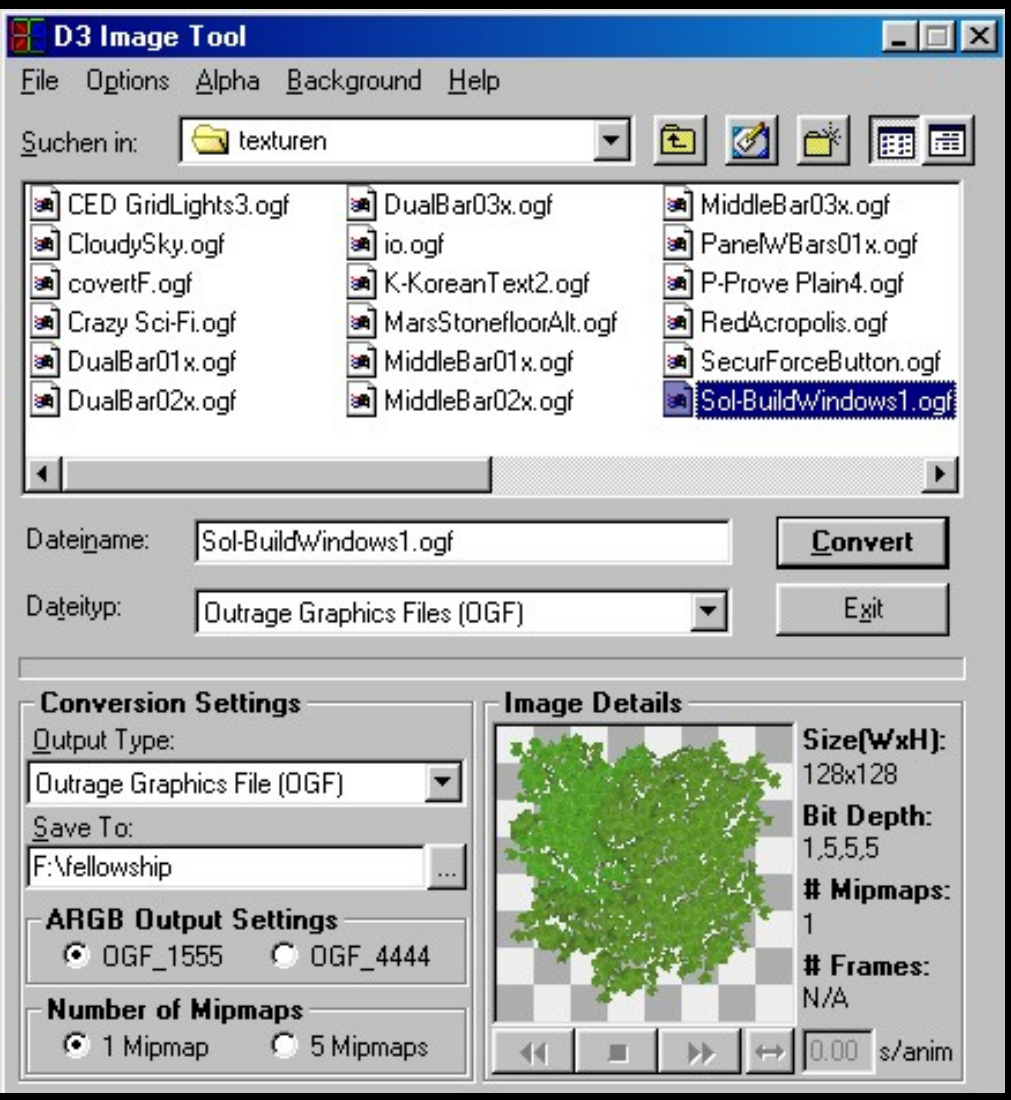
Hierzu habe ich mit PSP7 eine CMYK-Trennung vorgenommen und den Schwarz-Kanal auf 2 Bit Farbtiefe reduziert. Der Schwarz-Kanal wird dann noch invertiert und als Maske gespeichet. Dieser Alphakanal hat nur 1 Bit Tiefe (Nicht Transparent / Voll Transparent) da wir das File ja später als OGF (1555 ARGB) konvertieren wollen.



In diesem PSP7-Menü links wird der als Maske gespeicherte Alphakanal hinzugefügt, rechts sieht man schön die Vorschau des Kanals. Später im TGA und in der fertigen OGF-Textur wird alles schwarze in dem Alphakanal transparent sein.

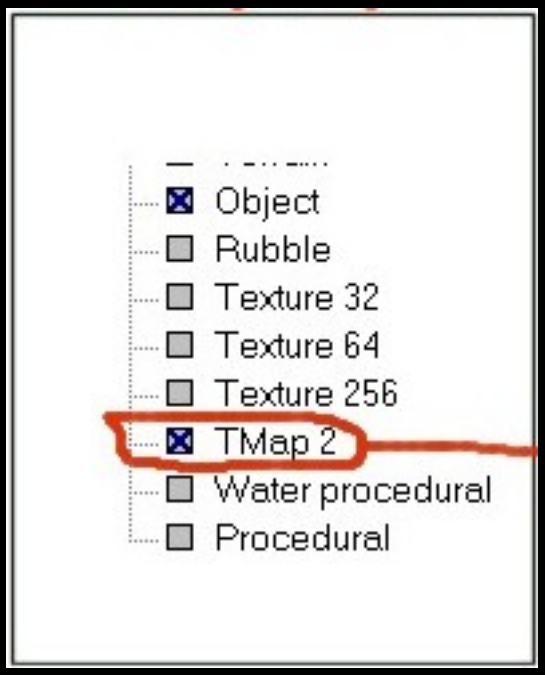
Hier rechts sieht man nun wie das fertige TGA mit angezeigtem Alphakanal aussieht. Der Alphakanal ist wie bei einem GIF-File in dem TGA-File mit integriert und braucht nicht extra gespeichert zu werden.

Nachdem man die Textur nun als TGA-File abgespeichert hat (ich hab hier als Dateinamen Sol-BuildWindows1.tga gewählt da ich später eine entsprechende Datei in Descent3 ersetzen will) fehlt nur noch das konvertieren. Zum konvertieren verwende ich das oben schon genannte Program D3Imagetool, das mitgelieferte OGF-Tool tuts aber auch.



In diesem Screenshot von D3Imagetool hat man einen guten Überblick über die Einstellungen. Oben wird das Inputfile ausgewählt (Muss naürlich das TGA sein, Ich hab hier für den Screenshot aus Versehen das fertige OGF gewählt), unten rechts sieht man die Vorschau und die Bildparameter, unten links ist der interessante Bereich mit den Kompressions-Einstellungen. Für diese Texturen wählen wir 1555ARGB, also 1 Bit Alphachannel. Es wäre auch möglich 4444ARGB Kompression zu wählen, da der Alphakanal des TGA aber nur 1 Bit Tiefe hat wäre kein Unterschied zur 1555 zu sehen. Mipmap-Level ist eigentlich egal, da bei nur einer MipMap die restlichen Maps in der Grafikkarte erzeugt werden. 1 oder 5 ist nur ne Frage von Ladezeit und Qualität der Grafikkarte.

Jetzt braucht man nur noch convert zu klicken und fertig ist die Blättertextur mit Teiltransparenz. Zum einbinden der Textur in Descent3-Level muß noch erwähnt werden, daß es in dem entsprechenden Gametable-Eintrag der Textur ein Flag gibt, das TMAP2-Flag. Diese Flag zeigt Descent3 an das die Textur über einen Alphakanal verfügt und benutzt werden soll, also das aktivieren des TMAP2-Flags nicht vergessen. Soll eine Textur einfach nur ersetzt werden um sich den Gametable-Eintrag zu sparen (siehe Linux und MAC-Kompatiblität) sollte man sich eine Textur zum ersetzen suchen, die dieses Flag gesetzt hat. (In diesem Fall müssen die Eigenschaften der Original-Textur und der custom Textur exakt gleich sein, siehe ARGB, Mipmap und Auflösung). Die Sol-BuildWindows Reihe z.B. hat TMAP2-Flags.



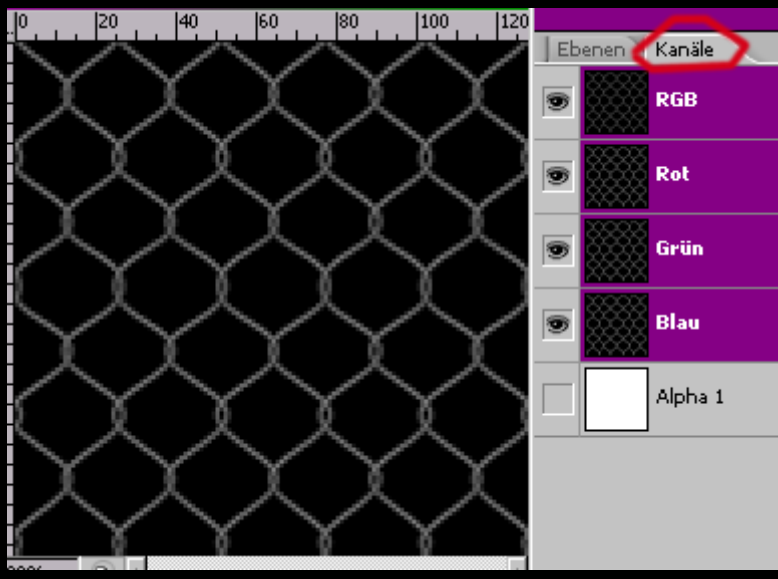
Dies ist das TMAP2-Flag in der Gametable, das Descent3 angibt ob ein Alphakanal in der Textur vorhanden ist und ob dieser angezeigt werden soll. Komischerweise hab ich es auch schon gehabt, das selbst ohne diese Flag der Alphakanal angezeigt wird, aber sicher ist sicher. (D3 scheint allgemein Probleme mit transparenten /animierten Texturen zu haben, siehe Bugliste 1.4)

Ich hoffe mal das dieser kleine Artikel nützlich ist für die Levelbauerzunft.

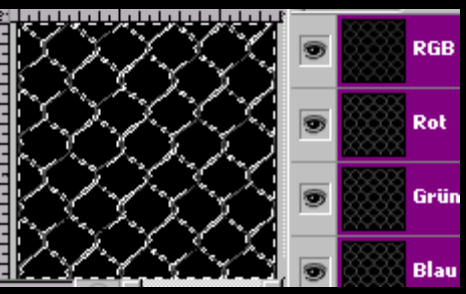
in diesem Sinne wünsche ich euch ein T_Joint freies Levelbauen,
[NuB2]DigiJo (digijo@gmx.de)

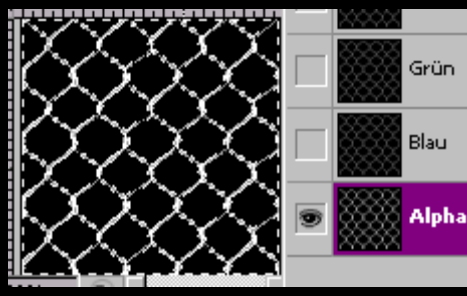
Nachtrag für Photoshop:

Im PS funktiniert das ganze so, daß die Kanäle sich in der Ebenenpalette befinden. Das Bild muß als 32-Bit `.tga` vorliegen (rechts)



In den Farbkanälen wählt nun mit dem Zauberstab jene Bereiche aus die Ihr transparent haben wollt. Dem Zauberstab gebt auf jeden Fall die Einstellung ‚Glätten', sodaß die Grenzen zwischen opak und transparent nicht zu hart ausfallen (links).





Dann klickt Ihr auf die Alpha-Ebene und drückt dort drin die Taste **Delete**; damit habt Ihr Eure Transparenzmaske erzeugt. Speichert nun die `.tga`-Datei.

Der Rest wie gehabt... wandeln zu `.ogf`, einbinden in eine `.gam`, diese als `.mn3` mit der `.ogf` zusammenspeichern, im Editor laden etc. etc.

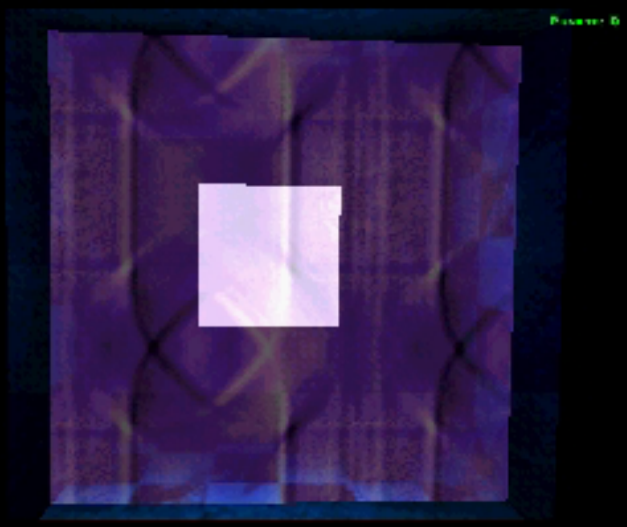
Ergebnis: Maschendrahtzaun in D3 😁
Wenn man aber genau hinguckt, sieht man daß ich die Transparenzmaske wohl nochmal machen muß...

065 - Animierte Texturen - Screenshots

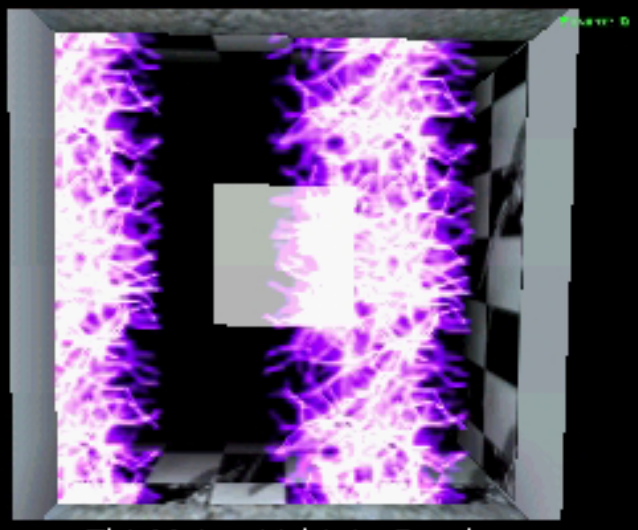
(LL)Dark

Die folgenden Bilder zeigen die Animierten Texturen die man in D3Edit nur als "Rainbow-Textur" oder fehlerhaft dargestellt bekommt. Am weißen Viereck im Hintergrund kann man jeweils die Transparenten Texturen erkennen.

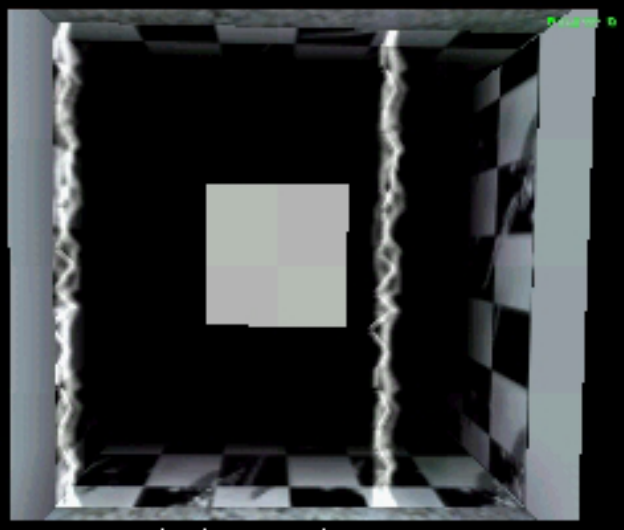
Tankwater
Light: 0.1 / 0.7 / 0.6 - normal,bewegte sich kraeuselnde Oberfläche mit seitlich darueber - laufenden Wellenfronten, dampft bei Beschuss



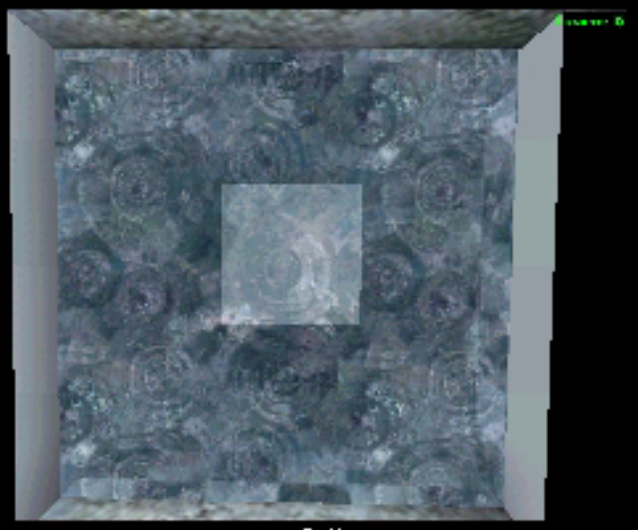
SS - Forcefield1
Light: 0.0 / 4.0 / 8.4 - Schild, sich ueberschneidende Kreise und Wellenfronten



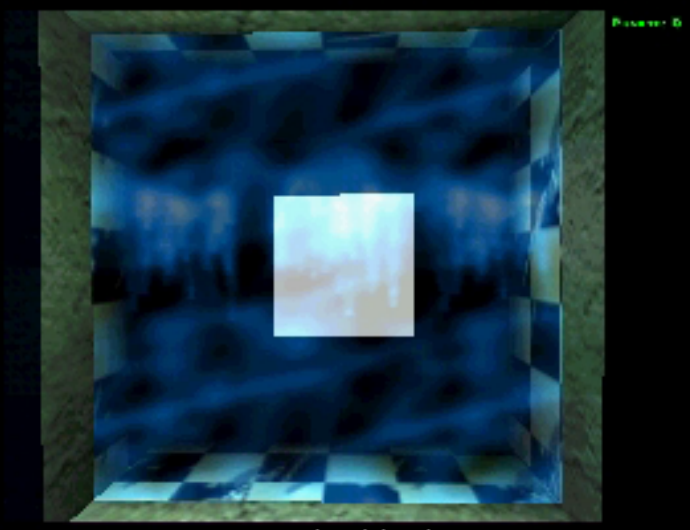
ThinMatcenLightningPurple
Light: no Light - normal, von unten nach oben durchlaufende Straenge von Blitzen



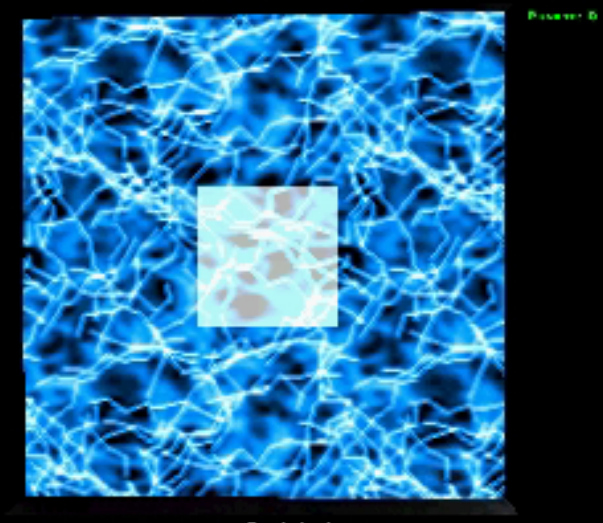
ThickLineLightning
Light: no Light - normal, schmale bänder die sich bewegen wie elektrische Entladungen



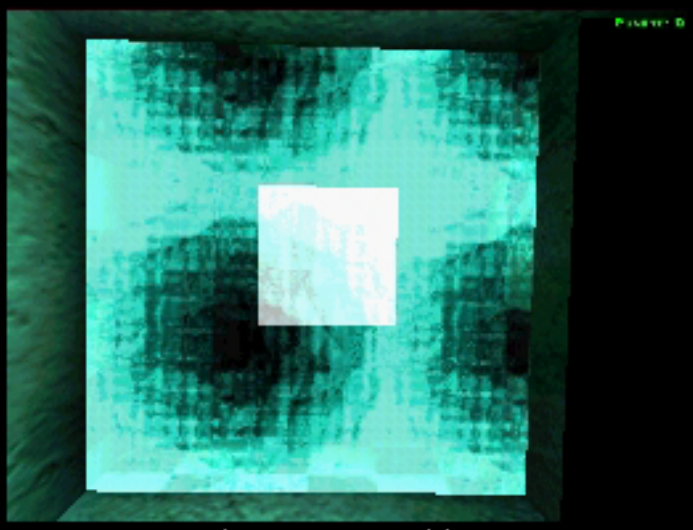
Waterfall1
Light: no Light - normal, von oben nach unten fliessend mit Tropfkreisen,dampft bei Waffenbeschuss



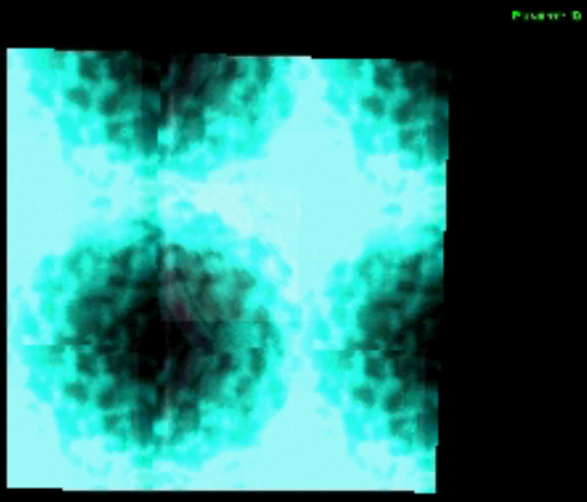
EXP_Shield_alter
Light: 15.0 / 20.0 / 15.0 -Schild, aufblitzende sich kreuzende Linien und Fontaenen , langsam von unten nach oben durchlaufend



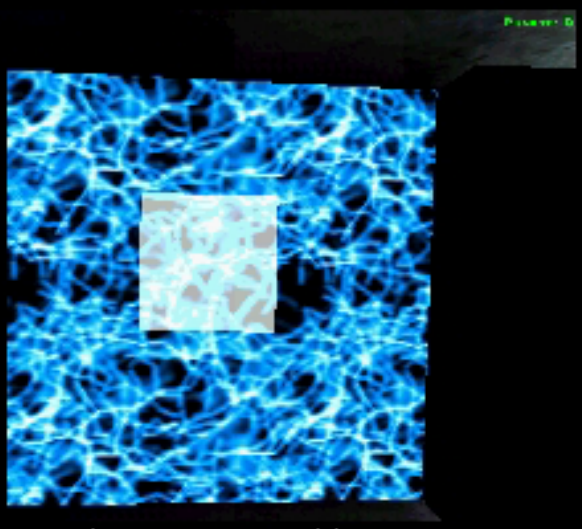
ForcefieldBlue
Light: 0.0 / 2.0 / 4.0 - Schild, durcheinanderzuckende Blitze , von links nach rechts durchlaufend



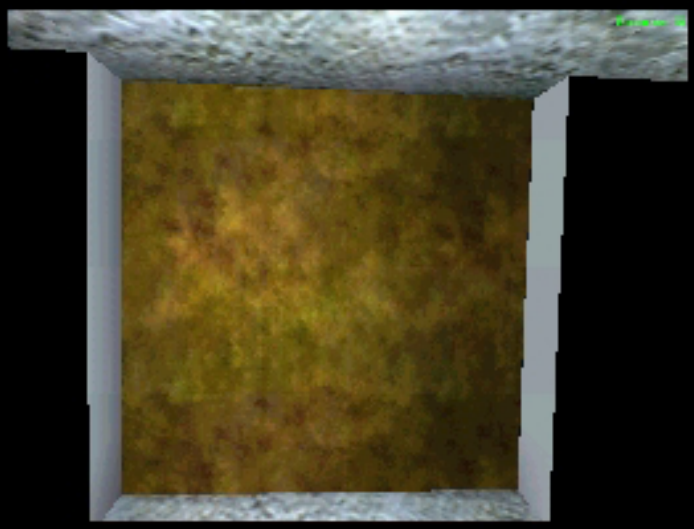
Alien Force Field_1
Light: 4.0 / 10.0 / 8.0 - Schild, schnell bewegte sich konzentrisch überschneidende Kreise



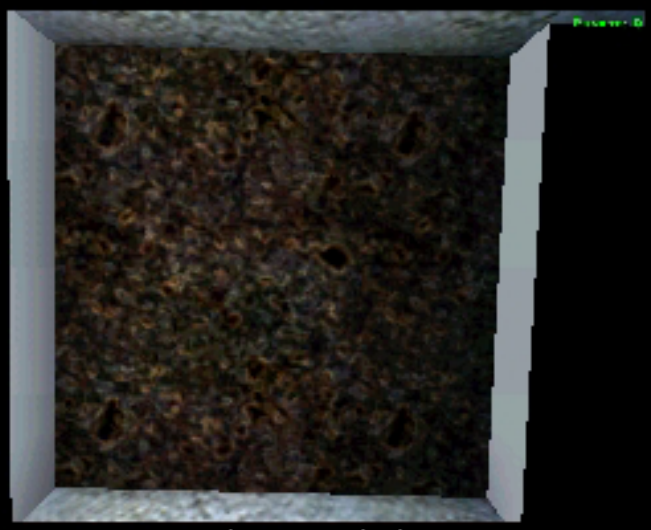
ALIENCPULINK
Light: 0.0 / 3.0 / 2.0 - normal , mittelschnelll auseinanderlaufende Wellenfronten



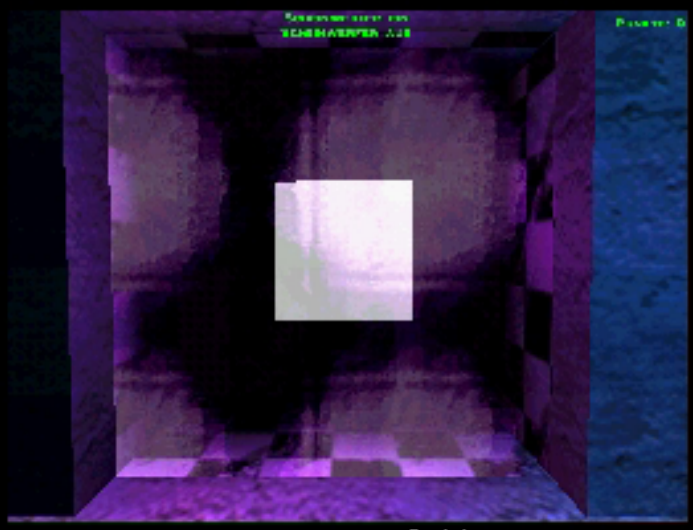
BlueMagneticField - V
Light: 1.0 / 1.5 / 2.0 - Schild, mittelschnell durcheinanderzuckende Blitze



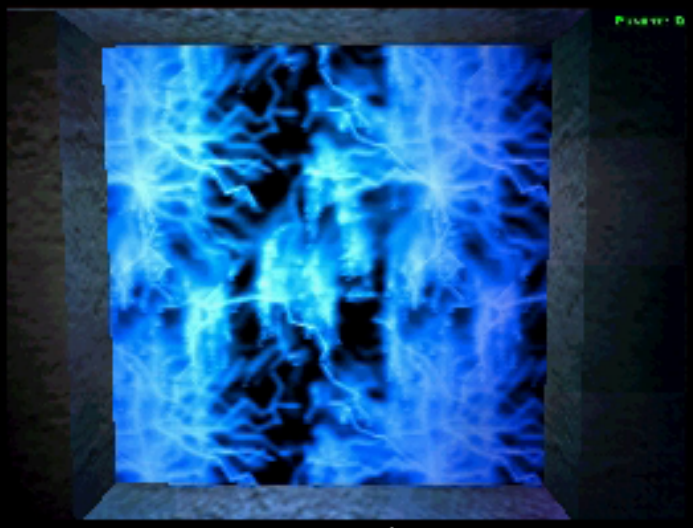
BrownSewage
Light: no Light - Saeureschaden, lnagsam von oben nach unten fließend mit leichten seitlichen Wellenfronten, dampft bei Waffenbeschuss



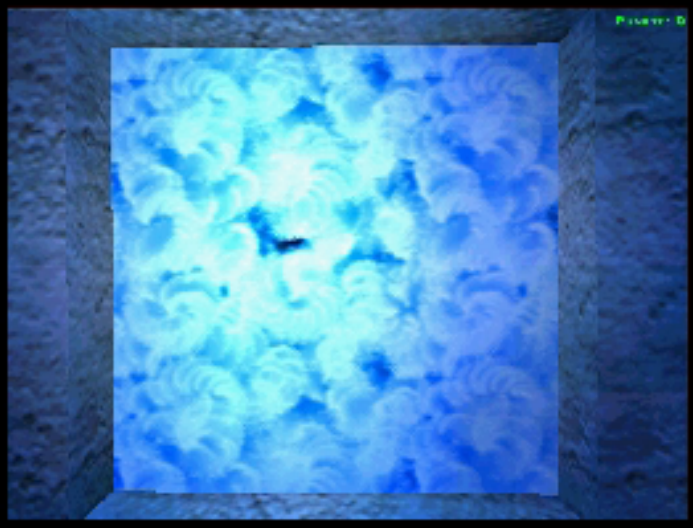
BrownVolcanoRockShimmer
Light: no Light - normal , enthaelt kleinere (hitze) wabernde Stellen



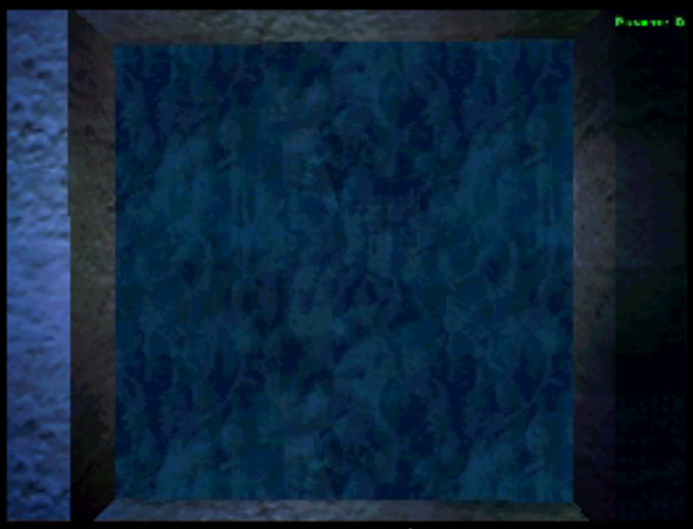
CED - Forcefield
Light: 15.0 / 5.0 / 20.0 - Schild , bewegte sich ueberschneidende Kreise und Wellenfronten



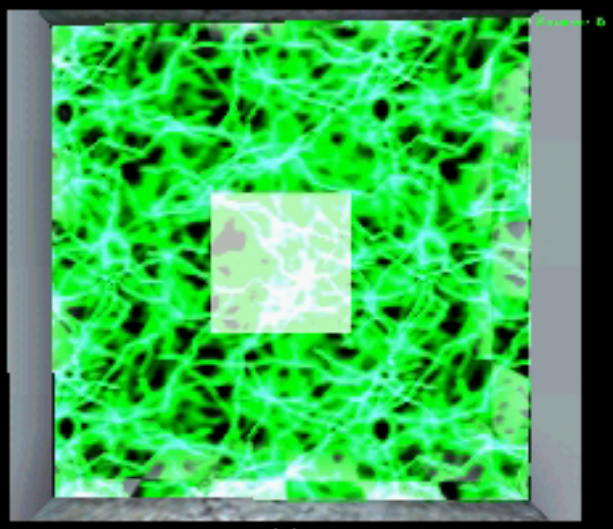
CED_CoreSkin01
Light: 2.0 / 5.0 / 9.0 - normal,von Punkten ausgehende Blitze mit herabrieselnden Fontaenen dazwischen



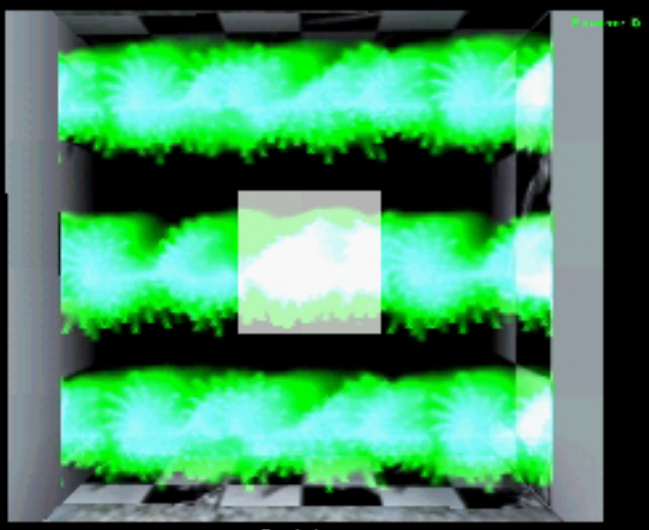
EnergyConvpro
Light: 20.0 / 50.0 / 100.0 - normal, sich ringelnde Energieentladungen (oder Plasma) von links nach rechts durchlaufend



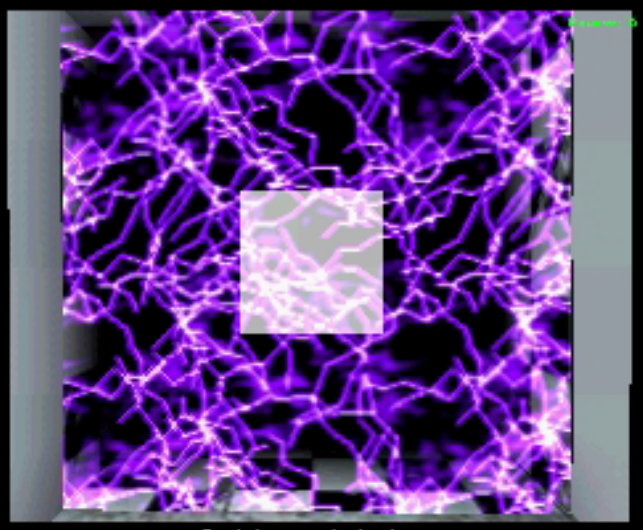
EuropaWaterGlowing
Light: 1.5 / 2.0 / 3.0 - normal, als ob kleine Tropfen ins Wasser fallen, dampft bei Waffenbeschuss



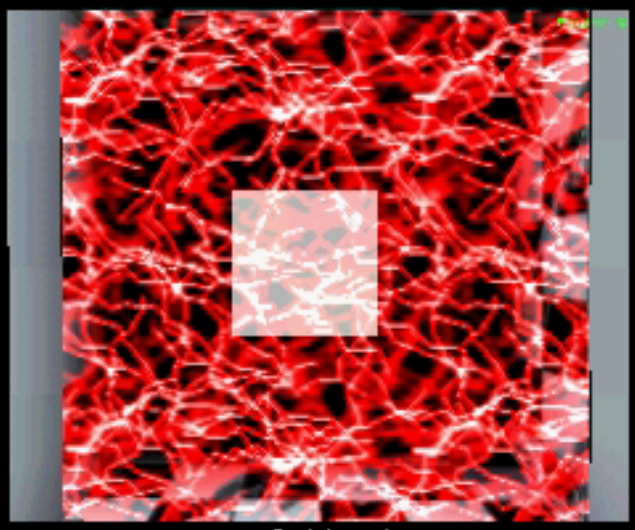
ForceFieldGreen
Light: no Light - normal, durcheinanderzuckende Blitze



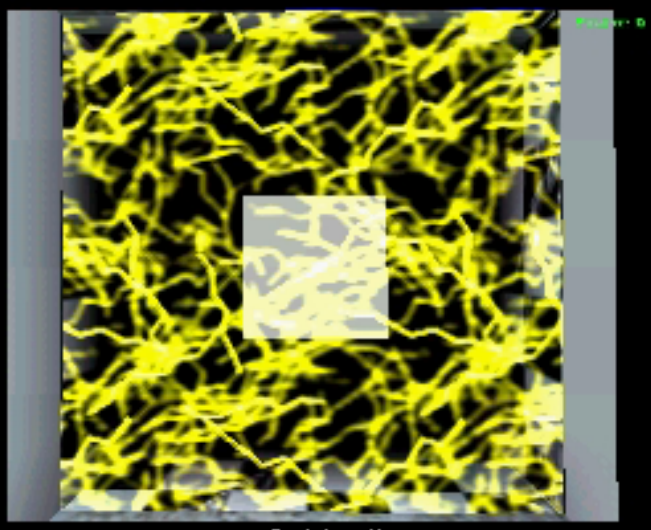
ForcefieldGreen2
Light: no Light - normal,sich drehende Spiralen aus kleinen Blitzen



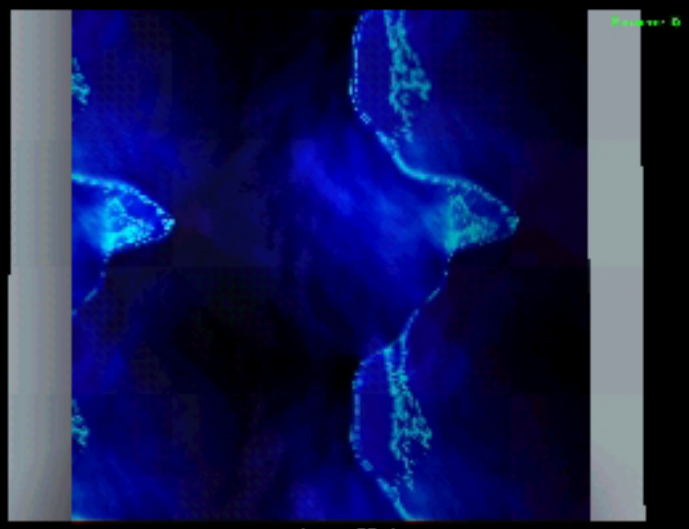
ForcefieldPurplelightning
Light: no Light - normal, durcheinanderzuckende Blitze, langsam von unten nach oben durchlaufend



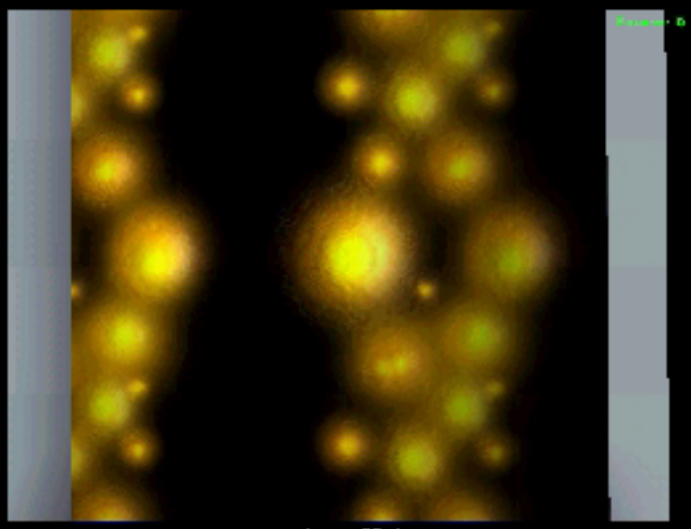
ForcefieldRed
Light: no Light - normal, sehr schnell durcheinanderzuckende Blitze



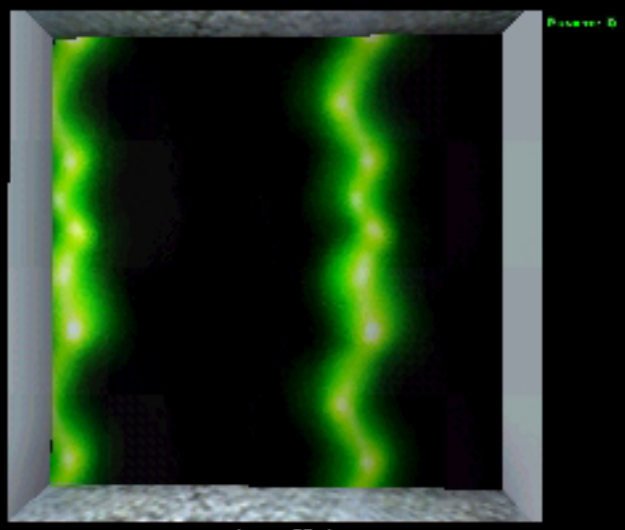
ForcefieldYellow
Light: no Light - normal, durcheinanderzuckende Blitze



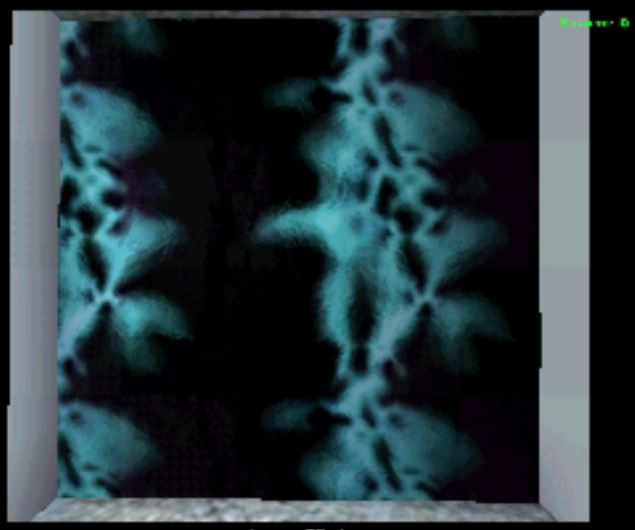
FunkyEffekt1
Light: no Light - normal, an den Strängen die zu sehen sind entlangkriechende el.-Entladungen



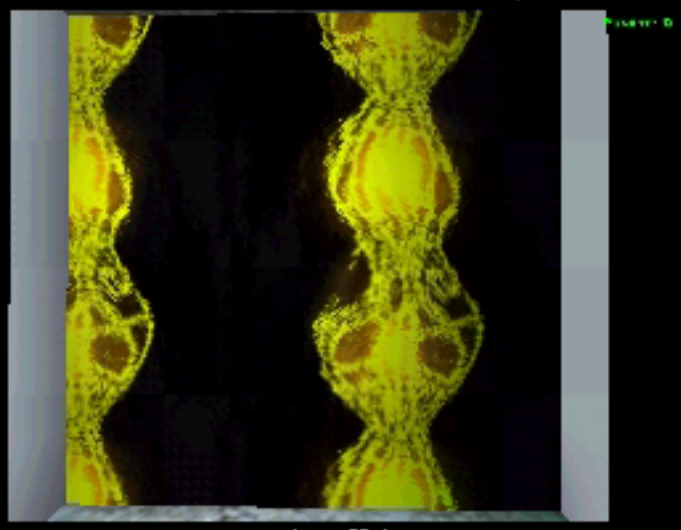
FunkyEffekt3
Light: no Light - normal, stillstehend, mit sanft nach aussen dringender Strahlung (wie soll man das sonst beschreiben ...)



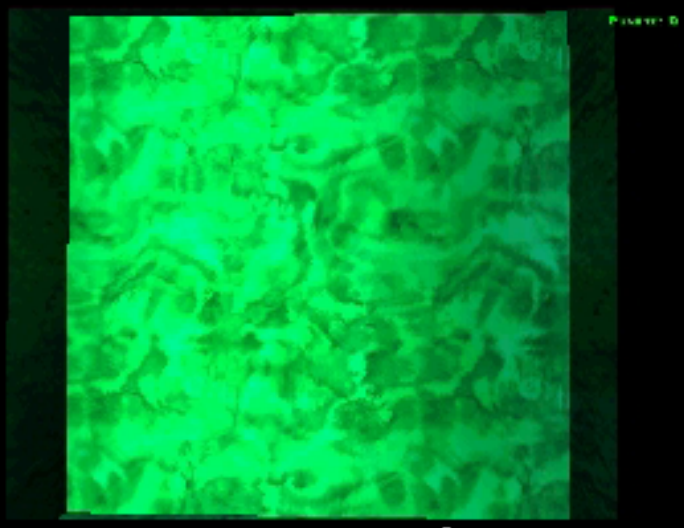
FunkyEffekt4
Light: no Light - normal, stillstehend, mit sanft nach aussen dringender Strahlung (wie soll man das sonst beschreiben ...)



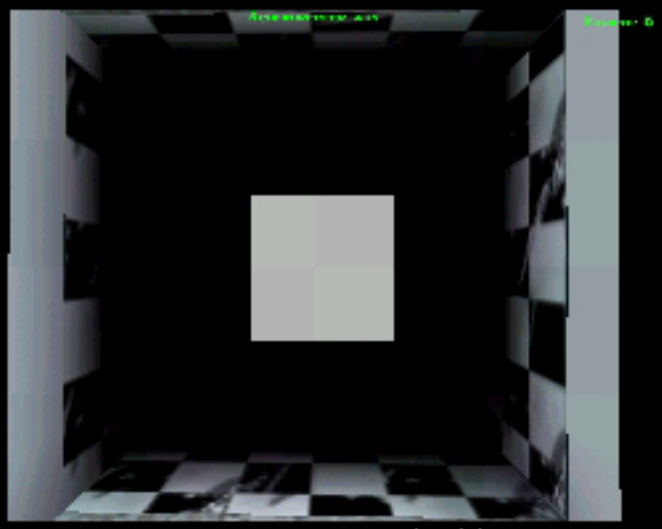
FunkyEffekt5
Light: no Light - normal, stillstehend, mit maeßig stark nach aussen dringender Strahlung (wie soll man das sonst beschreiben ...)



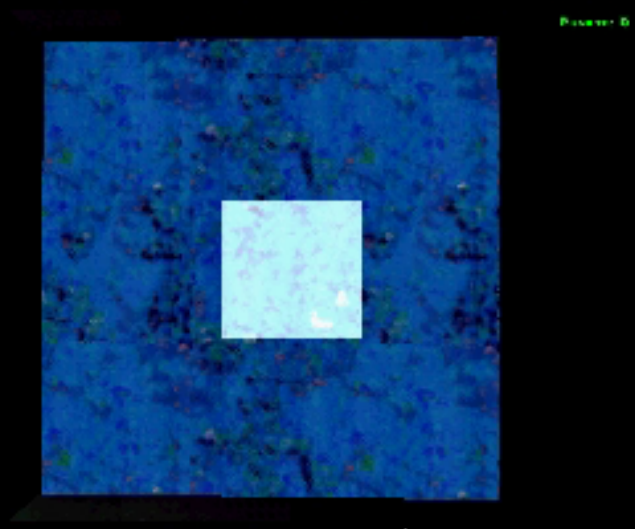
FunkyEffekt9
Light: no Light - normal, die Straenge bewegen sich ganz langsam wellenfoermig von unten nach oben



GreenMysterySurface
Light: 0.0 / 10.0 / 5.0 - Schild, prismenartig wabernde Oberfläche

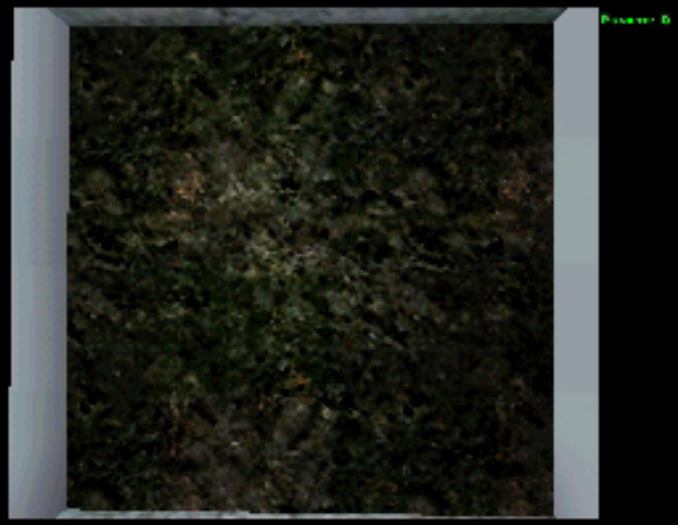


Korea_Matcen_Shield
Light: no Light - normal, absolut durchsichtig , ohne sonstige Besonderheiten

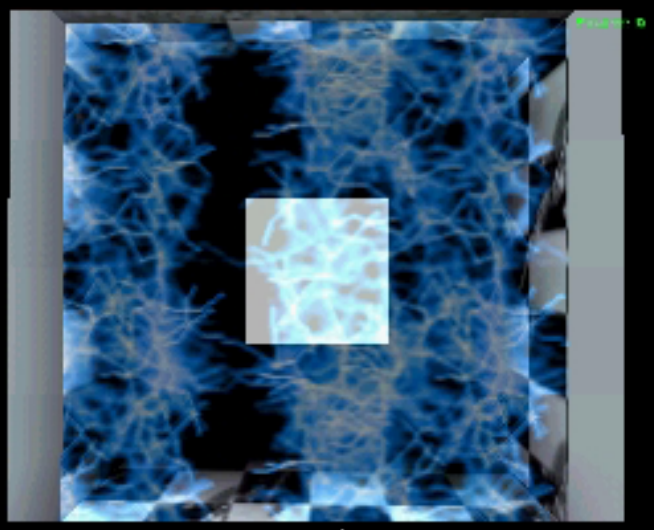


Large Water Body
Light: 0.0 / 2.2 / 2.4 - normal , leichte wellen, gekraeuselte Oberfläche, dampft bei Waffenbeschuss

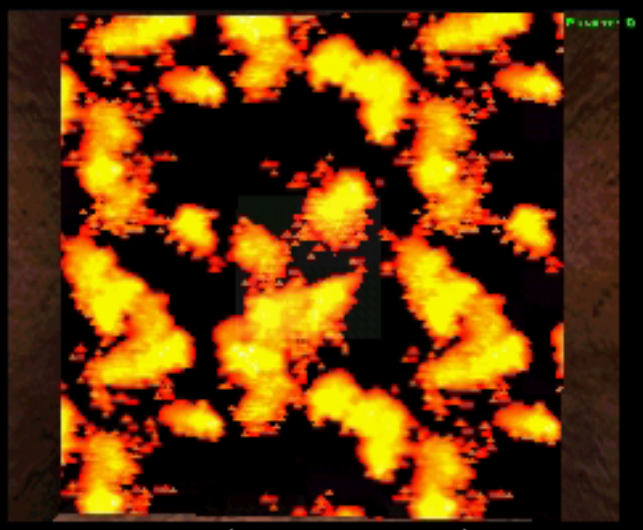
MarsForcefield
Light: 1.0 / 0.5 / 0.5 - Schild, starke durcheinanderlaufende Wellen und Kreise



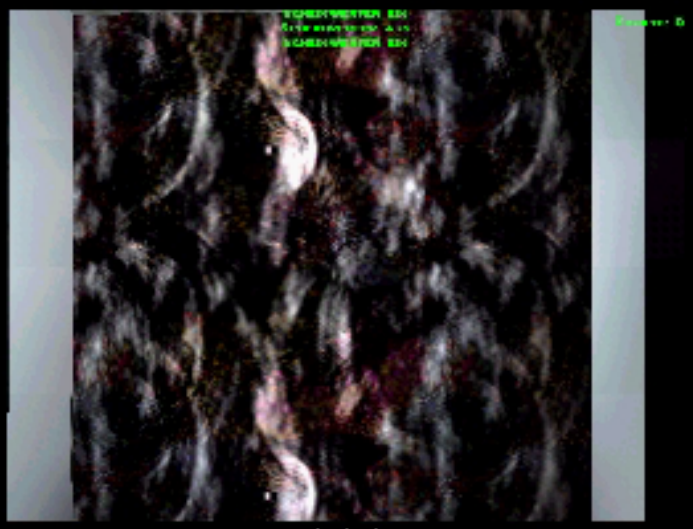
MarsVolcanoRockShimmer
Light: no Light - normal , mit ausgedehnten darueberhinweglaufenden (hitze) wabernden Flaechen



Matcen Lightning
Light: no Light - normal, Blitzstraenge von unten nach oben durchlaufend

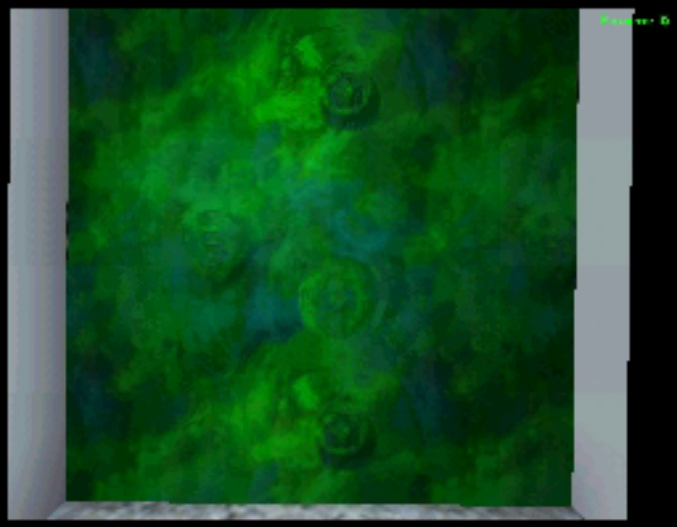


Nano Plasmic Cesspool
Light: 18.0 / 10.0 / 8.0 - verursacht Hitze schaden, sieht aus wie aufflackernde gluehende Kohlen, dampft bei Waffenbeschuss

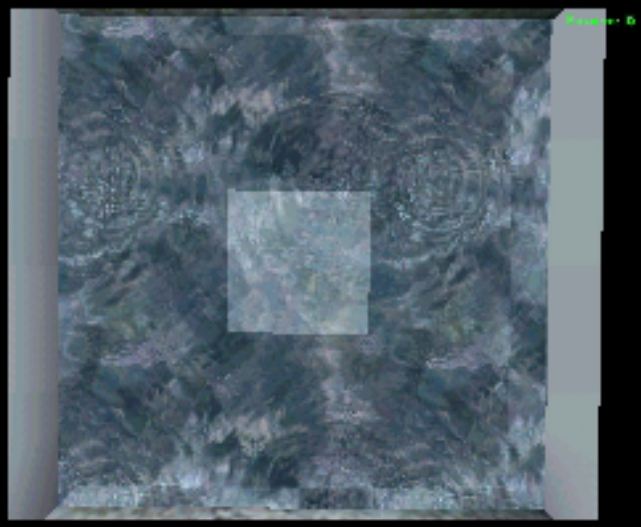


OilSlick1
Light: no Light - normal , als ob viele grosse Tropfen hineinfallen

OrangePlasma
Light: 2.0 / 1.5 / 1.0 - normal, leicht bewegte Oberflaeche, dampft bei Waffenbeschuss

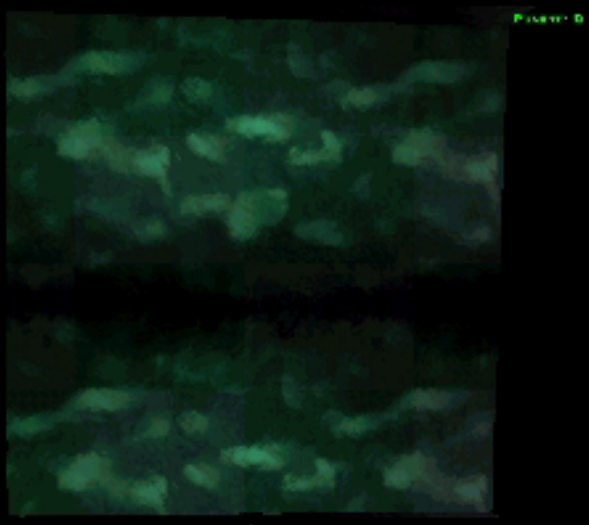


PhobosAcid
Light: no Light - verursacht Saeureschaden, konzentrisch auseinanderlaufende Kreise, dampft bei Waffenbeschuss

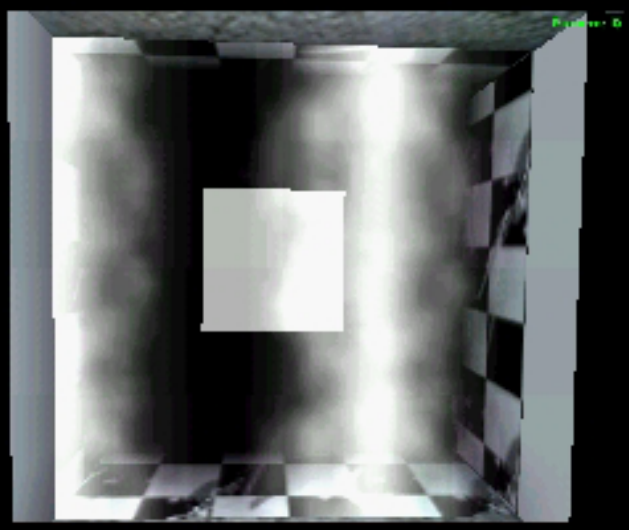


PondRipple1
Light: no Light - normal, auseinanderlaufende konzentrische Kreise (Wasser eben), dampft bei Waffenbeschuss

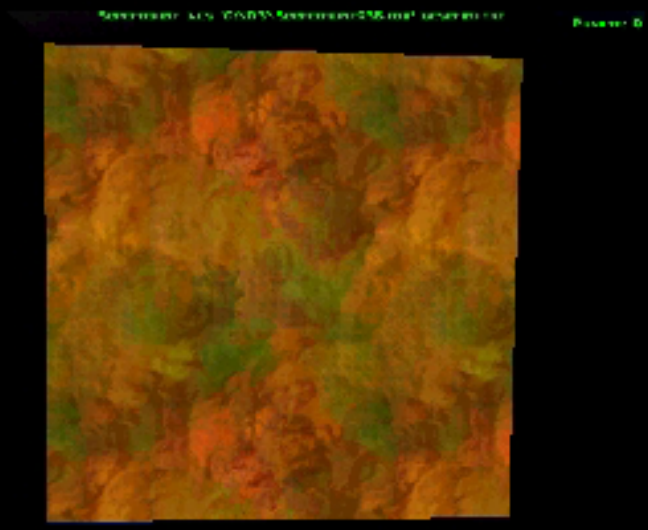
Rippley1
Light: 0.5 / 0.4 / 0.2 - normal, schnell von unten nach oben durchlaufend



Rippley2
Light: 0.2 / 0.5 / 0.3 - normal, sehr schnell von unten nach oben durchlaufend



rsa
Light: no Light - normal, schmaler und dann wieder breiter werdende dampfartige Straenge



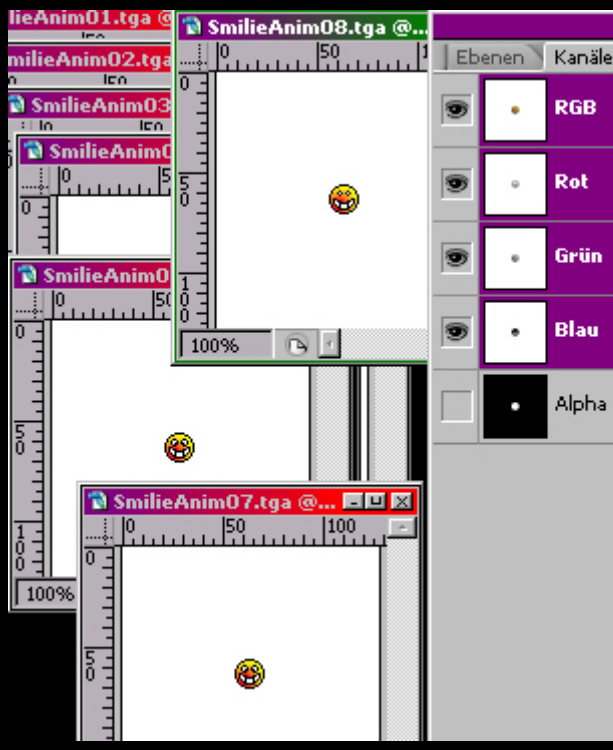
SewerWater
Light: 2.2 / 2.1 / 1.0 - normal, leicht bewegte sich kraeuselnde Oberflaeche,dampft bei Waffenbeschuss

(Auch von mir)

066 - Animierte Texturen - Selbstgemacht

Ragil Ral

Der Prozeß unterscheidet sich vom Erstellen/Einbinden selbstgemachter statischer Texturen nur wenig. Man erstellt einfach nur nicht eine, sondern mehrere `.tga`'s, die dann entsprechend in `.ogf`'s umgewandelt werden. Ab da läufts dann anders weiter.
Wie bei einem animierten `.gif` liegen in einer `.oaf` (Outrage Animated File) die Teilbilder der Animation in Reihe vor und sind in einer Datei zusammengefaßt.



Erster Schritt

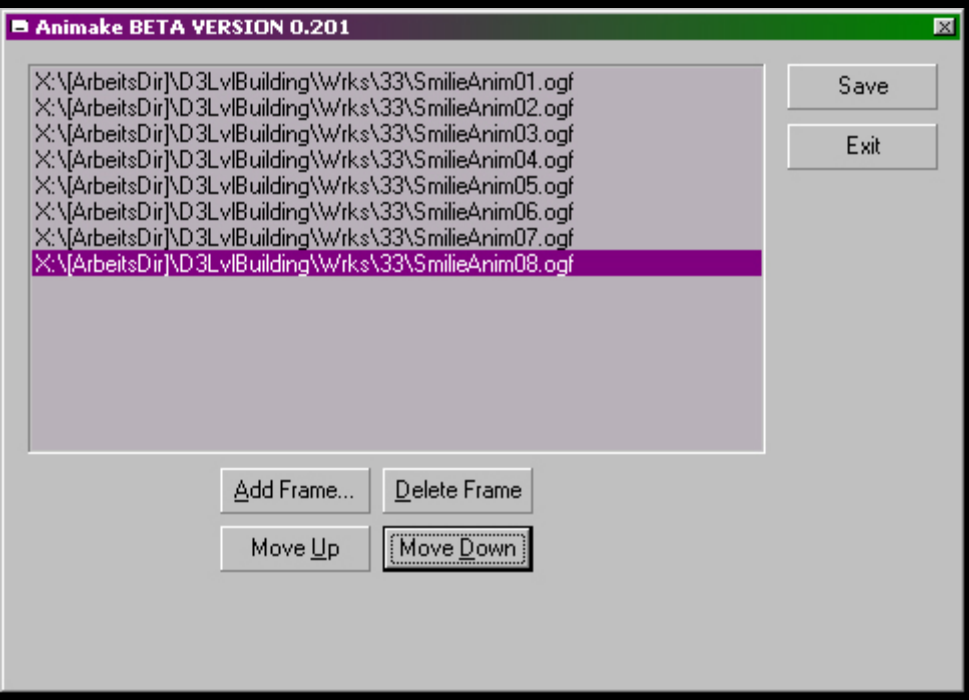
Erzeugt eure Einzelbilder. Sie können auch transparent, teiltransparent sowie andere Flags haben. Ich hab hier einen animierten Smilie aus dem Descentforum genommen, die Einzelbilder extrahiert, und für die jeweils eine Alpha-Maske angelegt. Es sind acht Bilder.

Falls bestimmte Frames gleich sind, kannst Du natürlich diese mehrmals verwenden, wir sind ja von Natur aus faul

Zweiter Schritt

ist kurz, mit OGFTool oder D3 Image Tool die `.tga`-Files nach `.ogf` wandeln.

Dritter Schritt



Nun brauchst Du das Tool ‚Animake', derzeit in Version BETA 0.201. Starte es, klicke **Add Frame...** und wähle Deine Einzelbilder aus (links). Du kannst mit **Strg** und klicken mehrere aufeinmal auswählen. Animake zeigt Dir dann eine Liste der Dateien. Ordne sie noch, indem Du einen Eintrag markierst und mit **Move Up** oder **Move Down** in der Liste bewegst.

Dann einfach auf **Save**, Du wirst dann gepromptet wo Du hinspeichern willst, und die Dateieendung wird mit `.ogf` gleich vorgegeben.

Vierter Schritt

ist dann nur mehr noch das Einbinden in eine `.gam` bzw. `.mn3`, und, natürlich, das Verwenden in einem Level
Da ein Screenshot hier nix bringt, gibts die `.oaf` in der gepackten Datei.

067 - Texturen @ 256x256

Ragil Ral

Es gibt die Option, Texturen mit 256 Pixel Kantenlänge zu verwenden, doch muß D3 hier pro Textur viermal soviele Daten laden als bei den Standard-128x128er-Texturen. Daher sollte man sich genau überlegen, wo man diese Texturen verwendet.

Die primären Einsatzgebiete sind Texturen fürs Terrain, teiltransparente und wenn man ein Bild im Level haben möchte, daß höher aufgelöst sein soll; etwa eine Karte oder ein Bildschirm mit Inhalten, die für den Spieler beim Lösen einer Mission nötig oder hilfreich sind.

Terrain-Texturen

Alle Texturen der Außenwelt werden extrem auseinandergezogen, daher empfiehlt es sich für Satelliten, den Himmel und das Terrain selbst 256x256er Texturen zu verwenden. Besonders beim Himmel, da dieser aus einer einzelnen Textur besteht welche über den Skydome gespannt und nicht gekachelt wird.

Teiltransparente Texturen

Wie in der Sektion zur Teiltransparenz angekündigt, habe ich den Maschendrahtzaun nochmal gemacht.

Wenn Ihr das Bild betrachtet, dürfte eigentlich alles klar sein...

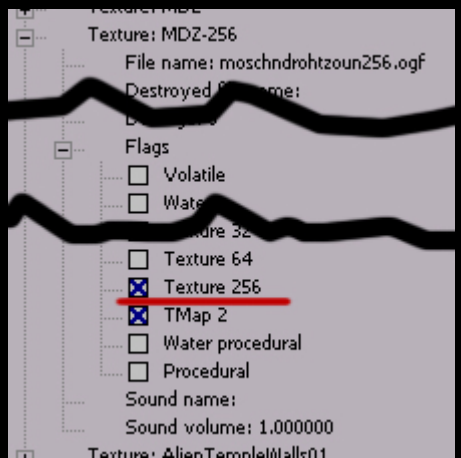
Erstellen

Da geht Ihr ganz gleich wie in den vorigen Sektionen beschrieben, nur eben mit 256 Pixel im Quatrat.

Der GAM-Eintrag

Die Flag für 256 setzen:

Das war's auch schon!



Abschnitt G – Sounds

Geräusche, Klänge und Musik beleben das Spiel, besonders in Singleplayer. Hier lernt Ihr, sie zu verwenden. Sounds sind auch für selbst erstellte Türen und andere Levelobjekte wichtig.

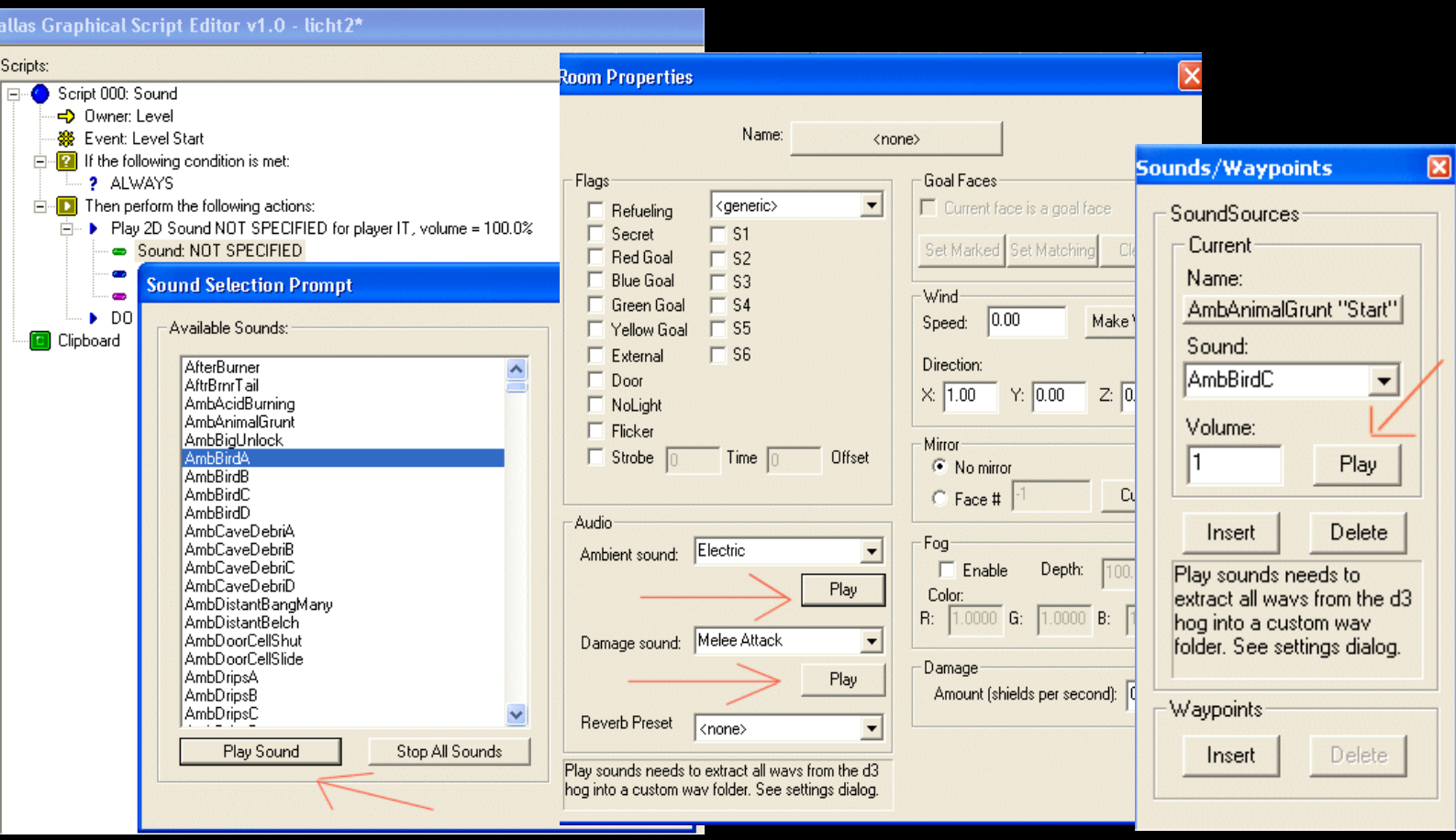
068	**Sounds abspielen**	Was man machen muß damit D3Edit die Sounds abspielen kann.	Atan	257
069	**Sound im eigenen Descent3-Level**	Einfügen von Sounddateien in D3-Levels	Toxxeon	260
070	**Sound: Probleme vermeiden**	Eine Erklärung, wie man Probleme mit dem Sound umgehen kann	Atan	264

Toc

068 - Sounds abspielen

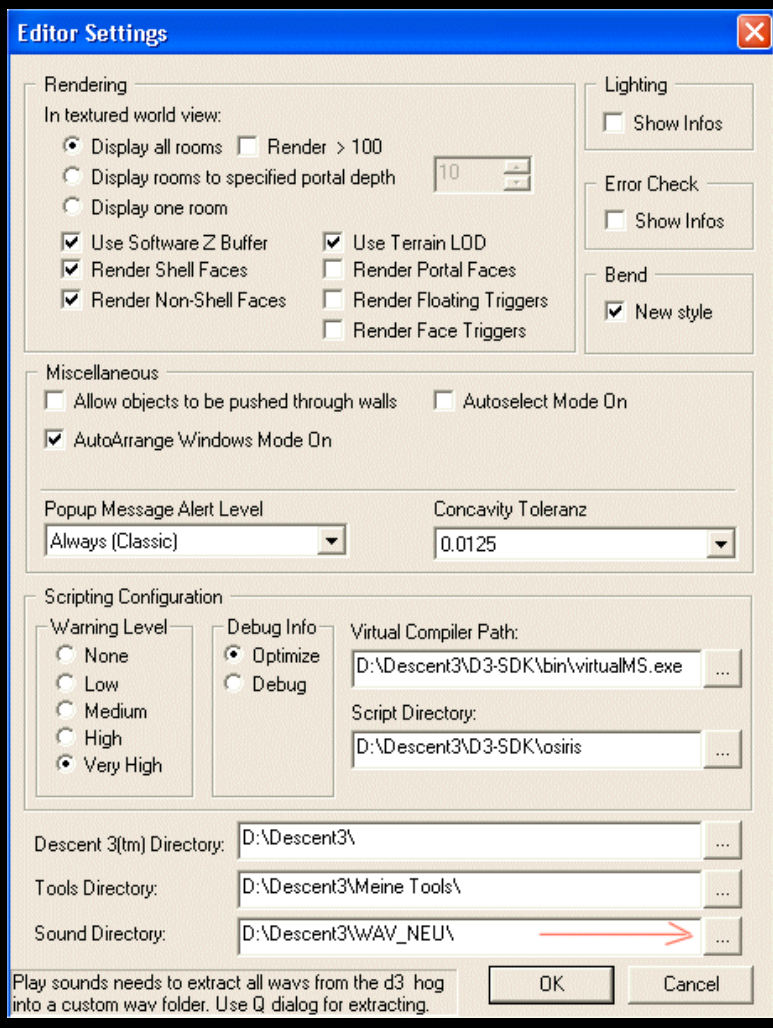
Atan

(V1.0)

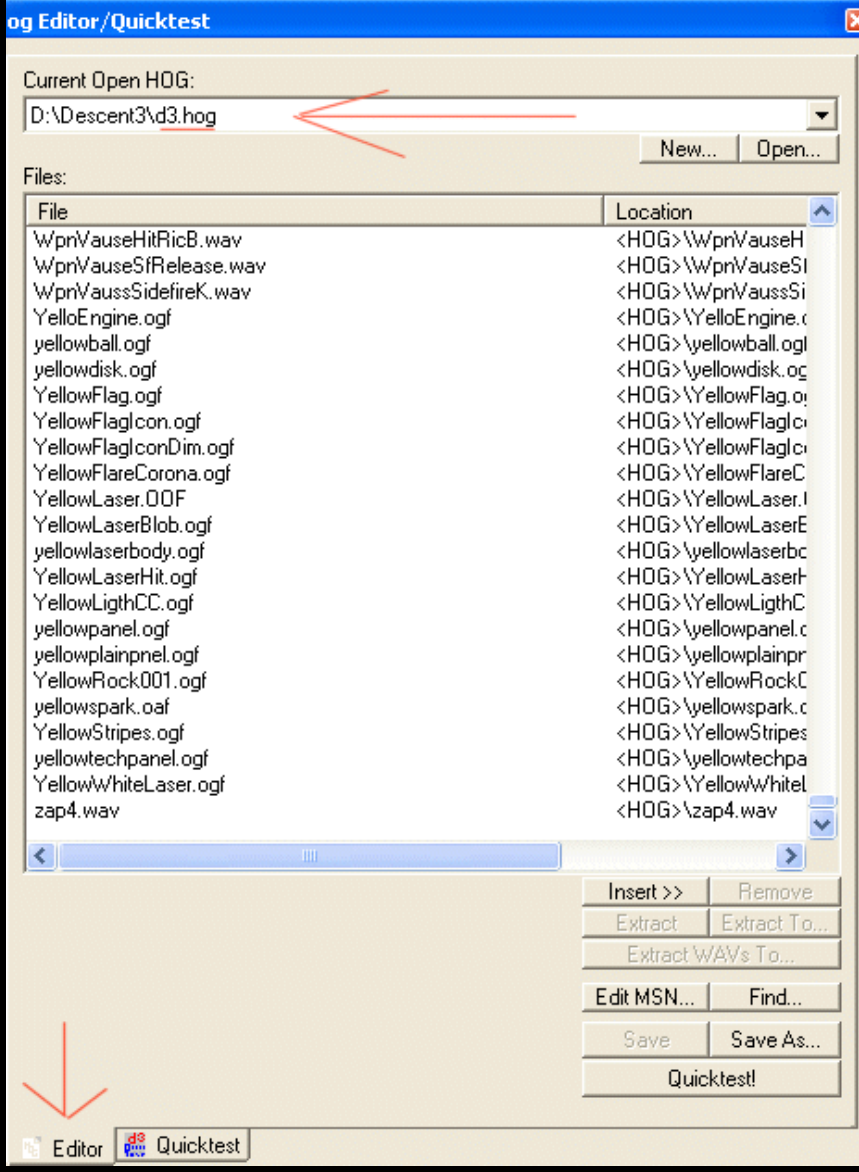


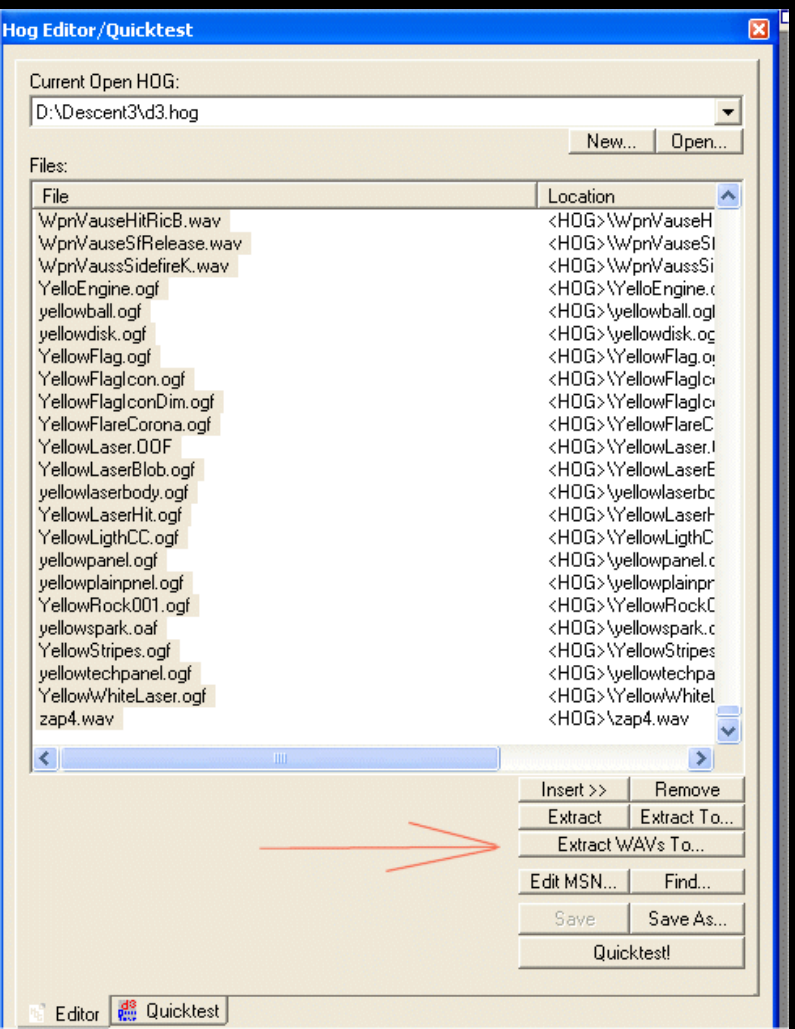
In den oben abgebildeten Dialogen können Audiodateien in Levels eingebaut werden, und damit auch abgespielt werden (rote Pfeile).

Bevor wir aber die Sounds abspielen können müssen wir erstmal ein paar Vorarbeiten dafür leisten.
Zuerst erstellen wir einen neuen Ordner der die `.wav`'s aufnehmen soll.
Dafür benötigt man ca 70 MB !!! freien Platz auf der Festplatte.
Im Editor Settings Dialog tragen wir diesen Ordner nun ein (rechts):



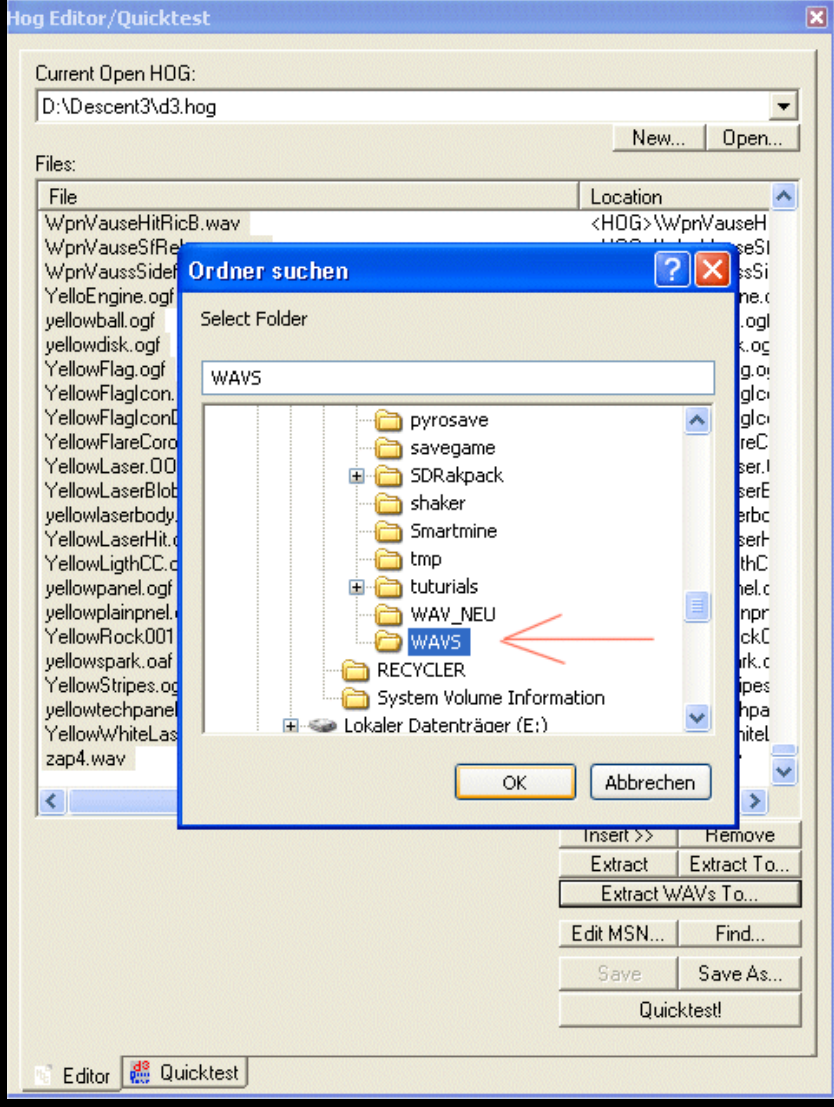
Zum Extrahieren der WAV's benutzen wir den Quicktest. Also starten wir den Editor und öffnen im Quicktest die Tabkarte Editor, und dort die D3 hog Datei:



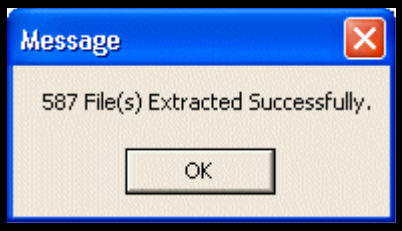


Danach markieren (Linke Maustaste) wir den untersten Eintrag, ziehen den rechten Scroll-Balken ganz nach oben und markieren bei gedrückter **Shift** - Taste den obersten Eintrag. Nun braucht es einiges an Geduld, je nach Rechnerleistung dauert es sehr lange bis alle Einträge automatisch markiert werden. Alternativ kannst Du auch nach Filetype sortieren, die .wav's stehen dann als letzte in der Liste und nur die markieren. Nachdem alle Einträge markiert wurden betätigen wir **Extract WAV's To**.

Es öffnet sich ein Auswahlfenster und wir wählen unseren gewünschten WAV Ordner an:



D3 Edit extrahiert nun alle WAV's aus der D3 HOG und kopiert sie in unseren WAV Ordner.
Nach einiger Zeit ist der Job erledigt und sollte so einen Nachrichtendialog bringen:



Diese Aktion brauchen wir nur einmal durchführen, danach stehen uns die Sounds in DALLAS und dem Room Properties-Dialog immer zur Verfügung.

069 - Sound im eigenen Descent3-Level

Toxxeon

In dieser Anleitung wird erläutert, wie man Sound-Dateien in eigene D3-Level einfügt.
Um eigenen Sound in das Level zu integrieren, brauchen wir das Programm *MusicTester*. Wenn ihr den originalen D3-Editor (D3edit v1.1 und jüngere) installiert habt, dann müsste sich dieses Programm im *Editor*-Ordner befinden bzw. über das Tool-Menü aufrufbar sein.

Außerdem ist der *Audio Taunt Manager* von Alex "LEX" Leuthold sehr zu empfehlen. Damit ist das Konvertieren beliebiger .wav-Dateien ein Kinderspiel... Dieses geniale Programm könnt ihr euch entweder auf Fischleins Tool-Seite (dort auch das "Original-Editor"-Paket inklusive *MusicTester*...) oder direkt bei LEX downloaden.

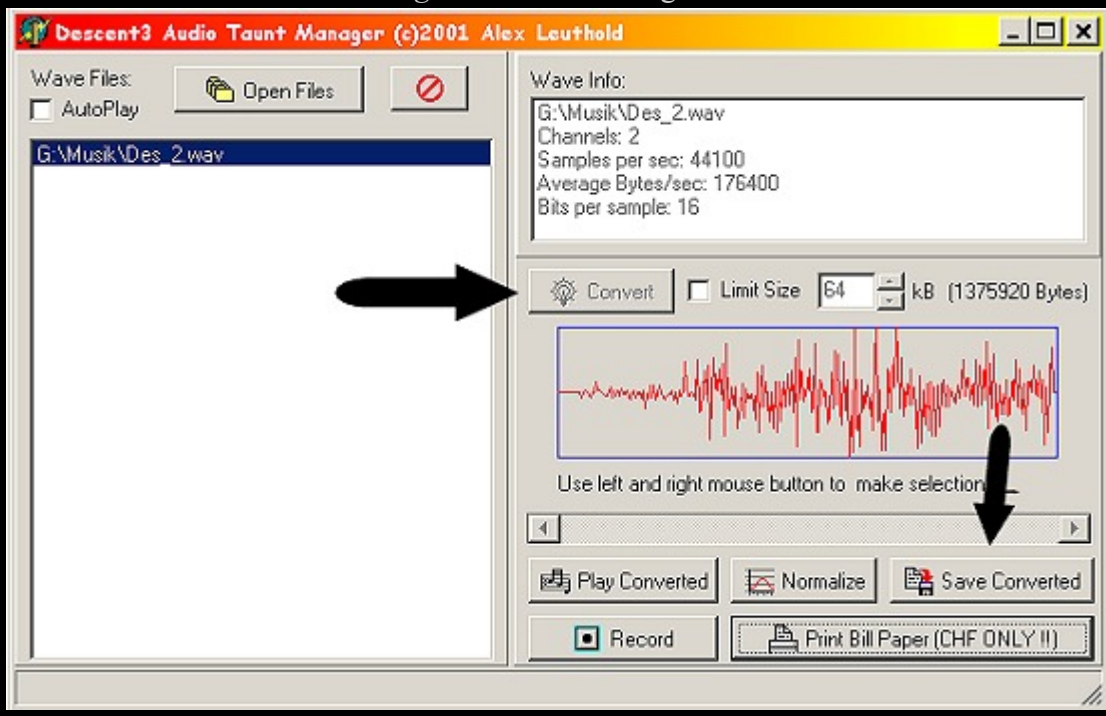
Kurzer Überblick:

Um eigenen Sound in ein D3-Level einzufügen, muss man die originale `.wav`-Datei in eine `.osf`-Datei (Outrage Sound Format) konvertieren. Zusätzlich muss noch eine `.omf`-Datei (Outrage Music File) erstellt werden. Mithilfe dieser Datei wird der Sound im Level gesteuert.

Als erstes braucht ihr logscherweise eine entsprechende Musiksequenz. Am besten eine Endlosschleife, damit die Spieler keinen störenden "Bruch" oder sowas in der Art hören müssen. Denn sowas wird im Laufe der Zeit echt nervig! Denkt außerdem daran, dass durchschnittlich 22,5 KByte pro Sekunde Sound anfallen. Also bei einem verwendeten Musikstück mit 1 Minute Länge sind das ca. 1,3 MByte an Daten. (im unge"zip"ten Level)

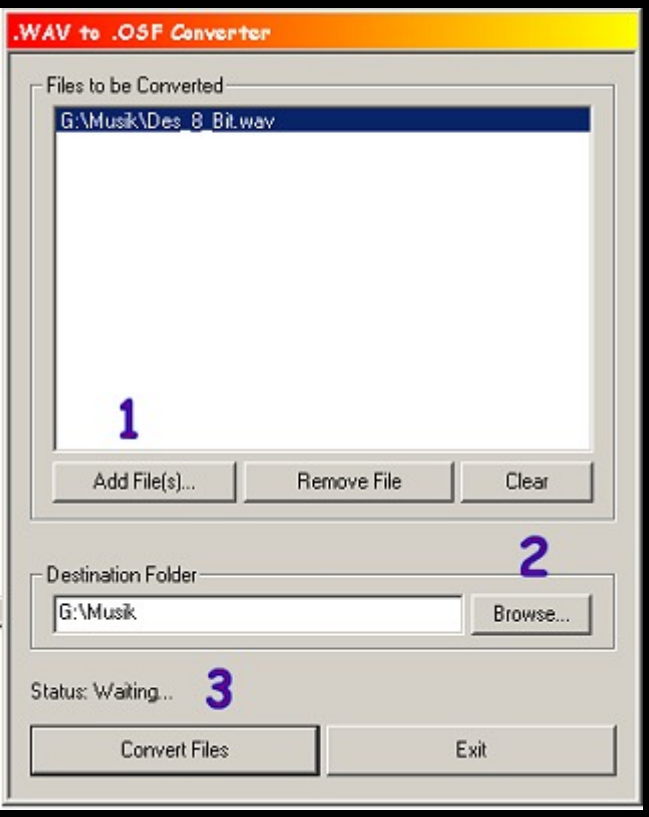
Wichtig!
Beachtet unbedingt eventuelle Copyrights der von euch verwendeten Musikstücke o.ä.!

Vorzugsweise solltet ihr bei den Urhebern nachfragen, ob ihr dessen Musik in euren Level verwenden dürft. Oder ihr erstellt eure eigene mit entsprechenden Musikprogrammen bzw. per Instrument...
Wenn dies alles geklärt ist, können wir jetzt mit dem eigentlichen Erstellen des Level-Sounds beginnen. Die von euch ausgewählte Sound-Datei muss erstmal "zwischenkonvertiert" werden. Und zwar in das 8-Bit-Mono-.wav-Format.
Am einfachsten geschieht das mit dem *Audio Taunt Manager*. Startet das Programm und ladet mittels **Open Files** eure Sounddatei hinein. Klickt nun auf **Convert**, um die Sound-Datei zu konvertieren. Ihr könnt und braucht nichts weiter einzustellen, es wird gleich das richtige Format gewählt. Übrigens, die Einstellung **Limit Size** ist wichtig, wenn ihr euren speziellen Audio-Hohn kreieren wollt ☺ Diese Sound-Dateien dürfen nämlich nicht größer als 64 KByte sein!



Play Converted solltet ihr unbedingt nutzen, weil manchmal der konvertierte Sound ganz anders als das "Original" klingt, da es eben das recht sparsame 8-Bit-Mono-Format sein muss. Die Option **Record** ist dazu da, um per Mikrofon was aufzunehmen, oder ihr spielt eine CD oder einen MP3-File ab...und "recordet" den Sound gleich mit dem *Taunt Manager* im richtigen Format. **Aber Achtung: Wie schon angesprochen, beachtet unbedingt die Urheberrechte Anderer!**
Ratsam ist es, den konvertierten Sound zu **Normalize**'n... Um das ganze letztendlich abzuspeichern...**Save Converted** nutzen.

Jetzt müssen wir den Sound weiterbearbeiten...



Dieses geschieht im **Music Tester**, einem Programm welches beim originalen D3-Editor "mit inklusive" ist.

Startet nun den **Music Tester** und konvertiert euren 8-Bit-Sound in das `.osf`-Format. Geht folgendermaßen vor: Klickt auf Actions und dann auf Convert Files... Es öffnet sich nun der **.WAV to .OSF Converter** (links).

1.) Die entsprechende(n) Datei(en) anwählen
2.) Einen Speicherort für die OSF-Datei raussuchen
3.) Sound konvertieren.

Wenn keine Fehlermeldung kommt, ist alles OK. Das Konvertieren kann übrigens eine Weile dauern, je nachdem, wie groß die Datei ist... Schließt mit **Exit** den Konverter.

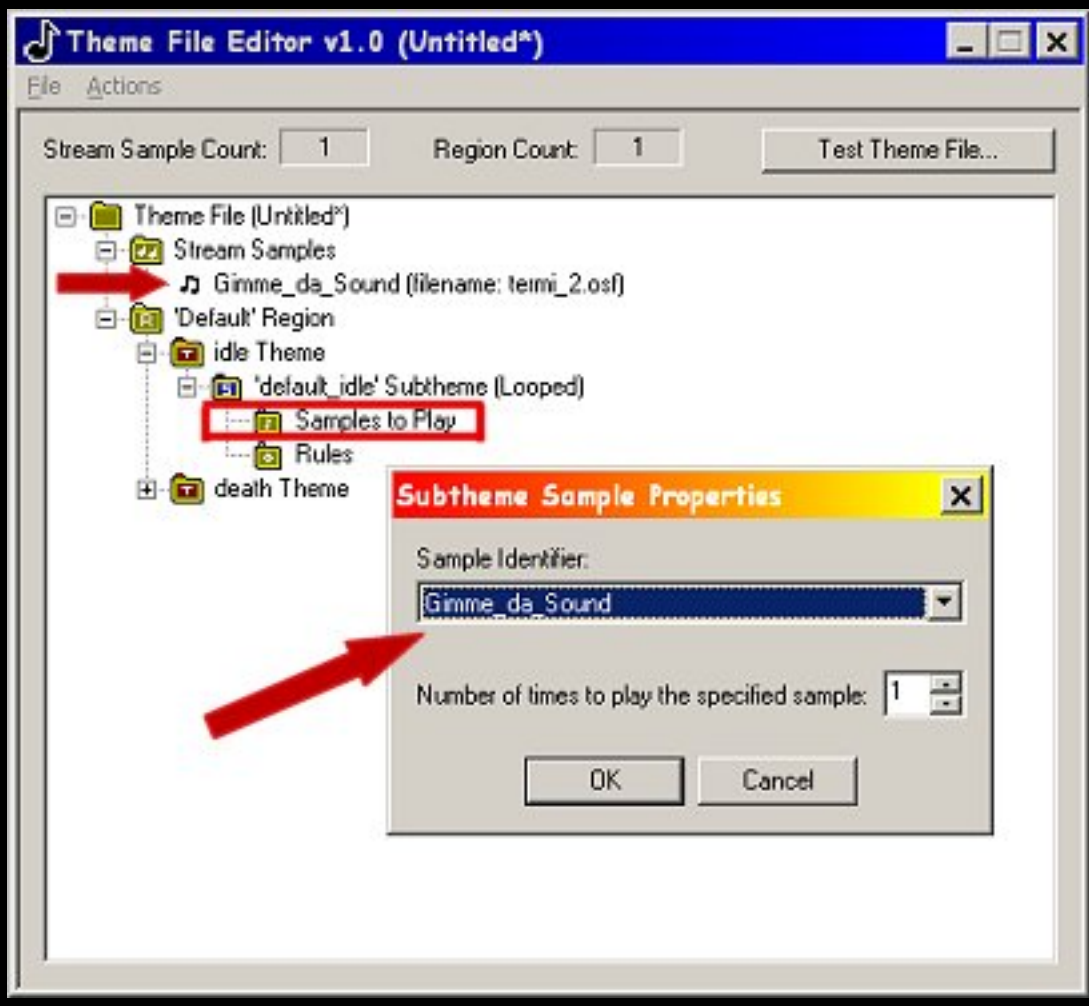
Ich empfehle euch, nun gleich eine wichtige Einstellung vorzunehmen: Klickt auf **Test Theme File...**, im neu erscheinenden Fenster auf **Options** und dann auf **Edit File Search Path...** Jetzt auf **Add Path**, und sucht den Speicherort eurer `.osf`-Dateien raus. Wenn diese Pfadzuordnung gemacht ist, könnt ihr später euren Sound hier im Music Tester anhören, und die OMF-Datei wird vom Programm richtig zusammengestellt.

Nun erstellen wir die **OMF**-Datei.

Erst mal zum Verständnis: Diese OMF-Datei ist eine Art Script, welches steuert, welche OSF-Datei wie oft und zu welchem Ereignis abgespielt werden soll. Klingt vielleicht kompliziert, ist es aber im Prinzip gar nicht ☺

Macht einfach folgendes...
Klickt mit der *rechten* Maustaste auf **Stream Samples** und danach normal auf **Add New Stream Sample...** Es öffnet sich ein neues Dialogfenster. In diesem vergibst du im oberen Teil einen eindeutigen Namen und suchst dann mit **Browse** deine entsprechende OSF-Datei raus.
Als nächstes klickst du auf das Pluszeichen bei **Default Region**, dann nochmal auf das bei **idle Theme** und zu guter letzt auch nochmal bei **'default_idle' Subtheme (Looped)**. Und nun wieder mit der *rechten* Maustaste auf **Samples to Play** klicken.

Im erscheinenden Dialogfenster müsste im Pull-Down-Menü dein von dir eben gerade vergebener Name erscheinen. Siehe Bild unten:



Natürlich können auch mehrere OSF-Dateien gleichzeitig in die OMF-Datei eingebunden werden. Wie schon weiter oben beschrieben bei **Stream Samples**. Zum Beispiel kann man eine *Intro*-, eine *Hintergrundmusik* und ein *death Theme* gleichzeitig vereinbaren. Im Level ist nur "Platz" für **eine** OMF-Datei...

Ein kleiner Trick:
Wenn ihr wollt, dass beim Eintritt in's Level ein *Begrüßungs*-Sound abgespielt wird, und anschließend "ganz normal" die Hintergrundmusik, dann so vorgehen:
Erst mal zwei OSF-Dateien unter **Stream Samples** einbinden, dann mit der rechten Maustaste auf **idle Theme**, und nun auf **Add new Subtheme...**. In die Dialogbox einen markanten Namen eintragen (z.B. Hintergrund), und auf **O.K.** klicken

Bei **'default_idle' Subtheme** die Intromusik-Datei angeben, bei **'Hintergrund' Subtheme** die entsprechende Hintergrundmusik. Jetzt mit der rechten Maustaste auf **'default_idle' Subtheme (Looped)**, dann auf **Edit...**. Das Häckchen bei **Enable continous playback of this subtheme** entfernen.

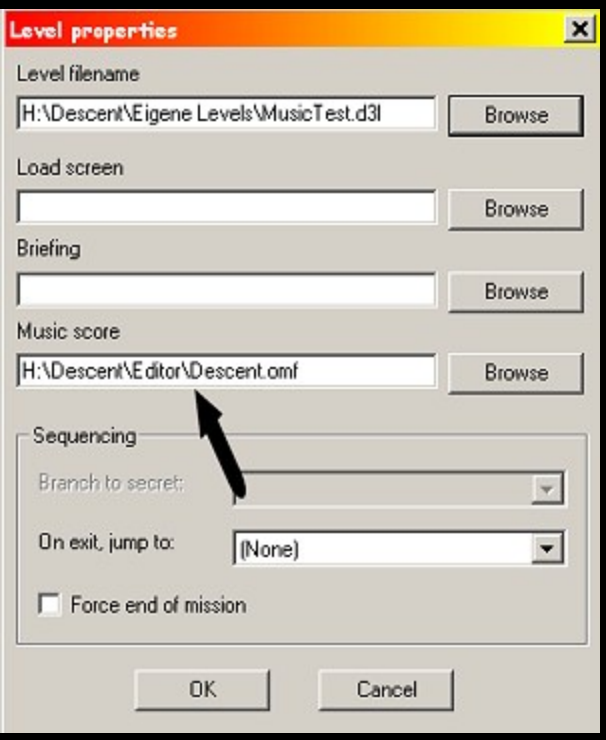
Insgesamt sollte es dann so aussehen:

Eigenartigerweise wird beim allersten Eintritt des Spielers der Intro-Sound zweimal abgespielt, beim nächsten Eintritt (nach Abschuss) nur einmal... Aber probiert's mal.

Jetzt abspeichern! Hier gibt es von mir noch einen Hinweis:
Wenn man unter File --> Save as die OMF-Datei abspeichert, habe ich die Erfahrung gemacht, dass man auch die Dateiendung mit hinschreiben muss. Also nicht nur wie üblich z.B. `Levelmusik`, sondern halt `Levelmusik.omf`. Dann müßte es das Programm richtig abspeichern.

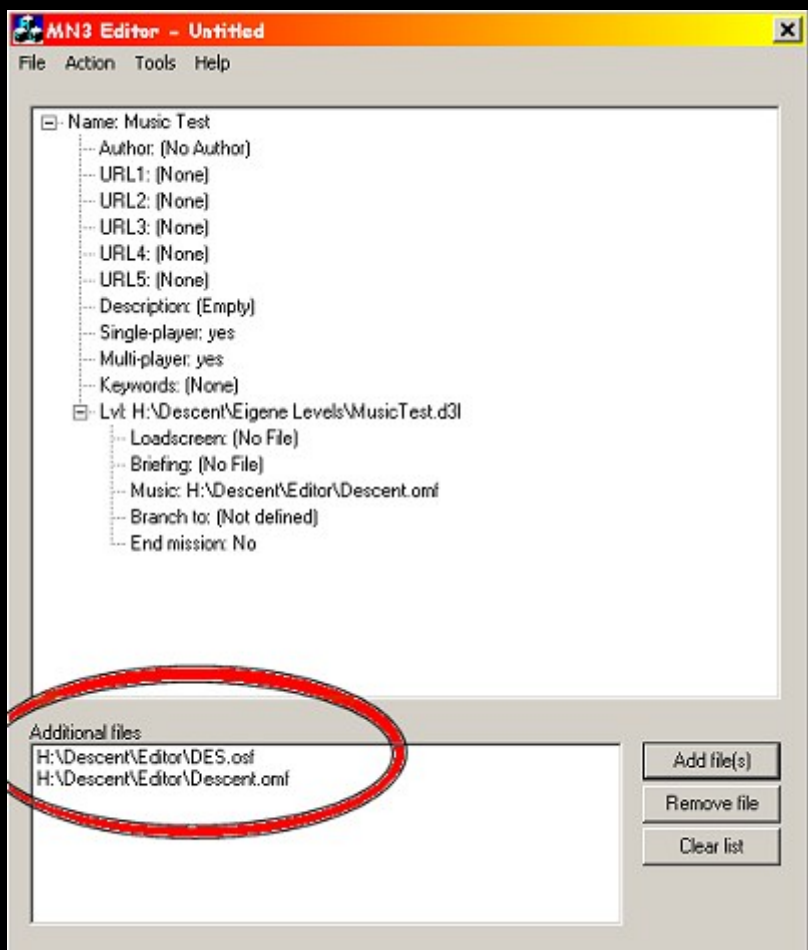
Nun könnt ihr eure Musik mal anhören.
Klickt dazu auf den Button **Test Theme File...**

Eine interessante Sache ist **death Theme** . Hier kann man nämlich einen alternativen Sound vereinbaren, der abgespielt wird, wenn ein Player im Level abgeschossen wird.
Der Unterpunkt **Rules** erscheint mir im *Music Tester* leider etwas halbfertig, dort hat man unter Umständen die Möglichkeit, das Abspielen des Soundes nach bestimmten Kriterien noch zu verfeinern. Das müsstest ihr aber bitte selbst noch austesten...



Zuletzt den Sound in das Level einfügen. Das geschieht im *MN3-Packer* (mn3edit.exe), sprich beim Zusammenschnüren des fertigen Levels.

Bei den Level Properties (links) unter Music score die erstellte `.omf`-Datei angeben.



Nun noch alle entsprechenden `.osf`- und `.omf`-**Dateien** unter *Additional Files* angeben. Dann ist schon alles geschafft. ☺

Viel Spaß beim Vertonen eurer Level!

070 - Sound: Probleme vermeiden

Atan

Problem:

Ein Schuss auf eine gegenüberliegende Wand erzeugt beim Auftreffen keinen Sound. Die Ursache liegt im Leveldesign und lässt sich beim Beachten einiger Punkte beheben, bzw. schon beim Erstellen des Levels vermeiden.

Ein paar Grundlagen:

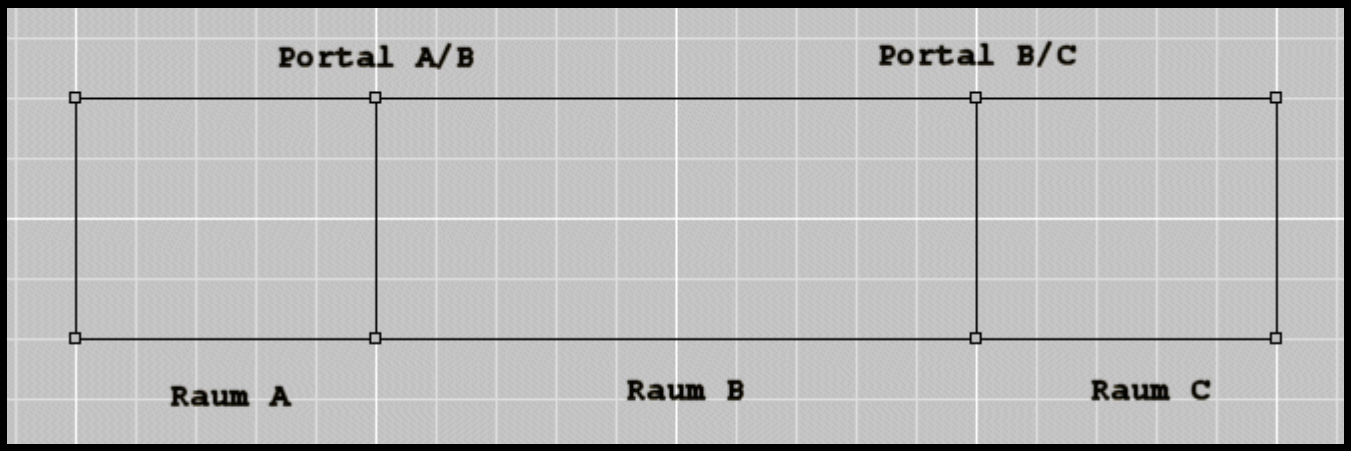
Während der Erzeugung der BOA Vis berechnet D3Edit auch die Soundentfernungen innerhalb des Levels und trägt diese in eine Tabelle ein. Diese Tabelle wird mit dem Level abgespeichert und von Descent3 für die Sounderzeugung benutzt. Wird die maximale Entfernung von 400 Units überschritten, dann wird diese Entfernung in der Soundtabelle für ungültig erklärt. Descent3 erzeugt bei einer ungültigen Entfernung keinen Treffer-Sound.

Wie wird diese Sound-Entfernungstabelle erstellt?

Es wird für jeden Raum geprüft bis in welchen Raum gesehen werden kann. Dabei werden die Entfernungen zwischen dem ermittelten Start- und Endraum addiert und (sofern gültig) in die Sound-Tabelle eingetragen.

Wie funktioniert das von Raum zu Raum ‚sehen'?

Unser Beispiel-Level soll aus 3 in Reihe liegenden Räumen bestehen (A, B, C) die durch Portale verbunden sind. (Raum A -> B; Raum C -> B; Raum B -> A und B -> C) Daher haben Raum A: 1 Portal, Raum B: 2 Portale und Raum C: 1 Portal.



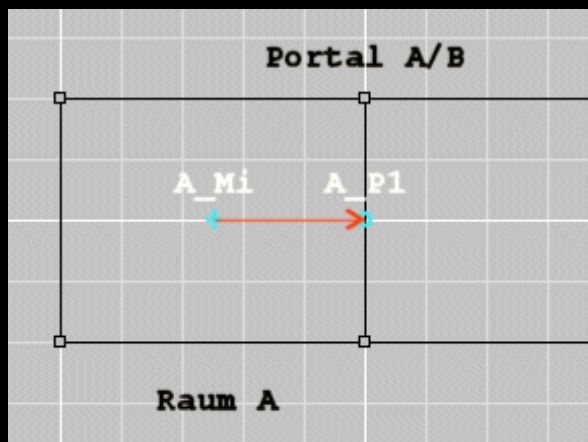
D3Edit ermittelt nun zuerst für jeden Raum die maximale Sichtweite und trägt für jeden dieser Räume (er ist der Start-Raum) den dazugehörigen End-Raum
in eine Tabelle ein. Das bedeutet für den Beispiel-Level:

Start-Raum	End-Raum
A	C
B	A
B	C
C	A

Wie wird die Raum-zu-Raum-Entfernung ermittelt?

Lassen wir D3Edit mit Raum A als Start-Raum beginnen. Nach der Tabelle wird nun die Entfernung vom Center des Raumes A zum Center des Raumes C ermittelt. Dazu wird zuerst die Entfernung (`Dist`) zwischen dem Raum-A-Mittelpunkt und dem ersten Portal dieses Raumes ermittelt.

Die Entfernung AC wird auf 0 gesetzt. Dann wird eine Gerade vom Raum-Center zum Center des Portal-Faces A/B gezogen.



Erreicht diese Linie das Portal-Face nicht, ist diese Entfernung ungültig:

```
Dist AC = ungültig.
```

Erreicht die Linie das Portal-Face wird diese Entfernung zwischengespeichert:

```
Dist AC = Dist AC + (Dist A_Mi -> A_P1)
```

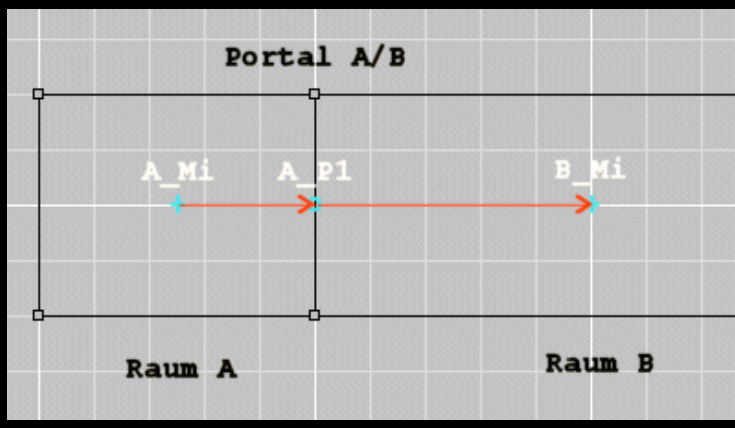
Jetzt wird vom Portal-Center aus eine Gerade zum Raum-Mittelpunkt des Raumes B gezogen.

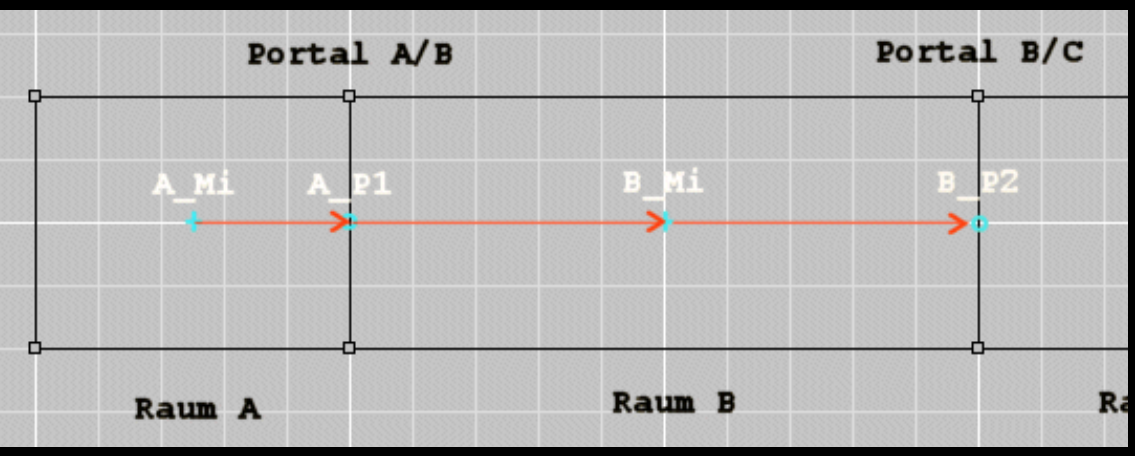
Erreicht diese Linie den Raum-B -Mittelpunkt nicht, ist die Entfernung ungültig.

```
Dist AC = ungültig.
```

Bei Erfolg wird die Entfernung zu der bereits gespeicherten addiert:

```
Dist AC = Dist AC + (Dist A_P1 -> B_Mi)
```





Dann wird eine Gerade vom Raum-B-Center zum Center des Portal-Face B/C gezogen.

Erreicht diese Linie das Portal-Face B/C nicht, ist die Entfernung ungültig;

```
Dist AC = ungültig
```

Bei Erfolg wird die Entfernung zu der bereits gespeicherten addiert:

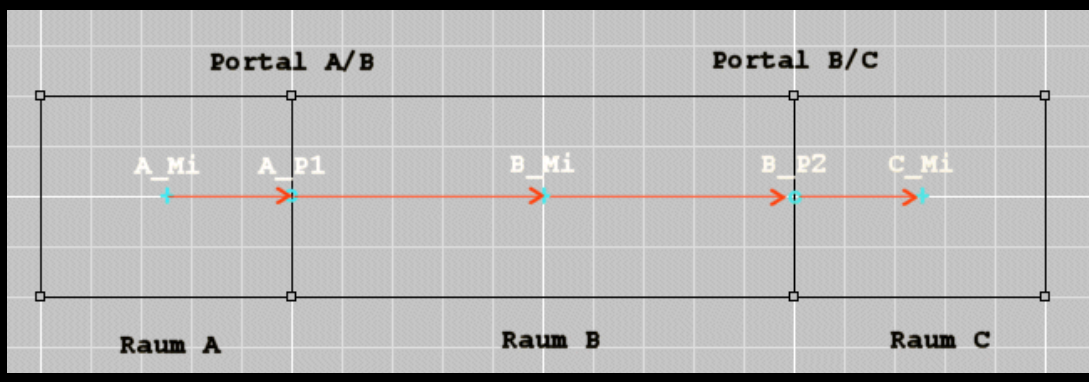
```
Dist AC = Dist AC + (Dist B_Mi -> B_P2)
```

Jetzt wird vom Portal-Center B/C aus eine Gerade zum Raum-Mittelpunkt des Raumes C gezogen.

Erreicht diese Linie den Raum C -Mittelpunkt nicht, ist die Entfernung ungültig.

```
Dist AC = ungültig
```

Bei Erfolg wird die Entfernung zu der bereits gespeicherten addiert:




```
Dist AC = Dist AC + (Dist B_P2 -> C_Mi)
```

Nun überprüft D3Edit ob die Entfernung AC > 400 ist.

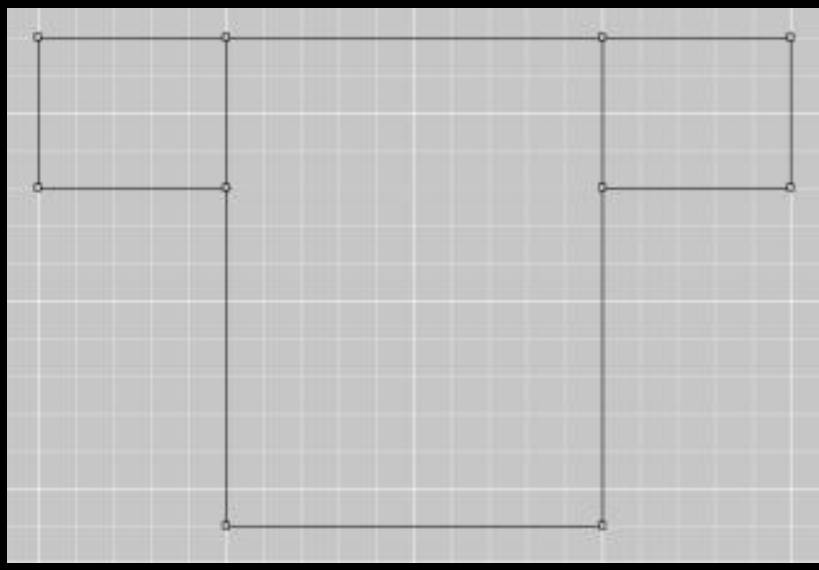
Ist das der Fall, wird Entfernung AC für ungültig erklärt. Andernfalls wird die ermittelte Entfernung nun in die Max Sound Entfernung Tabelle unter A/C eingetragen. Raum A ist fertig berechnet da er nur ein Portal besitzt.

D3Edit schaltet zu Raum B und ermittelt, wie unter Raum A beschrieben, die Entfernungen Zu Raum A und Raum C. Nachdem auch Raum C bearbeitet wurde ist die Liste der Max-Sound-Entfernungen fertig. Sie ist Bestandteil der `.d3l`-Datei und mit dem Level zusammen abgespeichert bzw. geladen.

Das Problem…

Im vorigen Beispiel würde man bei einem Laserschuss von Raum A nach Raum C den Einschlag in Raum C hören können, sofern die Entfernung < 400 ist.

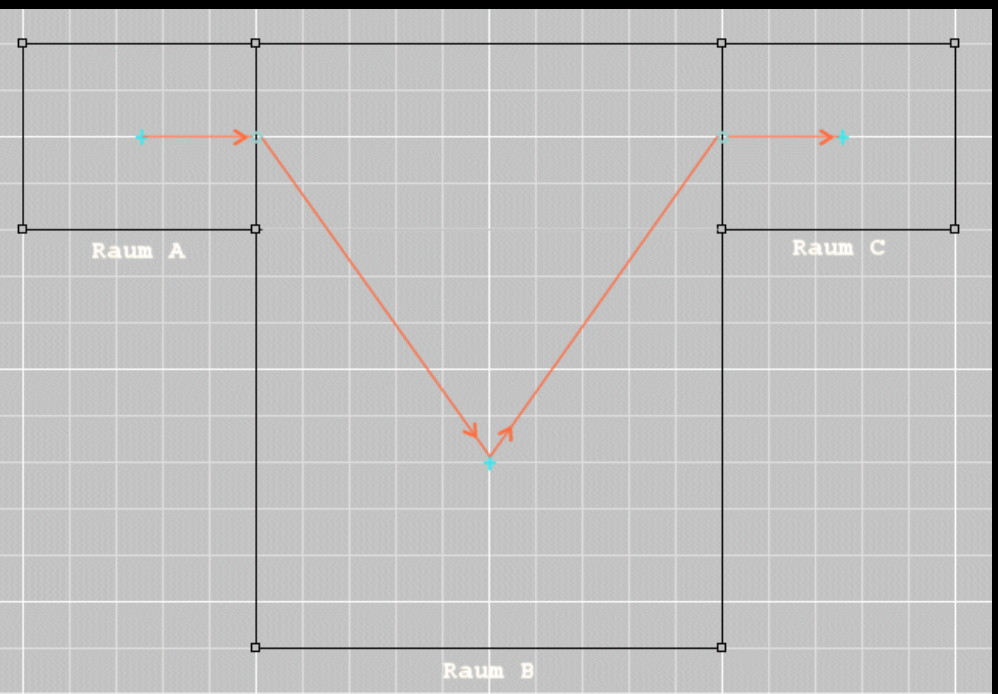
Schauen wir uns diesen Level mal an:



Auch hier schießen wir wieder von Raum A nach Raum C, aber wir hören keinen Einschlag in Raum C. Ein weiterer Testschuss direkt vor das Portal B/C ist aber zu hören. Fliegen wir mal in den Raum B und schießen erneut in Raum C, ist der Einschlag hörbar. Nun feuern wir aus Raum C in den Raum A, aber auch jetzt hören wir keinen Einschlag, ein Schuss direkt vor Portal B/A ist wieder ok. Fliegen wir von Raum C in den Raum B und schießen in den Raum A geht's auch… Die Räume A und C liegen doch dicht gegenüber, was ist nur los?
Wir erinnern uns daran, dass wir mit dem MD Zoom eine Entfernungsmessung vornehmen können. Also schauen wir mal nach und siehe da, es werden (z. B.) 250 Units angezeigt, also eine Entfernung die durchaus unterhalb der magischen 400er Grenze liegt. Warum hören wir aber trotzdem keinen Sound?

Die Lösung ist relativ einfach wenn man die o. a. Vorgehensweise einigermaßen kennt.
Schauen wir uns im Bild mal an wie die Entfernungen in diesem Level ermittelt werden: Ich habe das Room - Center von Raum B hier bewusst ein wenig weiter nach unten eingezeichnet um das Problem deutlicher hervorzuheben.

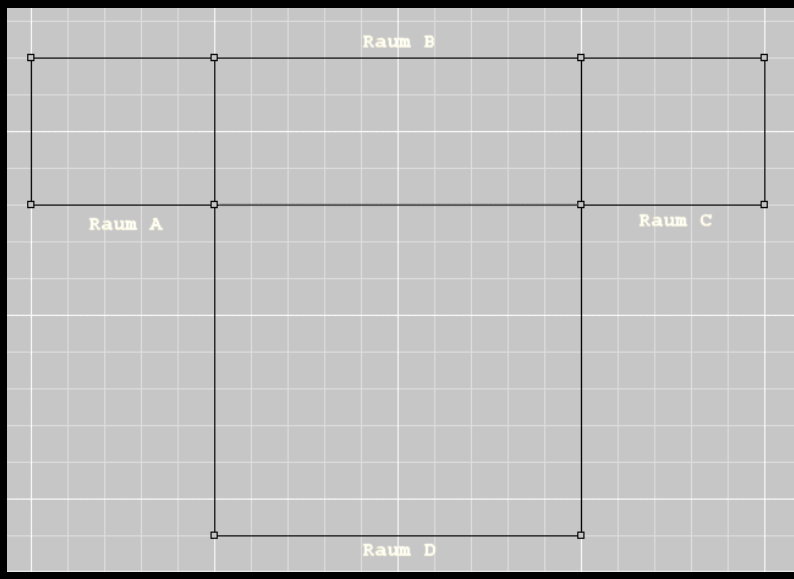
Wie war das gleich noch? Die Entfernungen werden von den Room Centern zu den Portalen berechnet. Darin liegt hier auch schon das Problem, denn obwohl die direkte Luftlinie zwischen Raum A und Raum C in dem Beispiel nur 250 Units beträgt, ermittelt D3Edit mehr als die doppelte Entfernung wie man unschwer aus dem Bild erkennen kann. Damit liegen wir dann aber weit über der zulässigen Soundentfernung und D3 spielt, beim Schuss von Raum A hinein in den Raum C, (und umgekehrt) keinen Aufschlag-Sound.



Abhilfe für dieses Problem…?

Solange D3Edit die Berechnung auf diese Weise durchführt können wir uns nur durch eine Änderung
des Leveldesigns behelfen. Hier ist es am einfachsten wenn wir den Raum B zerschneiden und einen weiteren Raum erzeugen.

Das folgende Bild zeigt den geänderten Level:

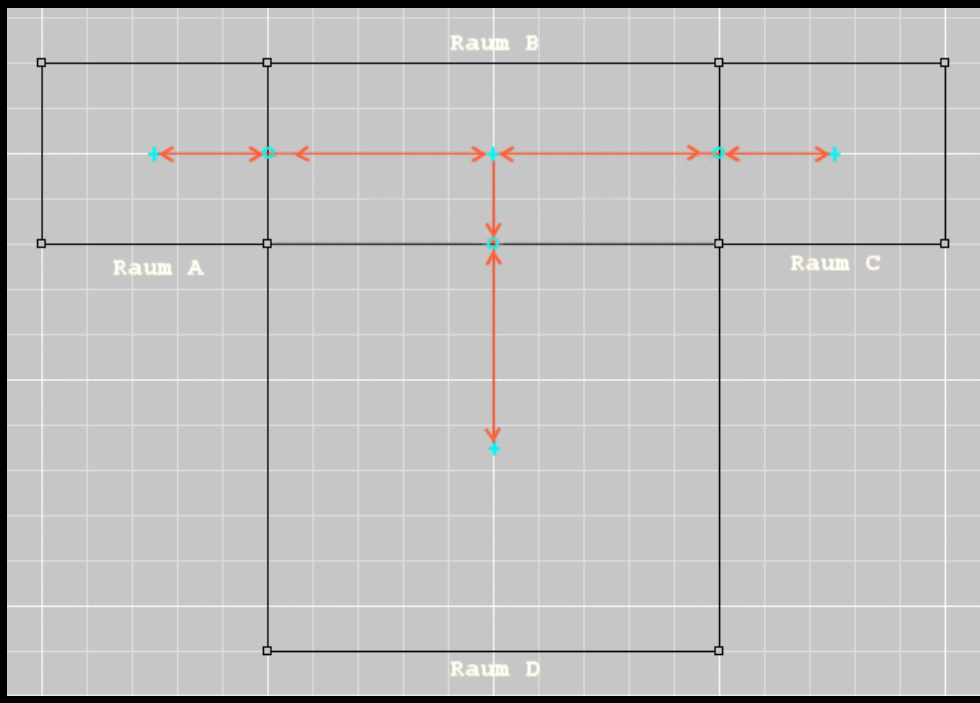


Die von D3Edit erzeugte Sicht-Tabelle sieht nun so aus:

Start-Raum	End-Raum
A	C
A	D
B	A
B	C
B	D
C	A
C	D

In diesem Level würde man bei einem Schuss von Raum A nach Raum C (und umgekehrt) den Einschlag jetzt hören. Im Spiel selber würde man am Leveldesign keinen Unterschied feststellen können, der Level sieht genauso aus wie die vorherige Version.

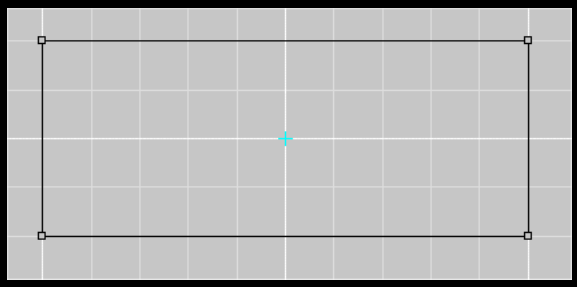
Das nächste Bild erklärt warum das Soundproblem nun beseitigt ist.



Man erkennt, dass die Entfernung Raum A nach Raum C nun der Luftlinie entspricht
und somit sicher in dem erlaubten Bereich liegt.

Ein weiteres Problem…

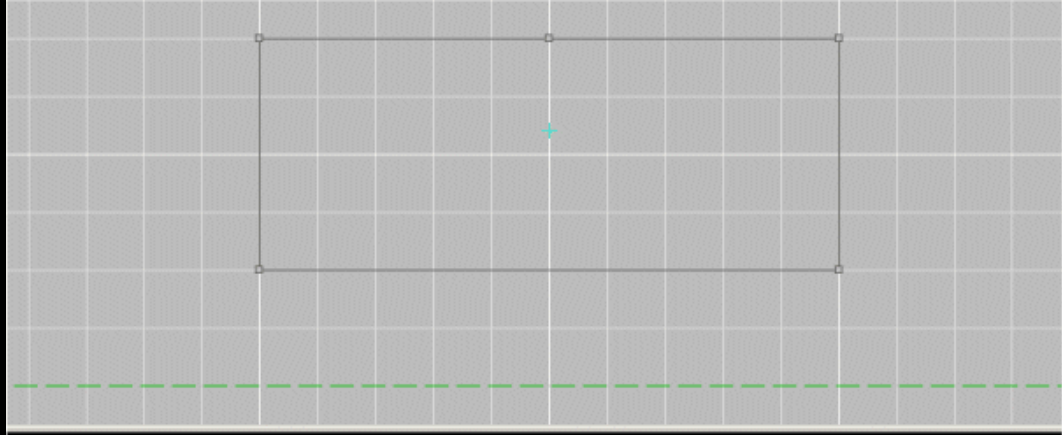
Eine Funktion in D3Edit wird so gut wie nie benutzt obwohl sie Soundprobleme verhindern könnte, es ist die View Room Center Funktion. Ist sie eingeschaltet erscheint in der Room View ein kleines Kreuz in der Mitte des Raumes. Man kann sich in allen 3 Ortho-Views ansehen wo sich der Raumschwerpunkt befindet. Wieso nenne ich es auf einmal Schwerpunkt und nicht Room Center oder Raum Mittelpunkt...? Schauen wir uns dazu wieder ein Beispiel (Draufsicht) an:

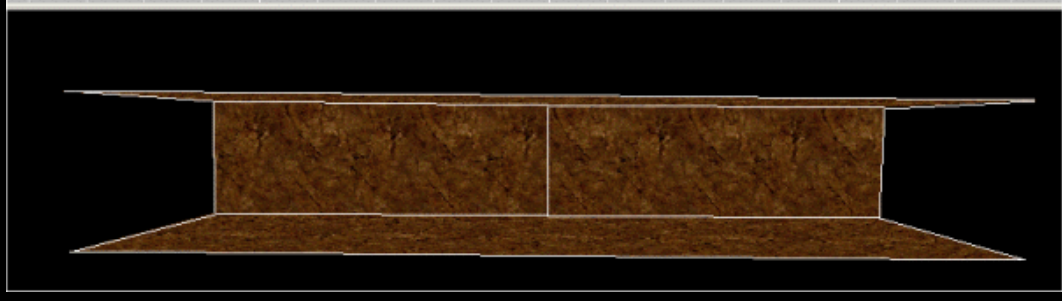


Das Room Center liegt wie erwartet mittig im Raum.

Nun Splitten wir ein Face des Raumes:

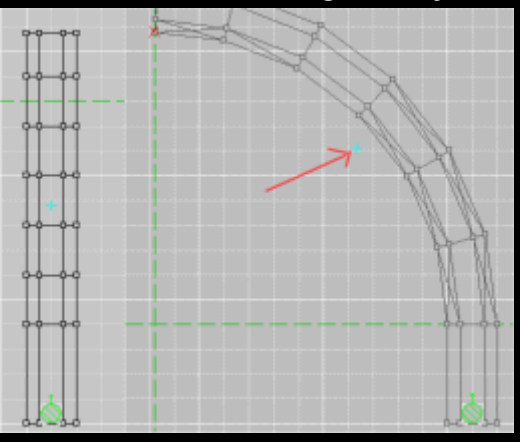
Obwohl der Raum noch dieselbe Form besitzt, verschiebt sich das Room Center in Richtung der neuen Verts. Die Berechnung des Room Centers ist mehr eine Gewichtung der Verts. Ein Face mit vielen Verts zieht das Room Center magisch an. Diese Eigenschaft kann Soundfehler erzeugen aber auch helfen Soundfehler zu beheben.

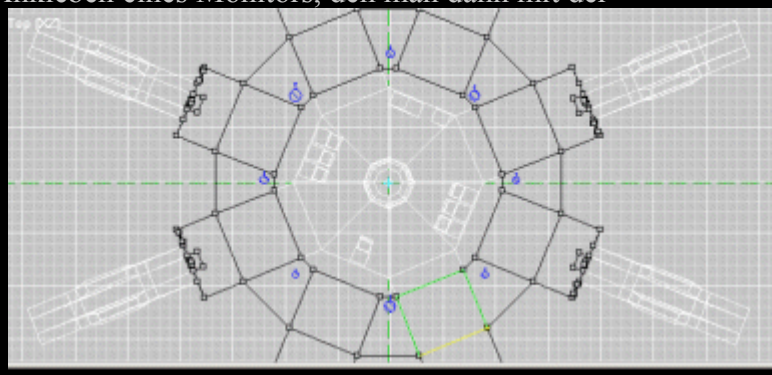




Beispiel:

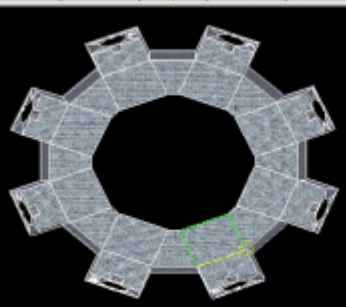
Nach dem Benden hat sich das Room – Center verschoben. Die Gewichtungsberechnung ermittelt einen Schwerpunkt ***außerhalb*** des Tunnels (siehe roter Pfeil). Das hat schwerwiegende Folgen für die Soundberechnung. Gehen wir mal davon aus, das der gebogene Raum der Raum C in unserem obigen Beispiel sei. Dann würde die Linie von Raum A durch Raum B bis zum Portal B/C verlaufen, aber die (Sicht-)Linie zwischen dem Portal B/C und dem Raum Center C (C_Mi) ist unterbrochen. D3Edit würde die Soundverbindung zwischen Raum A und Raum C für ungültig erklären, die Folge wäre daher das es keinen Treffersound wie oben beschrieben geben wird. Eine Lösung wäre aus dem Bogen zwei oder mehr Räume zu machen, oder aber, wenn der Abstand nicht zu groß ist, an der gegenüberliegenden Wandseite die Gewichtung der Verts zu erhöhen. Denkbar auch durch Ankleben eines Monitors, den man dann mit der Funktion Merge Object into Room Bestandteil des Raumes werden lässt. Die neuen Faces und Verts ziehen den Schwerpunkt u. U. wieder soweit in den Raum das das Room Center von den Portalen aus gesehen werden kann. Damit funktioniert dann auch wieder der Sound.





Hier noch ein schönes Beispiel was man nicht machen sollte:

Der Raum Mittelpunkt liegt perfekt in der Mitte, nur leider im falschen Raum ☺



Portale:

Die neueren D3Edit Versionen berücksichtigen inzwischen, dass die Portaltexturen durchlässig sein müssen wenn man keine Soundprobleme im Level haben will. Es wird automatisch die Palmleaf1 Textur auf Portale gesetzt.
Das ist die einzige D3Texture die für die o. a. Sichtlinien durchlässig ist und somit eine korrekte Soundberechnung durch D3Edit möglich macht. Möchte man den Sound kappen, dann muss man nur eine andere Textur auf ein Portal legen, hinter diesem Portal hört man dann keinen Treffersound mehr. Die Benutzung der Pamleaf1 Textur bringt auch Vorteile bei den BNodes.

```
Files:
tools/coding/d3-sdk14.zip
tools/coding/egcs-1.1.2-mingw32.zip
```

Vorgriff: DALLAS zum Laufen kriegen

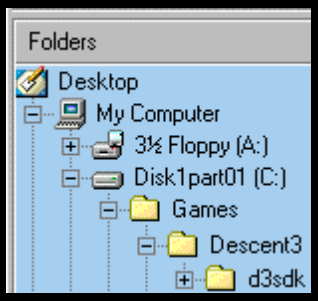
Obwohl Skripting eigentlich ein späteres Thema ist, wird Dallas schon ab hier benötigt. Daher bereits an dieser Stelle die Anleitung, wie man es installiert, einrichtet und zum Laufen bekommt.

Es ist Fischleins Übersetzung einer Anleitung von Papacat.

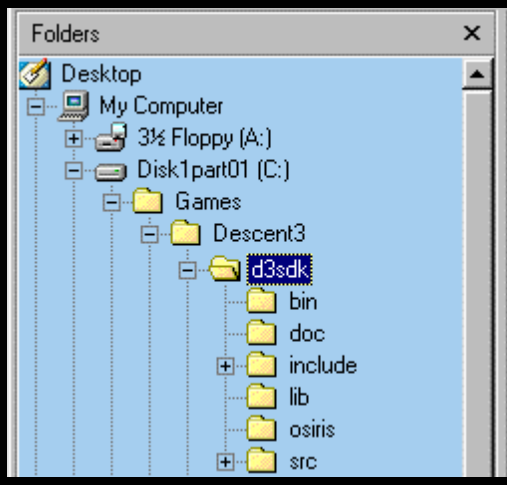
<_____[______***_/_____]_____>

Ich gebe nicht wortwörtlich wieder, was papacat von Gameedit in englisch geschrieben hat !

Alle Bilder die ich hier verwende, sind von Gameedit

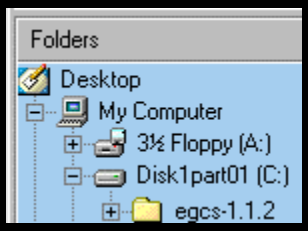


Zunächst braucht Ihr die Datei `d3-sdk14.zip` (Ihr solltet nach der aktuellen Version schauen: http://descent3.com/4_downloads.html), das wie folgt installiert wird.
Zunächst erstellt Ihr ein Verzeichnis (links).



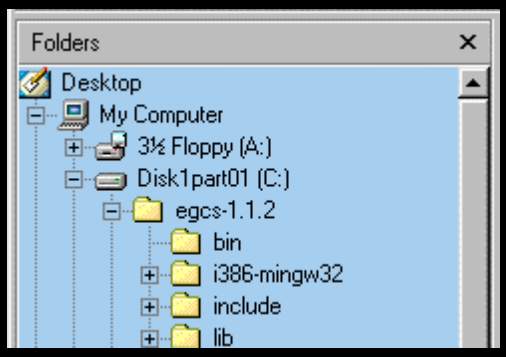
Entpackt nun die Datei `d3-sdk14.zip` mit Verzeichnisstruktur (rechts).

Als nächstes braucht man noch **EGCS-1.1.2**. Nötigenfalls googlet euch ‚egcs-1.1.2-mingw32'. Das ist eigentlich was für Leute die in C/ C++ programmieren, aber da Dallas C-artigen Code erzeugt, benötigt man das um aus seinem Code eine DLL - Datei zu machen (=Kompilieren).



Erstellt ein Verzeichnis "egcs-1.1.2" (links).

Diese egcs-1.1.2-mingw32.zip entpackt Ihr auch mit Verzeichnisstruktur (rechts).

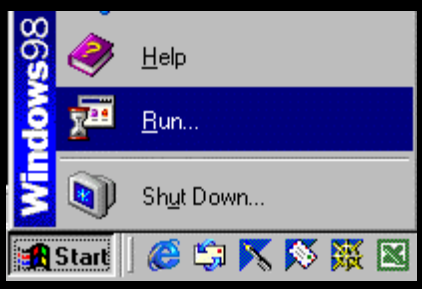


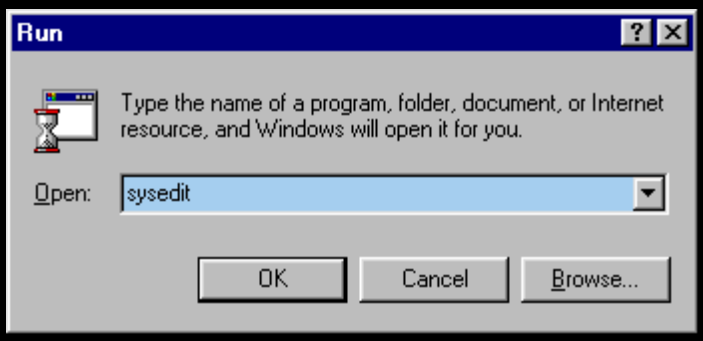
Jetzt müsst Ihr noch die folgende Anweisung in Eure `autoexec.bat` einfügen:

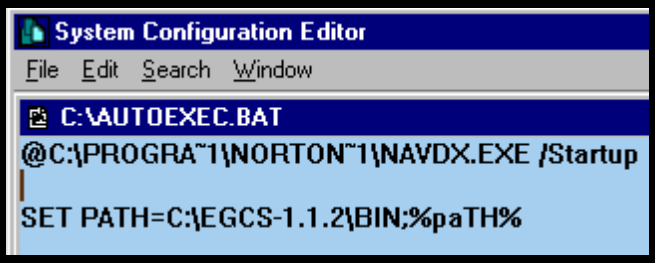
```
"SET PATH=C:\EGCS-1.1.2\BIN;%PATH%"
```

dann die Datei speichern und neustarten.

Für diejenigen, die noch nicht an den Startdateien 'rumgefriemelt haben:
In der Deutschen Version Ausführen... bei Öffnen schreibt Ihr "`sysedit`" rein und **OK** klicken





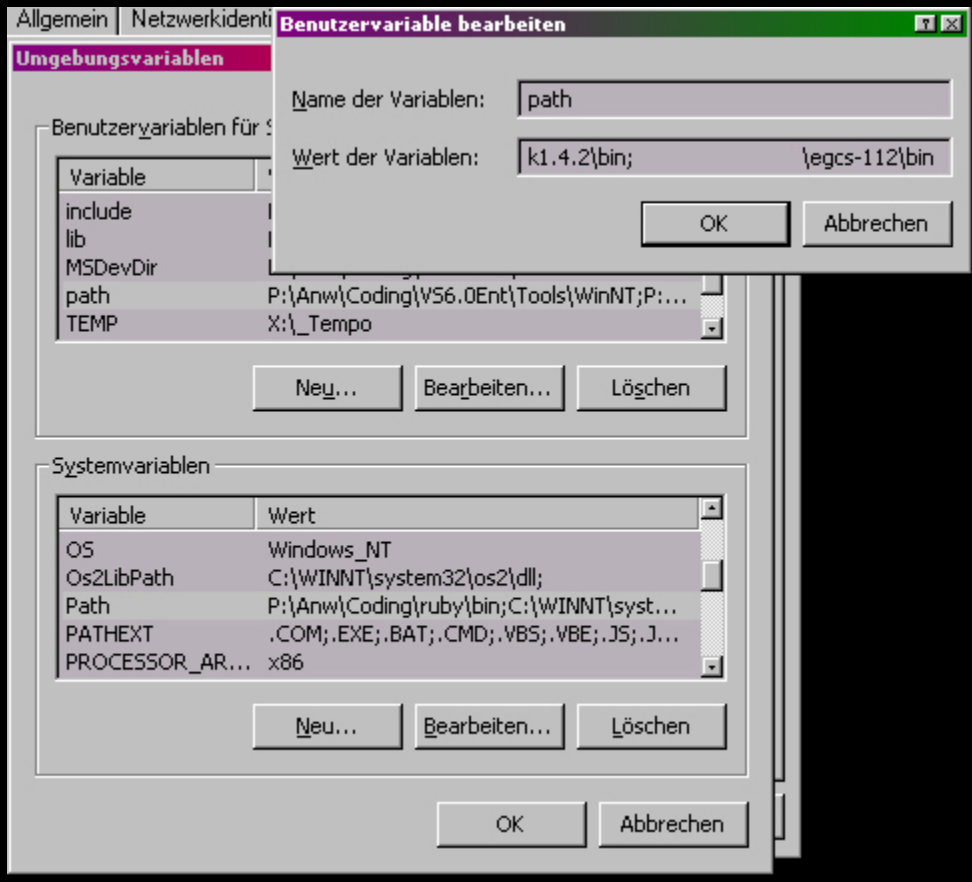


fügt die obige Anweisung ein, dann Datei -> Speichern. Nun noch neustarten, fertig.

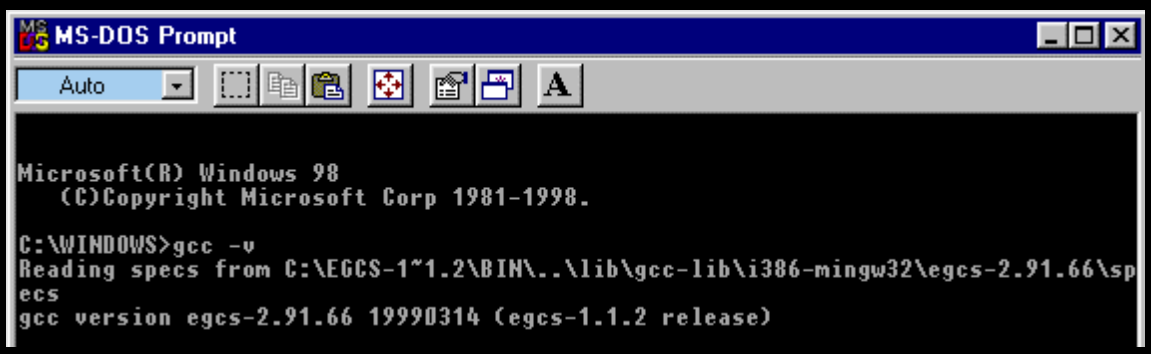
Alternativ könnt Ihr auch folgendes machen:

Arbeitsplatz, Rechtsklick -> Eigenschaften. Dort den Tab Erweitert wählen, da drinnen den Button **Umgebungsvariablen**:

Markiert die hier rechts heller unterlegten Einträge und klickt auf **Bearbeiten...**
hängt dann den Pfad zum Compiler dort dazu, bei beiden. Dort wo im Screenie leer ist, kommt Euer Pfad hin. Strichpunkt nicht vergessen ☺ !



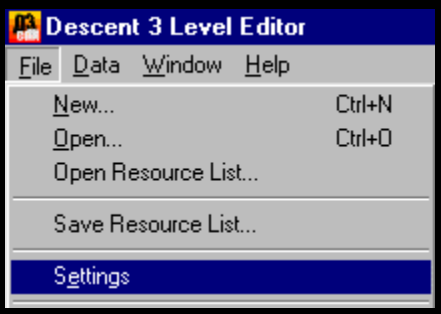
Nachdem Ihr neugestartet habt, öffnet eine Dosbox oder "MS-DOS-Eingabeaufforderung". Nun gebt Ihr folgendes ein: "`gcc -v`". Zwischen **GCC** und **-v** muss ein Leerzeichen sein ! Nach dieser Eingabe sollte folgendes zu sehen sein:



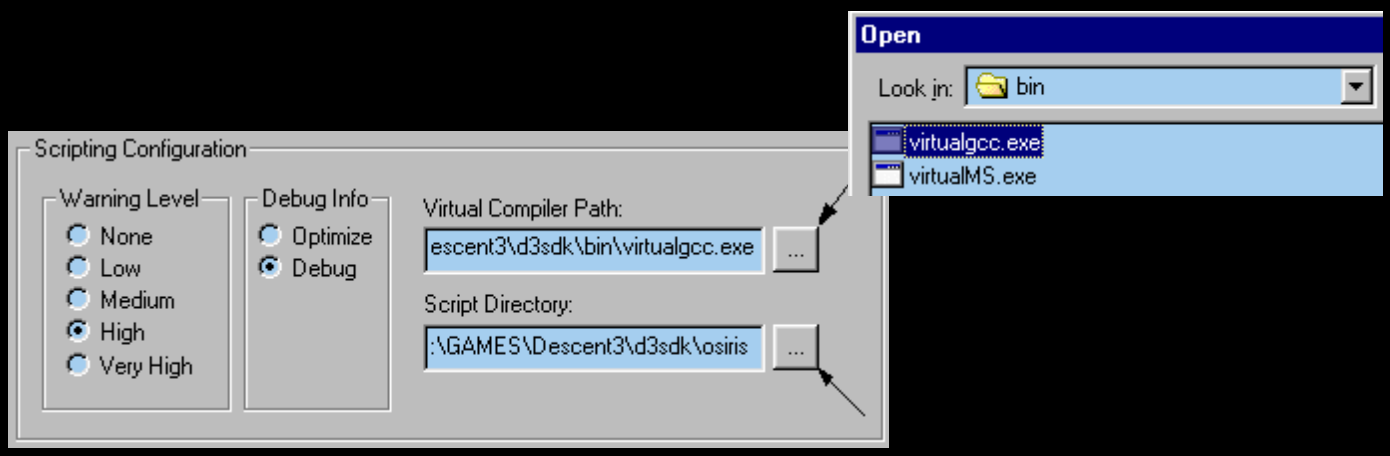
also, wenn die GCC-Versionsnummer angezeigt wird, ist es richtig installiert.

Öffnet jetzt D3Edit, hier müssen wir auch noch ein paar Einstellungen vornehmen.

Die Konfiguration aufrufen:



Bem unteren Teil, "Scripting Configuration", müßt Ihr den Pfad zm Compiler und dem Verzeichnis, in dem Ihr die Scripte und DLL's speichern wollt angeben. Bei "Virtual Compiler Path" müßt Ihr die `virtualgcc.exe` eintragen. Bei "Script Directory" das Verzeichnis `osiris`, wie im Bild.



So da war's, jetzt könnt Ihr Teleporter, Stargates und andere Effekte in Eure Level einbauen.

Abschnitt H – Türen & Objekte

Türen sind zwar eine Sache, die man einfach im Levelbau verwenden kann, doch es sind auch Objekte, die man selbst erstellen kann; der thematische Übergang ist so fließend, daß hier beides abgehandelt wird.
Zusätzlich ist es ab hier sinvoll, den Abschnitt über Sounds im Game durchgearbeitet zu haben.

Ferner gilt ab hier: mit ‚GAM' ist die in der `.gam`-Datei eingetragene Speicherseite gemeint.
Siehe dazu auch die Spezifikations-File am Ende (Seite 494) dieser Datei.

		Türen		
071	**Doors Tutorial**	Wie man Türen in Level einbaut.	Kyouryuu	273
072	**Eigene Doors**	Wie man selbst Türen baut, erzeugt und einbindet.	Toxxeon	276
073	**Sound für die eigene Tür**	Einer selbstgemachten Tür die entsprechenden Sounds verpassen.	Ragil Ral	282
		Objekte		
074	**Eigene Objekte 1**	Eigene Objekte erstellen & einbinden	Fischlein	283
075	**Eigene Objekte 2**	Vertiefung ins GamFile & Objekten mit Dallas "Leben" einhauchen.	Fischlein	285
076	**Eigene Objekte 3**	Gun Points zum Objekt hinzufügen & das Script dazu.	Fischlein	289
077	**Custom Robots Part 1**	Damit sich das Objekt wehren kann	Papacat	293
		= Papacat's Custom Weapon Workshop = Sehr umfangreicher Durchgang für dasErstellen und Einbinden einer selbsthergestellten Primärwaffe, von der Theorie bis zum Einsatz!		
078	**Einführung & Übersicht**	Überblick über den Arbeitsablauf	Papacat	296
079	**Die Powerup GAM**	Erklärung der GAM-Seite für Powerups		298
080	**Die Weapon GAM**	Einstellungen in der Waffen-GAM		299
081	**Eine simple Primärwaffe**	Einbinden der Waffe		303
082	**Die Schiffs-GAM**	Einstellungen der Schiffs-GAM		307
		= Papacat's OOFEdit-Exkurs = Eine genaue Erklärung was man mit OOFEdit so alles treiben kann.		
	· **OOFEdit-Exkurs**	Eine Übersicht was, warum & wie	Papacat	309
083	**OOFEdit: Terminologie**	Die wichtigsten Begriffe		311
084	**OOFEdit: Features**	Was kann OOFEdit alles?		312
085	**OOFEdit: Erste Schritte**	Bedienung des Programms		313
086	**OOFEdit: Ein Powerup**	Ein Powerup-`.oof` machen		316
087	**OOFEdit: Turrets**	Einen Gefechtsturm erzeugen		318
088	**OOFEdit: Animation**	Grundlagen zur Animation		321
089	**Ship Tutorial**	Einbinden eines Custom-Schiffes	Darkside Heartless	325

Toc

071 - Doors Tutorial

Kyouryuu

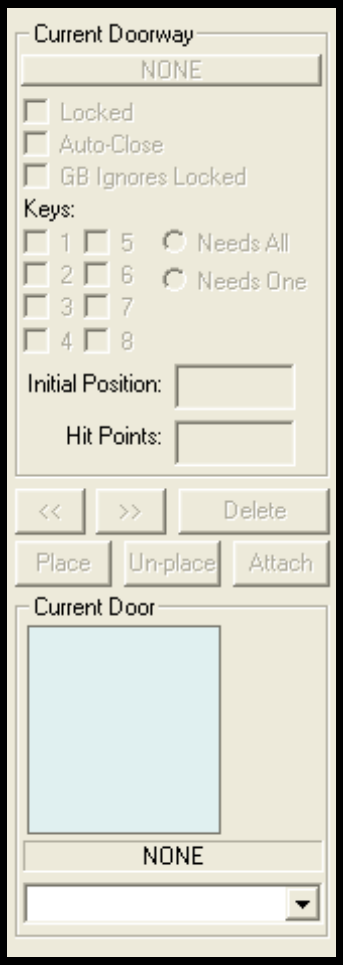
In diesem Tut lernst Du, wie Du Türen in Dein Descent3-Level einbaust. Hier mußt Du Räume verstehen und wissen, wie man sie anfügt.

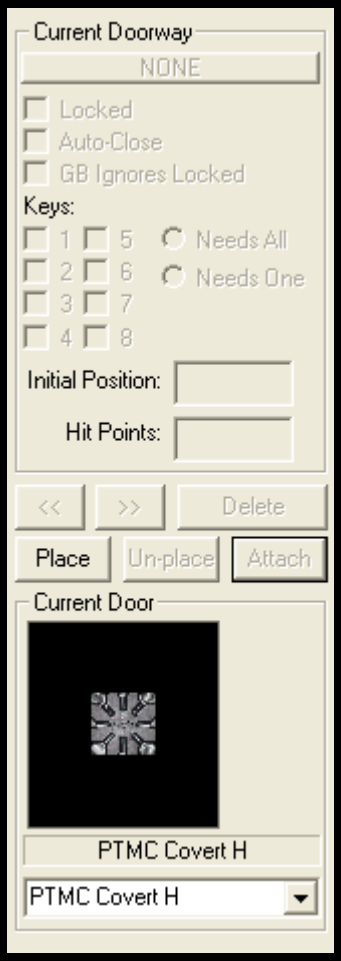
Einleitung

Türen in Descent 3 sind essentiell für Single-Player-Missionen. Sie sind großartig um Bereiche abzutrennen, und so zu verhindern daß der Spieler vorzeitig in bestimmte Bereiche gelangt. In Multiplayer-Levels werden Türen für gewöhnlich nicht verwendet. Viele Spieler beklagen sich wegen des Lags, der durch die Türen verursacht wird, da Türen eigentlich als Objekt im Level betrachtet werden, genau wie Powerups und die Schiffe selbst.
Um das zu verstehen, müssen wir uns zurückerinnern daß in Descent 1 und 2 Türen komplett flache Polygone waren, die eine animierte Textur drauf hatten die wie eine Tür aussah. Wenn ein Spieler auf die Tür schoß, hätte das Spiel eine vordefinierte Serie von Animationsclips abgespielt, die darin resultierten daß sich die Tür ‚öffnete' und den Spieler passieren ließ.

Descent 3 ist anders. Hier sind die Türen komplett in 3d, genau wie es all die andere Levelarchitektur ist. Die Teile der Tür werden sogar in Spalten zurückgezogen, wenn sie sich öffnet (Denke an einen Aufzug in einem Bürogebäude). Im Speziellen enthält eine Tür in D3 einen speziellen Raum (Den Rahmen um die Tür) und ein Tür-Objekt. Wenn Du eine Tür einsetzt, setzt Du eigentlich beides ein.

Das Türen-Paneel



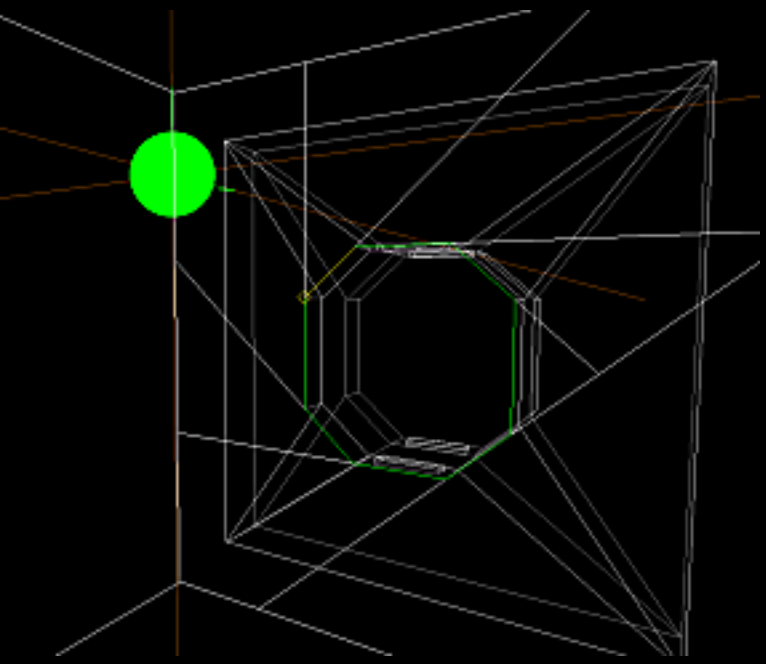


Türen haben ihre eigene Toolbar in D3Edit. Es ist das D im oberen Bereich des Editors. Der Button aktiviert das hier gezeigte Paneel.

Wenn Du erstmals das Door-Paneel öffnest, wirst Du wahrscheinlich das Linke der beiden Bilder sehen, alles ist ausgegraut. Obwohl, wenn Du das Menü im unteren Bereich öffnest, kannst Du aus einer ganzen Menagerie von Türen auswählen. Wähle zum Beispiel PTMC Covert H. Nun belebt sich das Menü, da eine ganze Option aktiv wird - **Place**. Nun, wenn Dir geläufig ist wie man Räume einfügt, sollte Dir das recht offensichtlich sein.

Anmerkung: Am Face 0 des ersten Raumes konnte bis v40 keine Tür angefügt werden!

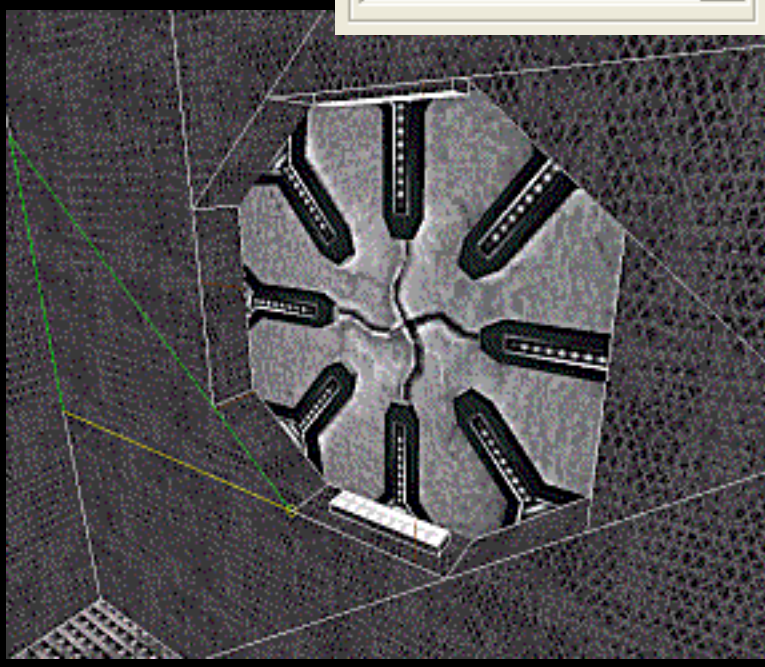
Im Bild rechts habe ich Wireframe in meiner World View aktiviert. Wähle ein current Face im Raum. Wenn Du **Place** klickst, siehst Du eine purpurne Liniendarstellung der Tür an diesem current Face. Du magst Dich vielleicht fragen warum die Tür so groß ist; das ist weil der Rahmen Spalten für die Tür haben muß, wo sie sich beim Öffnen hinein zurückzieht. Wie ein Aufzug in einem Bürogebäude, Gleiten die Tür-Komponenten in die Schlupfwinkel in der Wand zurück.



Nun wird **Attach** im Door-Paneel verfügbar, also klicke es. Du wirst in D3Edit sehen, wie es die Tür einsetzt.

Gratulation! Die Tür wurde zu Deinem Level hinzugefügt.

Zurück im Textured wh Outline Mode, sollten Dir ein paar Dinge auffallen. Erstens, der Türrahmen übernimmt die current Textur, die Du im Texturen-Paneel gesetzt hast. Wenn Du das nicht willst, kannst Du auf den Türraum klicken und ihn in der Raumansicht bearbeiten. Beachte, daß wenn Du in einer der texturierten Ansichten bist und auf einen Türraum klickst zentriert der Editor die Ansicht darauf. Zweitens wirst Du bemerken daß manche Türen Einbauten an den Türraumseiten haben. Dies sind die Stellen, wo Du eine Licht-Textur aufbringen kannst wenn Du willst. Im Bild rechts habe ich das gemacht.



Nocheins. Du kannst die Buttons << und >> im Türen-Paneel verwenden, um durch die Türen in Deinem Level zu cyclen.

Du kannst hier ein paar interessante Sachen machen.

Wenn Du auf den Button klickst, der mit „PTMC Covert H“ beschriftet ist, kannst Du der Tür einen Namen geben. Das ist nützlich, wenn Du die Tür in einem DALLAS-Skript referenzieren willst.

‚Locked' sagt dem Spiel ob die Tür verschlossen ist oder nicht und gibt die Standard-Nachricht am HUD aus, wenn der Spieler die Tür berührt oder anschießt. Du kannst das DALLAS-Kommando UNLOCK Door <name> verwenden um die Tür aufzusperren (oder Umgekehrt, mit LOCK verschließen).
‚Auto-Close' bestimmt, ob die Tür offen bleibt oder sie sich nach einer vorbestimmten Zeit wieder schließt.
Das Keys:-Zeugs ist etwas strange und ich hatte bis jetzt keinen Grund es zu verwenden. Ich werde hier abschweifen, also gehe zum nächsten Punkt außer Du bist neugierig und verstehst DALLAS. Es wird von DALLAS im Player-Kommando „**`Give player [PlayerObject] key [KeyObject], key number [KeyNum], name = [KeyName], show HUD message = [Yes/No].`**“ verwendet. So haben sie in Novak Corporate Prison (Retribution Level 2) den Schlüsseln Namen gegeben (wie "X-1 Security Pass"), und die **`[KeyNum]`** referenziert höchstwahrscheinlich eine Ziffer zwischen 1 und 8 aus dem Door-Paneel. Auf diese Art solltest Du dann wirkliche Schlüssel für Türen haben, die tatsächlich als Schlüssel funktionieren. Das heißt, daß in einem COOP-Spiel jeder seinen eigenen Schlüssel haben muß. In normalen SP-Levels ist es manchmal eher vorzuziehen, die Tür beim Aufnehmen des Schlüssels vom Spieler einfach aufzusperren, anstatt ihn ins Invenory aufzunehmen und mit Namen und Zahlen herumzuhantieren. Ich habe letzteres nicht gemacht, aber Du kannst im Level 2 DALLAS Source Script nachsehen wenn Du Dich daran versuchen willst. (`071-Level2-DALLASSourceScript.rar`)
Initial Position läßt Dich bestimmen, bei welcher Position eine Tür steht, wenn Du den Level startest. Verwende das, um eine Tür halb zu öffnen.
Hit Points kontrollieren, wie oft Du auf die Tür schießen mußt, bevor sie kaputtgeht. Das ist auf gitter- und rostartige Türen anwendbar, wie etwa die Kanaldeckel im Seoul-Level (Level 4, das ist der mit der sch...önen U-Bahn 😬)

Türen können auch gelöscht werden, unter Verwendung des Buttons **Delete**. Es ist strengstens empfohlen, daß Du diese Delete-Funktion verwendest anstatt den Raum zu löschen wie jeden anderen Raum. Du willst ja sicherstellen, daß Du das Tür-Objekt mit dem Raum mitlöschst. Achte auch darauf, daß die Tür die Du löschen willst auch im Door-Paneel angezeigt wird.

File: 072-own_door.zip

072 - Eigene Doors

Toxxeon

In dieser Anleitung wird erläutert, wie man eigene Doors erstellt und in die Level einfügt.

Wir brauchen neben einem aktuellen D3-Editor noch den **OOF-Editor**, zum Extrahieren der `Table.gam` den **HOG2-Workshop** sowie zum Erstellen eigener `.gam`-Files das **gamTool**. Wenn ihr diese Programme noch nicht habt, dann ladet sie euch runter.

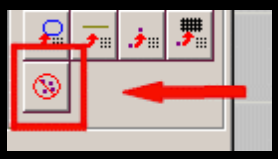
Zum Anfang erst mal als Überblick:

Eine Tür ...Door... ist ein Raum mit einem oder mehreren animierten Objekten. Die Bewegung dieser Objekte wird per Script gesteuert. Um eine selbsterstellte Tür in das Level zu integrieren, müssen wir eine vorhandene Tür mit unserer ersetzen. Das hat den Vorteil, dass wir für unsere Tür kein Extra-Script erstellen brauchen. Wir nutzen einfach das der Original-Tür. Aber immer der Reihe nach...

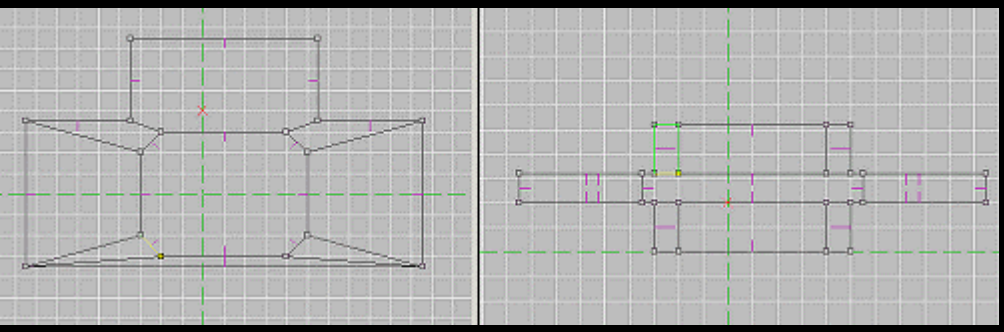
Wir werden in dieser Anleitung eine dreiteilige Tür bauen, bei der sich die einzelnen Tür-Objekte in verschiedene Richtungen öffnen.

Öffnet im D3-Eitor einen neuen Raum. Erstellt erst mal den "Türrahmen", so wie ihr ihn speziell in eurem Level benötigt. Ihr braucht einen Durchgang, durch den ihr später durchfliegen wollt, und seitlich (bzw. ober- oder unterhalb) auch noch Taschen. Dorthin schieben sich dann die Türobjekte hinein. Denkt daran, dass dieser Raum ringsrum geschlossen ist. Weil sonst kann man aus dem Level "hinausschauen".

SEHR WICHTIG: Hier müssen **erstmal** die Faces **unbedingt ein geschlossenes Polygonnetz bilden!** Es dürfen also keine Extra-Vertices übrigbleiben, wenn ihr Faces kopiert und aneinanderschiebt. Wenn ihr doppelte Verts habt, dann klickt hier drauf. (Bild rechts)



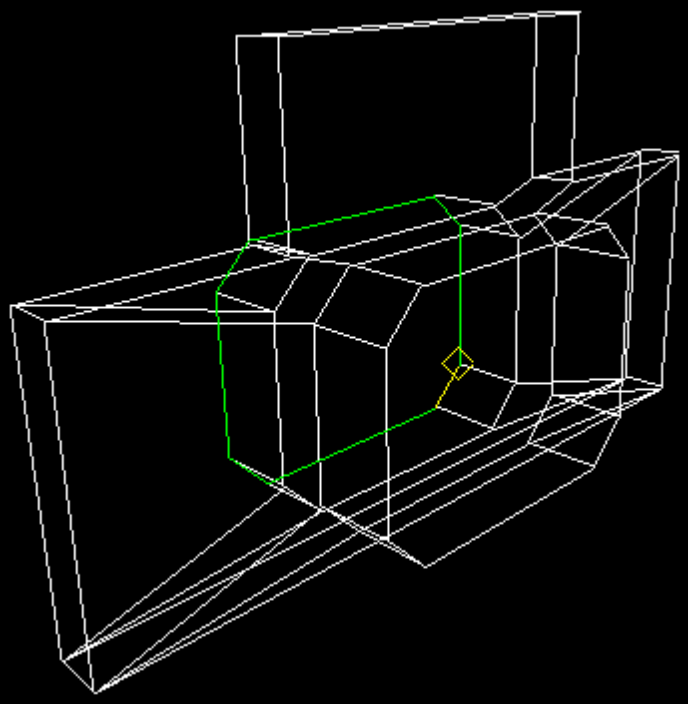
Ihr könnt euren "Türrahmen" hier nach eurem Geschmack texturieren, um in der 3D- Ansicht einen besseren Überblick zu bekommen. Manchmal werden allerdings die an dieser Stelle definierten Texturen nicht mit übernommen. Dann müsst ihr das später, wenn die Tür in unsere `*.d3l`-Datei eingefügt ist, noch mal extra machen. Aber das ist kein Problem. O.K. Unser Türrahmen ist fast fertig...und sieht so aus (Bild links)



Wie schon weiter oben angesprochen, muss bis hierhin alles ein geschlossenes Polygonnetz sein. Jedoch müssen wir nun **eine** Fläche, die später als *FrontFace* dient, aus diesem Netz lösen! Diese Fläche muss ein späteres Portal sein!

Dazu klickt ihr am einfachsten in der 3D-Ansicht eine Fläche an, die dieses FrontFace werden soll, und markiert es mit der **Leertaste**. (nächstes Bild) Dann kopiert Ihr es in den Zwischenspeicher. Nun markiert ihr das Face wieder und löscht es. Anschließend in eine 2D-Ansicht klicken und das Face aus dem Zwischenspeicher wieder in den Editor einfügen. Jetzt das Face am Grid ausrichten und genau wieder an den Platz des soeben gelöschten Faces verschieben.

Ihr könnt das Face natürlich auch mit **Shift-Strg-V** einfügen, so landet es genau am richtigen Platz ☺

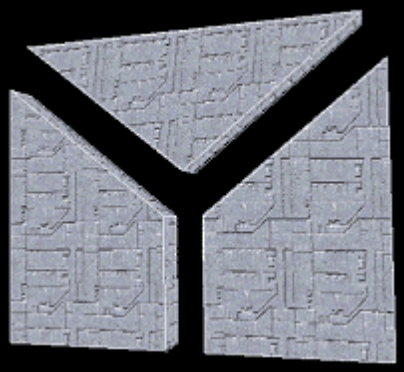


ACHTUNG: Ab jetzt dürft ihr NICHT MEHR DIE EXTRA-VERTICES LÖSCHEN!!! Sonst wird das Polygonnetz wieder verbunden!

Was wir bis jetzt modelliert haben, wird unser Door-Shell und das FrontFace. Aber dazu später.

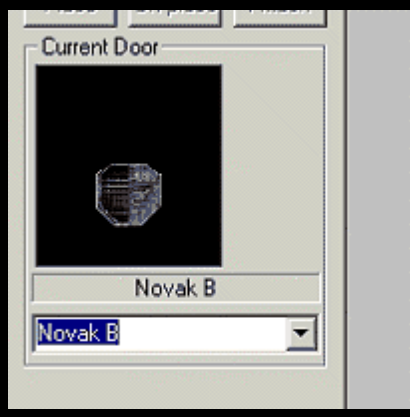
Nun zu den Tür-Flügeln:

Die erstellt ihr am günstigsten an einer anderen Stelle im selben Raum und schiebt sie erst ganz zum Schluss an die richtige Position. Genauer gesagt, an die Positionen, so wie der geschlossene Zustand der Tür aussehen soll. Diese Tür-Flügel könnten bei einer dreiteiligen Tür so aussehen. (Bild links) Im Bild sind die drei Teile nur zum besseren Erkennen noch etwas auseinandergeschoben...



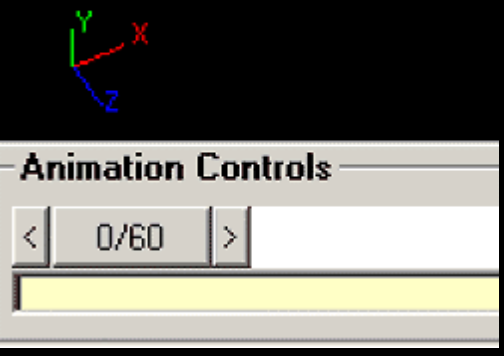
Und nicht vergessen: **Die Extra-Vertices nicht löschen!** Jede Einzel-Tür muss ein selbstständiges Polygonnetz sein. Außerdem müsst ihr hier schon festlegen, welche Textur die Türflügel haben sollen. Die Texturen, die man den Flügeln zuweist, werden übernommen und können dann nicht mehr geändert werden!
Speichert den Raum unter "`own_door.orf`".

Jetzt erstellen wir unsere Tür mit den Tür-Objekten...
Als erstes schauen wir im D3-Editor nach, welche Tür wir ersetzen wollen. In diesem Beispiel nehmen wir dazu *Novak B* (Bild rechts)



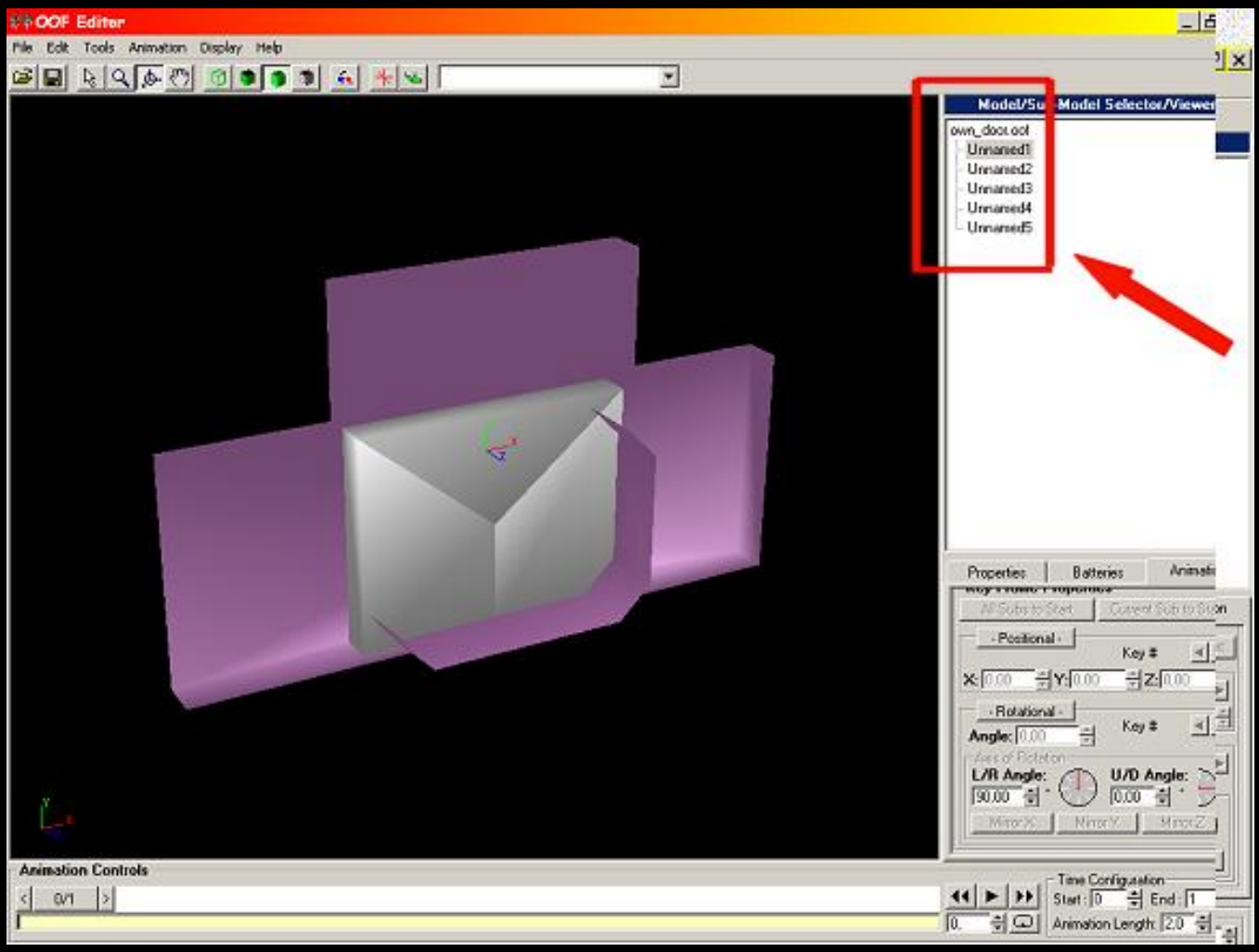
Dann öffnet ihr mit *HOG2-Workshop* die Datei `d3.hog` und extrahiert die darin befindlichen Dateien `Table.gam` und `novak_b.oof`.

Jetzt startetet ihr das Programm **OOFEditor.exe** und ladet das Objekt `novak_b.oof`. Schaut in der linken unteren Ecke nach, wieviel Key's die Animation dieses Objektes besitzt. Hier hat der letzte Key die Nummer "60" (Bild links). Das ist wichtig, weil **unsere neue Tür muss die identische Key-Anzahl besitzen, damit's funktioniert!**



So, jetzt öffnet ihr im OOFEditor einfach unseren Raum `own_door.orf`. Das Ergebnis seht ihr im folgenden Bild...

Mehr zum OOF-Editor später, bei Papacat's OOFEdit-Exkurs.

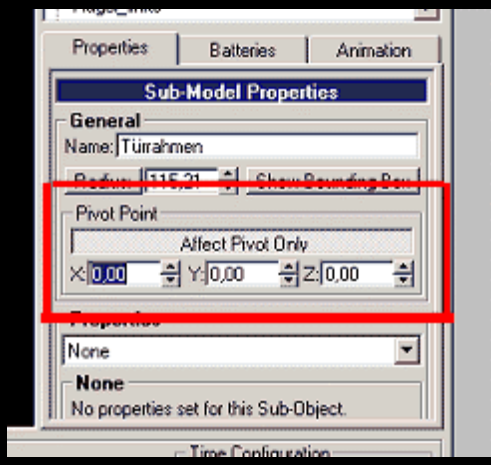


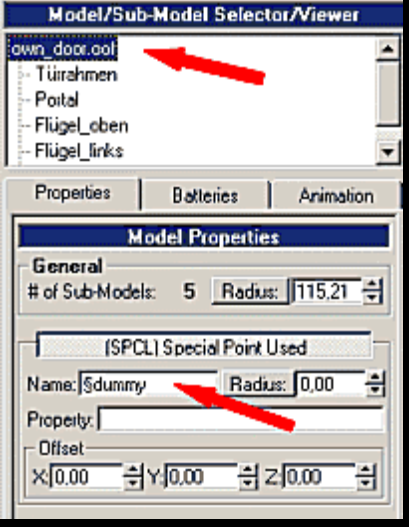
Auf der rechten Seite (oben) seht ihr den Namen unseres Objektes mit allen seinen Unterobjekten = *Sub-Models*. Bei uns sind das 5 Stück, sie sind alle mit *Unnamed* bezeichnet. Wenn ihr die der Reihe nach anklickt, dann färben sie sich in der 3D-Ansicht fliederfarben ein. Hier seht ihr dann auch, dass der Türrahmen als ein Körper zu sehen ist...und das die einzelne Portal-Fläche als eigenes Sub-Model vorhanden ist. Nun könnt ihr den Sub-Models unter *Properties* erstmal beliebige Namen vergeben. Ich nenne sie:

- `Türrahmen`
- `Portal`
- `Flügel_oben`
- `Flügel_links`
- `Flügel_rechts`

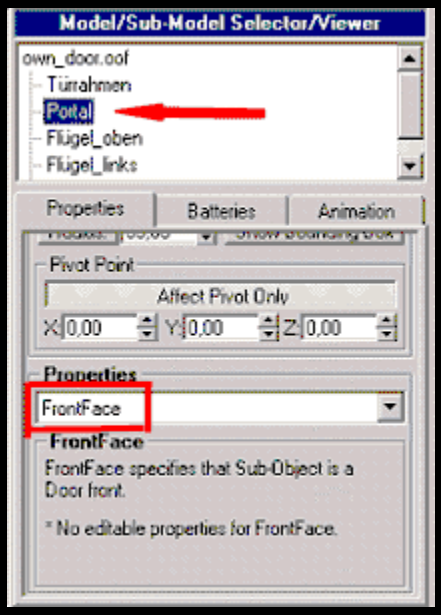
Jetzt alles unter `own_door.oof` speichern!

Anmerkung:
Wenn ihr sehr komplexe Türen erstellt, kann das Tür-Objekt sehr viele Sub-Models besitzen. Um da ein wenig mehr Übersicht zu erlangen, kann man auch die Sub-Models untereinander *verschmelzen*. Diese können auch räumlich voneinander getrennt sein. Schaut euch also genau an, welche Türteile zum Beispiel die selben Bewegungen haben sollen.
Dann zieht im **Model/Sub-Model Selector/Viewer** bei gedrückter Shift-Taste per Drag'n Drop das eine auf das andere Sub-Model. Fertig. Damit haben dann diese Objektteile die gleichen Eigenschaften und Bewegungen...
In unserem Beispiel hier dürfen wir das jedoch nicht machen!
Als nächstes aktiviert ihr das erste Sub-Model und klickt dann im Register **Properties** den Button **Affekt Pivot Only** an. Dort setzt ihr dann **alle** Koordinaten (X,Y,Z) auf **0.00** (Bild rechts) Macht das bei jedem Sub-Model, weil sonst später die Türflügel nicht genau an der gewünschten Position sind. Und lasst bei jedem Submodel **den Button aktiviert!**





Markiert nun im Model/Sub-Model Selector/Viewer das (Haupt)-Model und klickt dann den Button (SPCL) Special Point Used an, und gebt dann dort unter Name einfach `$dummy` (Dollar-Zeichen und dummy reinschreiben) ein. So wie's im Bild links zu sehen ist. Lasst den Button dann aktiviert!



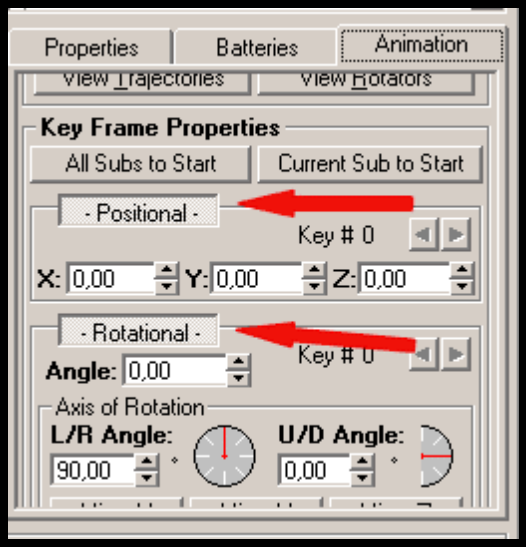
Wichtig: Jetzt müssen wir dem Submodel Türrahmen im Untermenü Properties die Eigenschaft Shell zuweisen. Klickt auf das Pull-Down-Menü und sucht das Entsprechende aus. (Bild rechts)
Genauso macht ihr das mit dem Submodel Portal (Das ist die einzelne Fläche von vorhin). Wählt dort die Eigenschaft FrontFace aus. (Bild links) Bei den anderen Sub-Models, also den Türflügeln müsst ihr dort None auswählen! Dies ist aber standardmäßig voreingestellt, da braucht ihr also nichts zu verändern.

Zwischenspeichern!

So, jetzt machen wir uns an den Bewegungsablauf unserer Tür: Wechselt in das Register **Animation.**

Ganz rechts unten unter *Time Configuration* gebt ihr bei *End* **60** ein, das sind die schon beschriebenen Animations-Keys. Bei *Animation Length* braucht ihr nichts verändern, die Zeit wird mittels Gam-File gesteuert! Hier könnt ihr lediglich testen, ob euch die Animation der Tür gefällt, wenn ihr wollt, dass zum Beispiel der Öffnungsvorgang 3 Sekunden dauern soll. Zum Start der Animation den *Play-Button* drücken.

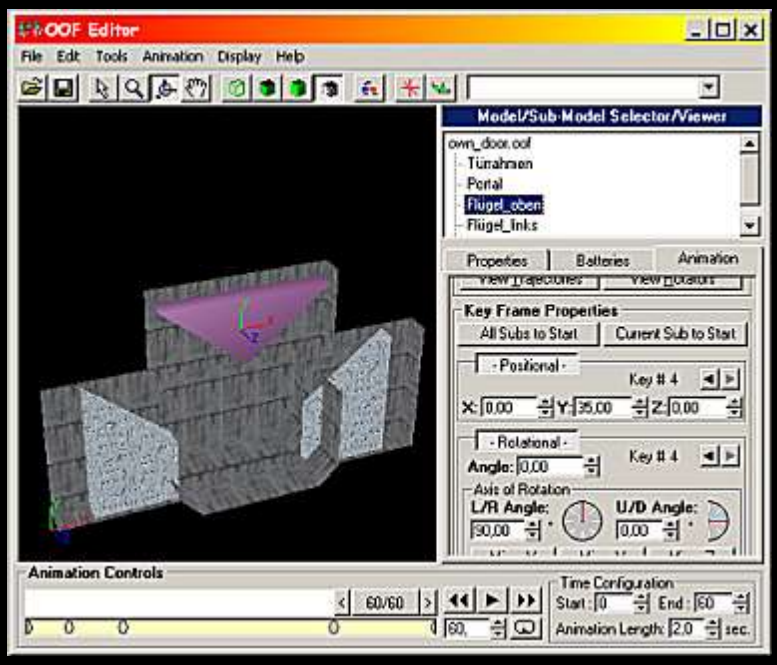
Dann aktiviert ihr der Reihe nach die einzelnen Sub-Models und klickt einach nur auf **Positional** und auf **Rotational** , ohne dort die Werte zu verändern! Passt auf, dass jeweils
Key # 0 (der erste Key) angezeigt wird. (Bild rechts)



Wichtig: Auch der **Türrahmen** und das **Portal** müssen einen **Key # 0** aktiviert haben. **Weitere Keys aber nicht**!
Einen Key setzt ihr einfach, indem ihr auf -*Positional*- und/oder -*Rotational*- klickt. Man erkennt gesetzte Keys in der unteren Leiste an den ovalen Markierungen.

Auch wichtig: Setzt bei Key-Frames aber immer beides, Positional und Rotational. Wenn man nicht beides setzt, animiert Descent 3 die Tür nicht, sondern zeigt sie nur entweder zu oder offen.

Nun könnt ihr rumexperimentieren, bis ihr einen euch angenehmen Bewegungsablauf für die Tür-Flügel gefunden habt. Die Sub-Models *Flügel_oben; Flügel_links und Flügel_rechts* sind mit den Koordinaten X, Y, Z oder mit den **Winkelangaben** verschiebbar. Ihr braucht nicht alle 60 weiteren Keys aktivieren, vielleicht nur 2 oder 3 oder mehr, je nachdem, wie raffiniert der Schließ-Vorgang eurer Tür sein soll. An welcher Stelle diese Keys innerhalb der Animation sind, ist ebenfalls egal. Es kann auch jedes einzelne Türflügel-Sub-Model verschiedene Keys besitzen. Jedoch muss bei diesen Sub-Models mindestens noch der **Key # 60** als Abschluss aktiviert sein. (Nur bei **diesen** Sub-Models!) Diese Keys sind zur Synchronisation der Einzelbewegungen sehr gut verwendbar.



Zu erwähnen ist noch, dass hier nur die Bewegung vom geschlossenen bis zum geöffneten Zustand zu definieren ist. Also bei Key # 60 muß der geöffnete Zustand der Tür zu sehen sein. Im Level wird dann einfach die Animation rückwärts abgespielt, wenn sich die Tür wieder schließt. Das brauchen wir hier bei unserer Animation nicht zu beachten!

Unsere Tür sieht im geöffneten Zustand im OOFEditor so aus:

Speichert das Objekt wieder, und schließt den OOFEditor.

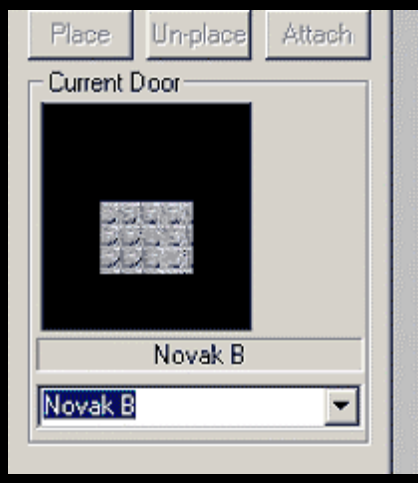
Jetzt öffnen wir mit dem **gamTool** die zuvor schon extrahierte Datei **Table.gam**
Dort drin suchen wir uns den Abschnitt **Door: NovakB** raus, kopieren diesen mit **Strg + C** in den Zwischenspeicher, klicken auf File -> New und fügen mit **Strg + V** den Abschnitt in den neuen Gam-File hinein.

(Natürlich kann man auch gleich einen neuen `gam`-File öffnen, **Alt + D** drücken und dann alle notwendigen Daten selber reinschreiben.)
Unter **Model File Name** den Pfad zu unserem Objekt `own_door.oof` angeben. (So wie's bei euch halt speziell gespeichert ist...)
Die Optionen *Opening Time, Closing Time* und *Open Time* könnt ihr entsprechend anpassen.
Alles andere unter Flags lasst so stehen! Wir wollen ja eben `generic.dll` für unsere neue Tür nutzen. **Auch den Door-Namen (NovakB) nicht verändern**!
Das `gam`-File dann speichern, **und beachten, dass diese Datei denselben Namen wie die spätere MN3-Datei besitzt.**
Konkret bei unserem Beispiel: `own_door.gam` ==> `own_door.mn3`

Geschafft...eigentlich war's das schon!

Wir öffnen jetzt wieder den D3-Editor und laden unter Data -> Tablefile Manager die Datei `own_door.gam` hinein.

Wenn wir jetzt im **Door-Menü** unter *Novak B* nachsehen, dann sieht man, dass das Original-Door durch die neue Tür ersetzt wurde ☺ (Bild rechts)

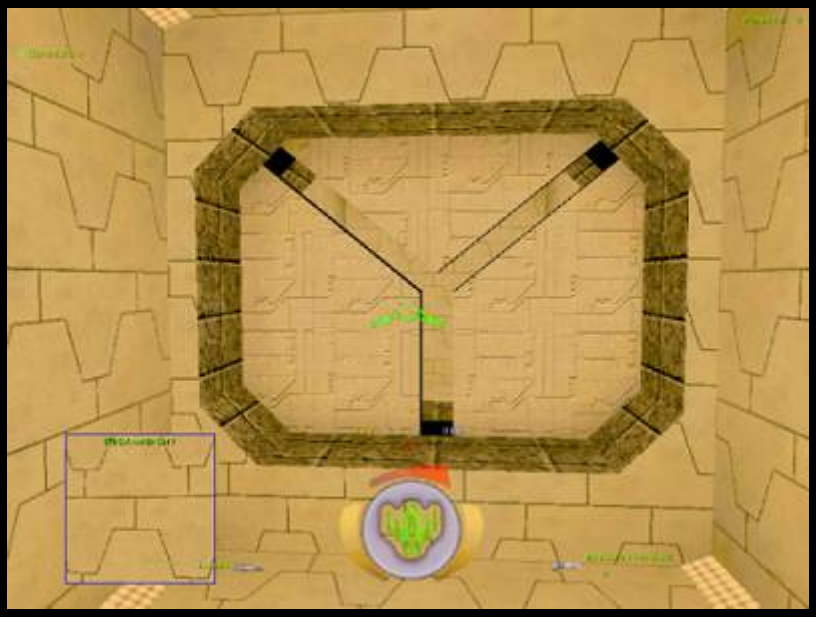


Wundert euch nicht über die Darstellung, in der Door-Vorschau im Editor werden nur die Tür-Objekte (also die Türflügel) dargestellt.

Fügt nun die selbsterstellte Tür in euer Level ein, und texturiert den Türrahmen nochmal nach eurem Geschmack, falls es notwendig ist... Wenn ihr zum Schluss die MN3-Datei packt, dann gebt unter **Additional Files** die Datein `own_door.gam` und `own_door.oof` mit an.

Viel Spaß beim Experimentieren! Der Phantasie sind eigentlich kaum Grenzen gesetzt, was für heftige Doors ihr in die Levels einbauen könnt! Zum Beispiel könnten sich ja erst noch diverse Verriegelungen lösen, bevor sich eine Tür öffnet. Oder ihr baut euch *themenspezifische* Doors...

Wenn ihr Fragen oder Verbesserungen zu dieser Anleitung habt, schreibt an **descent@toxxeon.de**



Update:

Mittlerweile ist es nicht mehr nötig, vorhandene Türen gegen die eigenen auszutauschen. Neuere D3Edit-Versionen beherrschen es inzwischen, daß man seine Eigenkreationen ganz normal als `.gam`/`.oof`/`.mn3`-Paket einbindet und in der `.gam` mit eigenen Namen versehen kann. Diese Türen erscheinen dann – ähnlich wie bei den Texturen – ganz am Ende der Liste.

Ebenso ist es nicht mehr möglich das Door-Objekt einfach zu löschen, siehe rechts

Ich verweise hier auch noch auf die Section ‚Quicktest mit D3Edit einrichten' von WillyP, das nimmt einem viel Arbeit ab.

Achtung:

D3Edit kann maximal 60 Türen handhaben, was bedeutet daß Du maximal 11 zusätzliche, eigene Türen verwenden kannst. Einträge, die weiter unten in der Gamfile stehen, werden in der Door-Palette dann nicht mehr angezeigt! Man könnte dann wieder beginnen, originale Türen zu ersetzen.

073 - Sound für die eigene Tür

Ragil Ral

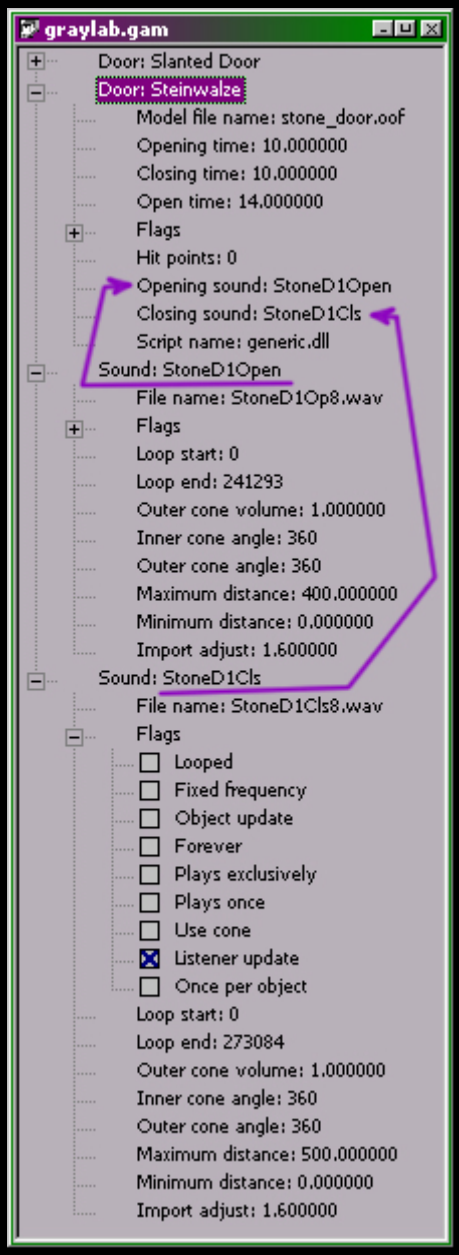
Nun habt Ihr also eine Tür. Das Blöde ist jetzt, daß die nun entweder garkeinen oder einen der Standard-Türensounds hat. Um Eurer Tür nun noch die entsprechende Geräuschkulisse zu verleihen müßt Ihr wieder Arbeit investieren.

SoundFX erstellen

Besorgt Euch aus dem Netz Geräusche, es gibt einige Seiten die solche als Gratis-DL anbieten; in den meisten Fällen wird von Euch verlangt, die Autoren zu erwähnen, was ich nur fair finde. Ihr werdet allerdings kaum welche finden, die Ihr sofort verwenden könnt, deshalb braucht ihr ein Programm mit dem man Sounds zusammenschneiden und mischen kann. Wenn Ihr mit solchen Programmen keine Erfahrung habt, müßt Ihr das jetzt halt lernen.
Analysiert nun die Animationssequenz Eurer Tür nach Zeit, dh. wenn bei Frame 69 was passiert, müßt Ihr wissen wieviel Zeit bis dahin vergangen ist; denn genau danach müßt Ihr die Geräuschkulisse anpassen. Macht Euch am besten eine kurze Liste, wo Ihr festhaltet bei welchem Frame/welcher Zeit was passiert. Die Sounds zu erstellen ist eigentlich die meiste Arbeit, auch recht zeitaufwendig.
Die fertigen Sounds sollten dann das Format 16 bit, 22 kHz mono `.wav` haben.

SoundFX einbinden

Das ist schnell erledigt: Ihr müßt in der `.gam` nun zwei zusätzliche Einträge setzen. Entweder kopiert Ihr zwei ‚Sound'-Einträge aus der D3-`.gam`, oder macht Edit->New->Sound (**Alt-S**). Tragt die Werte entsprechend ein (Screenie). Der Tür verpaßt nun bei `Opening Sound` und `Closing Sound` die Namen Eurer Soundeinträge als Wert.



Die Einträge für Sound sind wie folgt:

File name - Name der `.wav`-Datei.
Loop start - Bei welchem Sample beginnnt der Loop
Loop end - Bei welchem Sample endet der Loop (aufrunden)
(Länge der Soundatei in Samples = Länge in Sekunden mal Samplingrate, wobei 22 kHz mit 22050 zu rechnen ist)
Outer cone volume - ...
Inner cone angle - ...
Outer cone angle - ...
Maximum Distance - Entfernung, ab der man nichts mehr hören kann.
Minimum Distance - Distanz, unter der nichts mehr zu hören ist.
Import adjust - Um welchen Faktor die Lautstärke Eures Sounds ingame angepaßt wird.

Bei den Flags setzt noch Listener Update, sodaß sich die Lautstärke für den in Bewegung stehenden Spieler mit der Entfernung auch verändert.

Genaue Erklärung der Flags findet sich in der `.gam`-Spezifikation am Ende des Dokuments.

Packt dann mit dem `.oof` und der `.gam` zusätzlich auch noch die `.wav`-Dateien mit in die Missions-Hog rein.

Das wars auch schon

File: 074-eigene_obj1.zip

074 - Eigene Objekte 1

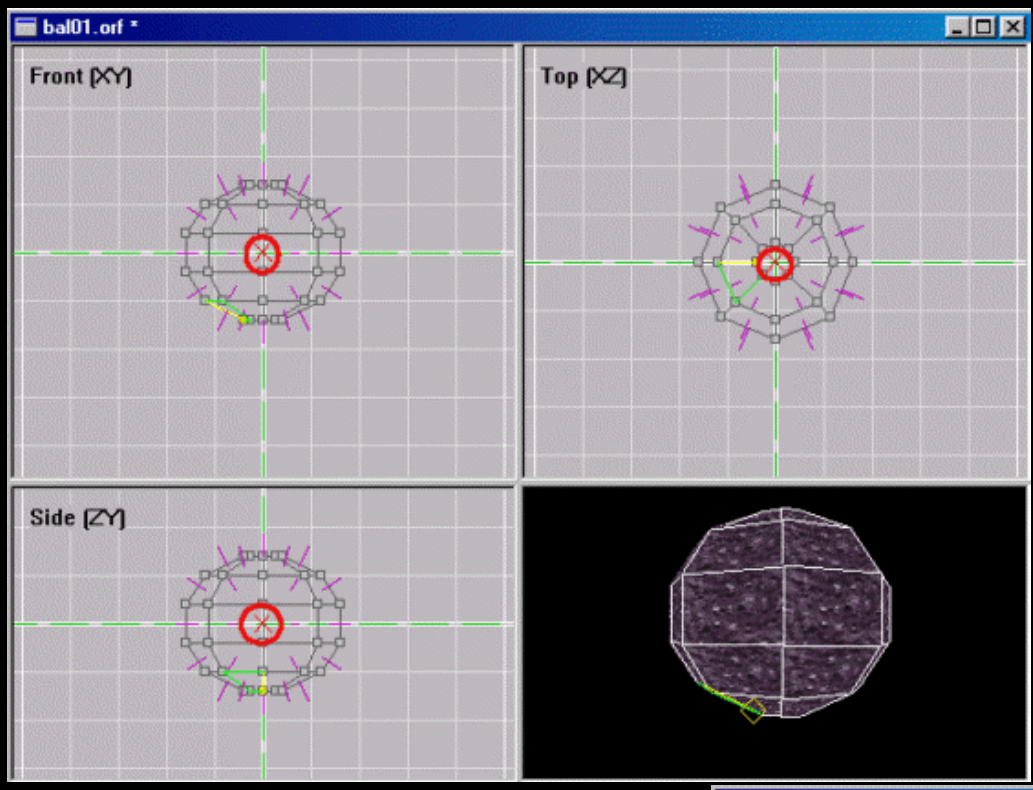
Fischlein - Inspiriert von Papacat

Die Dateien zu dem Thema sind in der zip-File.

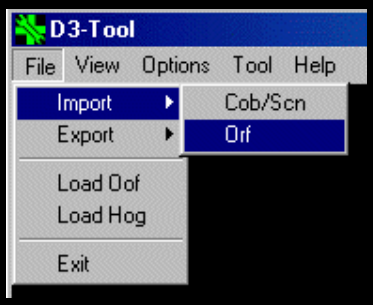
Eigene Objekte zu erstellen ist nicht so schwer wie Ihr annehmt. Ihr braucht dazu D3Edit, das Gamtool und D3Tool.

Hinweis: Wenn Ihr ein eigenes Objekt erstellt, muss es immer absolut auf dem Raster Nullpunkt ausgerichtet sein ! (Bild unten)

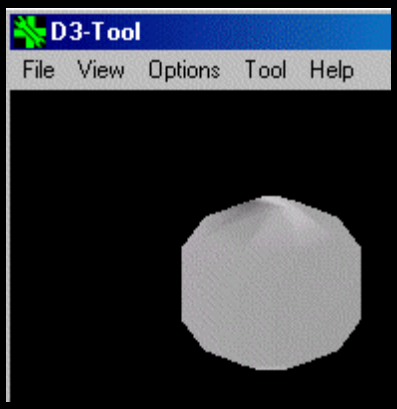
Ich habe den Raum unter "`ball01.orf`" gespeichert, Ihr könnt es nennen wie Ihr wollt. Schreibt den Dateinamen klein! Ich denke das es sonst zu Probleme bei einen Linux Server kommen könnte (Vermutung !).



Öffnet nun das D3Tool. Wählt die Import Funktion

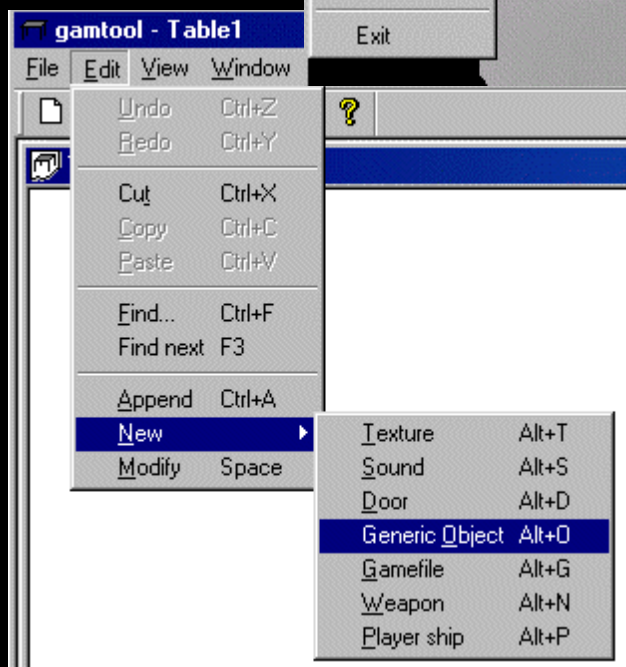


und "`ball01.orf`" aus. Danach müsste es wie im Bild unten aussehen.

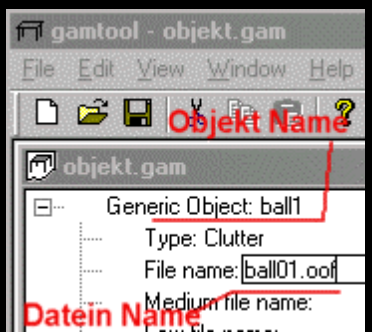


Anschliessend exportiert Ihr den Raum als `*.oof` (Outrage Objekt File, rechtes kleines Bild).

Öffnet das GamTool und wählt Edit/New/Generic Objekt. (rechts, größeres Bild) Als erstes tragt den Namen mit Pfadangabe (**maximal 34 Zeichen sind zulässig !!!**) ein. Den Pfad braucht Ihr nur angeben, wenn Eure eigenen Objekte nicht im D3-Pfad abgelegt sind!!!

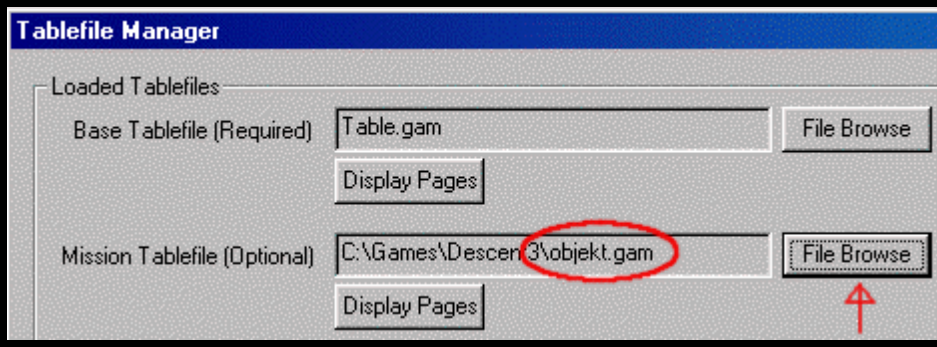


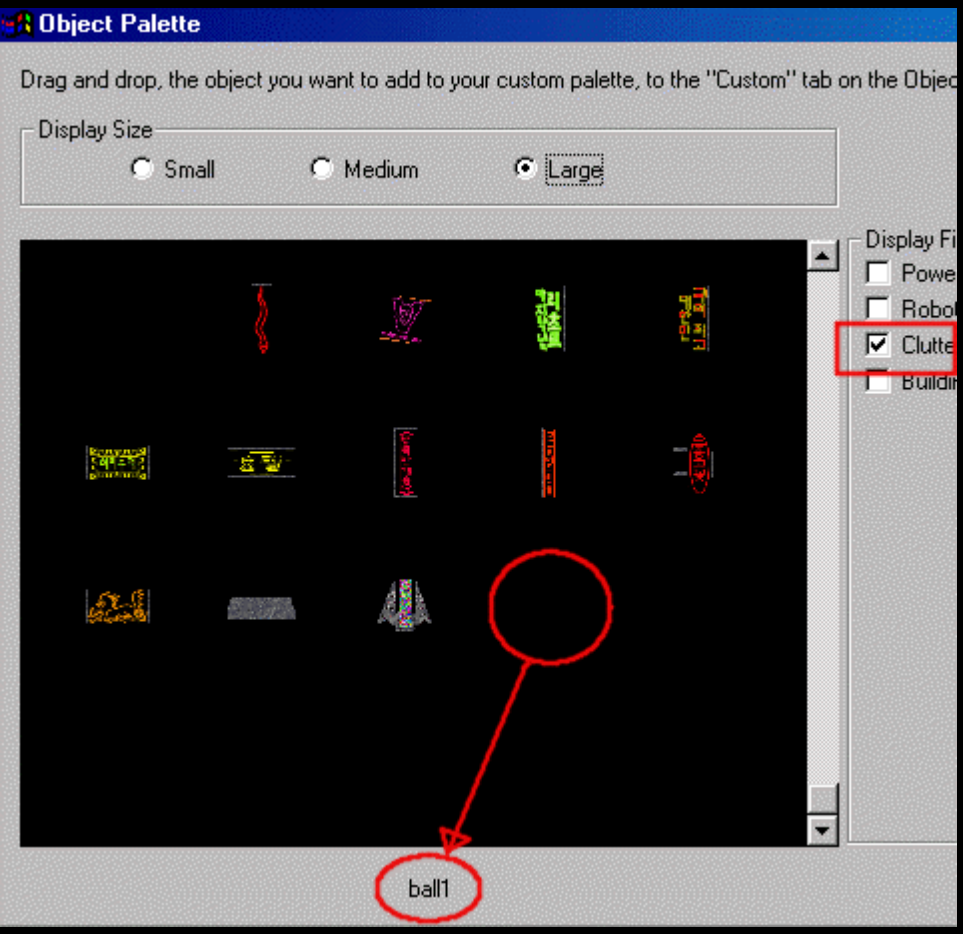
Wir haben nun ein Objekt Namens ball1 erstellt. Ich habe die Tabelle als `objekt.gam` gespeichert (klein geschrieben).



Öffnet nun D3Edit und wählt Data im Menü, Bild links. Nun klickt Ihr auf **File Browse** und sucht Euer `*.gam` File. Jetzt könnt Ihr noch auf Display Pages klicken und könnt sehen,was alles in dem `*.gam` File definiert ist (Bild unten).

Ok, erstellt Euch nun ein Level, oder nehmt ein Bestehendes und ruft das Objektmenü auf. Klickt auf die Schaltfläche **Available Objects...**, dann Filter Typ: **Clutter** und Scrollt bis ganz nach unten.





Ich weiss nicht genau warum D3Edit eigene Objekte nicht grafisch darstellt, aber die Objekte oder Texturen, die Ihr selbst entworfen habt, findet Ihr immer am Ende. Fügt nun das Objekt in Eure Level ein und speichert den Level.

Update:
Neuere Versionen beherrschen es inzwischen, eigene Objekte anzuzeigen. Lest Euch dazu auch die Sektionen Custom-Objekte in D3Edit sichtbar haben im Anhang und 124 - Customs, nochmal durch.

Achtet darauf, das der Level den selben Namen wie Euer *.gam File hat !!!!

Startet den MN3 Packer und fügt das Objekt `*.oof` und das `*.gam` File ein.

Auf Level Namen achten !!!

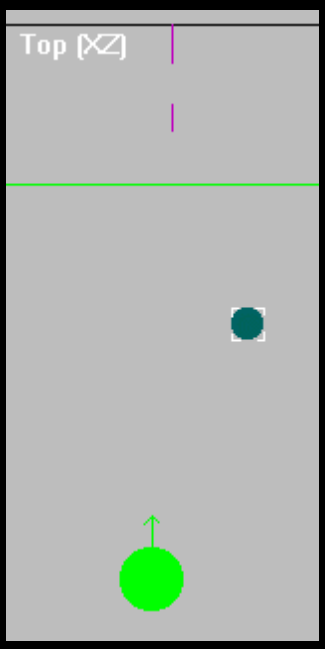
File: 075-eigene_obj2.zip

075 - Eigene Objekte 2

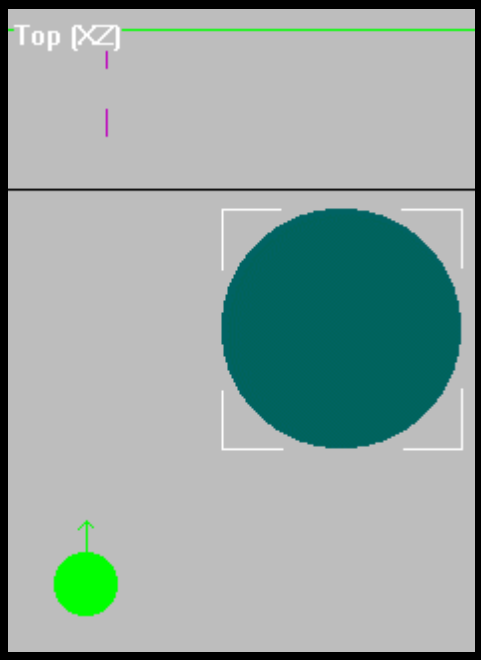
Fischlein, inspiriert von Papacat

Die Bilder auf dieser Seite sind teilweise von Papacat!

Die Dateien dazu, wie immer, in der zip-File.
Im vorherigen Thema haben wir ein Objekt erstellt und dazu auch das Gam File. In diesem Thema werden wir einige Eigenschaften für das Objekt festlegen, was auch wieder im Gam File gemacht wird. Aber zunächst erst einmal eine nicht so wichtige Eigenschaft, die aber durchaus nützlich ist, **SIZE**.



Im linken Bild könnt Ihr sehen, wie es aussieht, wenn Ihr diesen Wert auf Standard lasst. Im rechten Bild, nachdem ich SIZE eingetragen habe. Den Ball, den ich gebaut habe, hat einen Durchmesser von 30 Einheiten, daraus folgt ein Radius von 15 Einheiten. Ich muss also unter SIZE eine 15 eintragen. (nicht den Wert des Durchmessers)



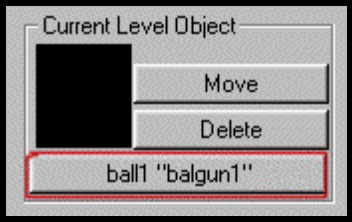
Im folgenden Bild könnt Ihr sehen, welche Einstellungen wir vornehmen:

- Generic Object: ball1
  - Type: Clutter
  - File name: ball01.oof
  - Medium file name:
  - Low file name:
  - Impact size: 0.000000
  - Impact time: 0.000000
  - Damage: 0.000000
  - Score: 500
  - Unused string (obsolete):
  - Module name:
  - Script name override:
  - [ ] Has description (obsolete)
  - Icon name (obsolete):
  - Medium LOD distance: 30.000000
  - Low LOD distance: 60.000000
  - Physics chunk
    - Mass: 1.000000
    - Drag: 1.000000
    - Full thrust: 0.000000
    - Flags
      - [ ] Roll when turning
      - [ ] Leveling
      - [x] Bounce
      - [ ] Wiggle
      - [ ] Stick
      - [ ] Persistent
      - [ ] Uses thrust
      - [ ] Gravity
      - [ ] Ignore own conc forces
      - [ ] Wind
      - [ ] Uses parent velocity
      - [ ] Fixed velocity
      - [x] Fixed rot velocity
      - [ ] No collide parent
      - [ ] Hits siblings
      - [ ] Reverse gravity
      - [ ] No collide
      - [ ] No robot collisions
      - [ ] Point collide walls
      - [ ] Homing
      - [ ] Guided
      - [ ] Ignore concussive forces
      - [ ] Destination pos
      - [x] Lock X
      - [x] Lock Y
      - [x] Lock Z
      - [ ] Lock P
      - [ ] Lock H
      - [ ] Lock B
      - [ ] Never use big sphere
      - [ ] No same collisions
      - [ ] No door collisions
    - Rotational drag: 1.000000
    - Full rotational thrust: 0.000000
    - Number of Bounces: -1
    - Initial velocity: 0.000000
    - Initial rotational velocity
      - x: 0.000000
      - y: 15000.000000
      - z: 0.000000
    - Wiggle amplitude: 0.000000
    - Wiggles per second: 0.000000
    - Coefficient restitution: 1.000000
    - Hit death angle: -1.000000
    - Max turn-roll rate: 0.000000
    - Turn-roll ratio: 0.000000
  - Size: 15.000000
  - Lighting chunk
  - Hit points: 100
  - Flags
    - [ ] Uses AI
    - [x] Uses physics
    - [x] Destroyable
    - [ ] Inventory selectable
    - [ ] Inventory nonuseable
    - [ ] Inventory type mission
    - [ ] No inventory remove
    - [ ] Inventory vis when used
    - [ ] AI scripted death
    - [ ] Do ceiling check
    - [ ] Ignore forcefields and glass
    - [ ] No diff scale damage
    - [ ] No diff scale move
    - [ ] Ambient object
  - AI settings

Score: Hier wird eingestellt, wie viele Punkte der Spieler erhält, wenn man es zerstört.
Bounce: Gibt an, das das Objekt, wenn es getroffen wird, von den Wänden und anderen Objekten abprallt.
Fixed rot velocity: Dieses Häkchen steht dafür, wenn ein Spieler auf das Objekt schiesst, beginnt es sich zu drehen und das mit der Geschwindigkeit, die man unter Initial rotational velocity angibt.
Lock X, Lock Y, Lock Z: Wird ein Häkchen gesetzt um zu vermeiden, das bei einem Treffer sich das Objekt verschiebt oder abprallt.
Number of Bounces: Setzt diesen Wert auf -1, wenn Ihr einen höheren Wert wählt, könnte es sein, das das Objekt aus dem Raum fliegt, da es dann sehr schnell von der Wand abprallt. Es würde förmlich aus der Level geschleudert!!!
Initial rotational velocity: Wählt hier die Achse, um das sich das Objekt drehen soll und die Geschwindigkeit. Ihr müsst hohe Zahlen verwenden!!!
Size: Siehe oben.
Hit Points: Gibt an, wie oft das Objekt getroffen werden muss, bis man es zerstört hat.
Uses Physics: Das die Optionen des Physics chunk verwendet werden.
Destroyable: Ob das Objekt zerstört werden kann oder nicht.

Nehmt die ganzen Einstellungen vor und speichert das GAM File.

Öffnet nun D3Edit und ladet das Gamfile (Data/Tablefile Manager). Jetzt öffnet Ihr Euer Level, ruft die Objekt Bar auf (O in der Menüleiste) und wechselt in den Objektmodus (**STRG+G**). Nun macht Euer Objekt zum Aktuellen (bei mir ist das Ball1) und klickt auf die Schaltfläche, die im rechten Bild rot umrandet ist. Wir müssen unseren Objekt einen Namen geben, um später mit dem Dallas Script darauf zugreifen zu können.
Ich habe es "balgun1" genannt. Wechselt jetzt in die World View Ansicht und dann ins Menü File/Dallas Graphical Script Editor (Ihr werdet nun zwei Hinweis-Boxen bekommen, weil Ihr noch kein Script für Eure Level erstellt habt, einfach auf **OK** klicken.).
Wenn Ihr noch kein Dallas Script Editor installiert habt, dann lest Euch das Thema Vorgriff: DALLAS zum Laufen kriegen durch!



Script 000: Start Level
- Owner: Level
- Event: Level Start
- If the following condition is met:
  - ? ALWAYS
- Then perform the following actions:
  - Set movement type for balgun1 to None
    - Object: balgun1
    - MovementType: None

Script 001: Gunfire and Rotation
- Owner: balgun1 (Object)
- Event: Collided (with IT)
- If the following condition is met:
  - ? (IT is a player weapon) is TRUE
    - Bool: IT is a player weapon
      - Object: IT
    - Operation: is TRUE
- Then perform the following actions:
  - Fire weapon "laser" from gun number -1 of object balgun1
    - WeaponName: "laser"
    - GunNum: -1
    - Object: balgun1
  - Set movement type for balgun1 to Physics
    - Object: balgun1
    - MovementType: Physics

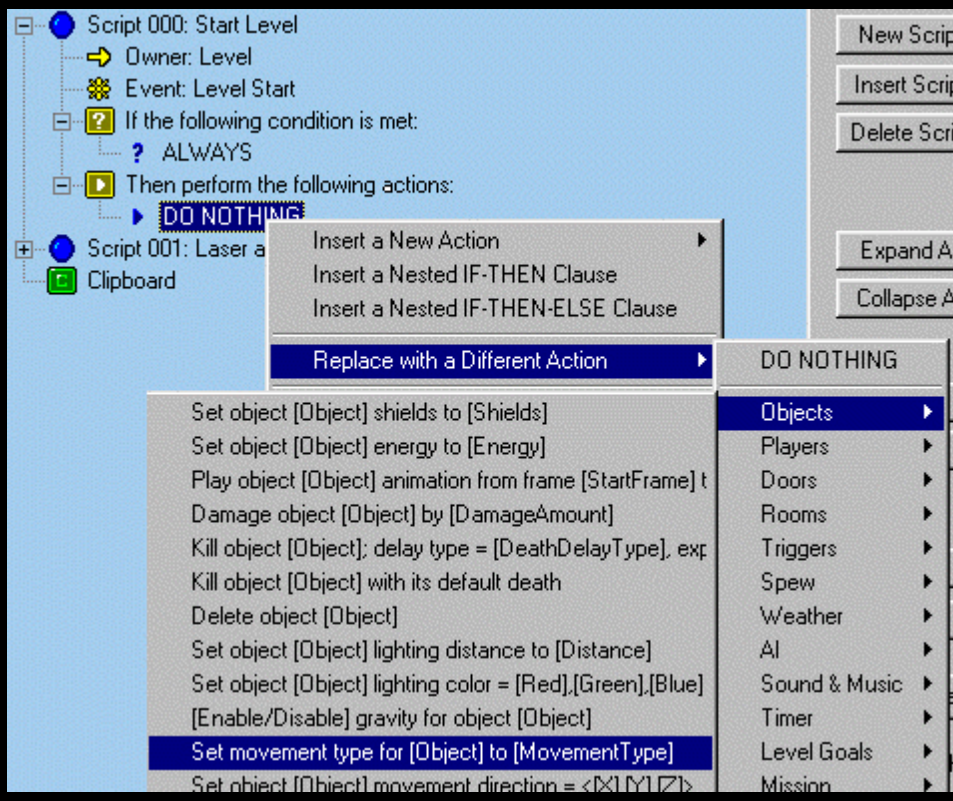
Clipboard

Für all die von Euch, die sich schon im Dallas Script auskennen, können alle Einstellungen aus dem linken Bild übernehmen.

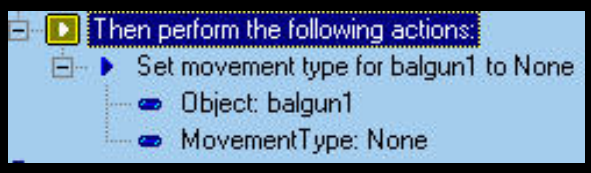
Erklärung zum Bild:

Script 000: Start Level
Wenn Ihr den Dallas Graphical Script Editor geöffnet habt, seht Ihr ein leeres Fenster. Klickt auf die Schaltfläche **New** bei Scripts und gebt als Bezeichnung Start Level ein. Als nächstes muss bei Owner ein Wert eingetragen werden, klickt mit der rechten Maustaste auf Owner und wählt dann aus dem Untermenü Level. Nun muss das Event angegeben werden, wieder mit der rechten Maustaste auf Event klicken und aus den Untermenü "Level Start" wählen.

Zum nächsten, Then perform the following actions und schaut Euch im Bild rechts an, wie das gemacht wird.



Nachdem Ihr das im vorigen Bild gemacht habt, müsst Ihr noch das Objekt angeben. Klickt mit der rechten Maustaste auf Objekt und dann Clutter und darin den Namen Eures Objekts.



Script 001: Laser and Rotation

Das geht jetzt genauso wie beim ersten Script. Owner stellt Ihr auf Euren Objektnamen (bei mir `balgun1`). "Event" stellt Ihr auf "Collided(with IT)". Das könnt Ihr aber alles aus dem Bild der vorigen Seite, links oben, entnehmen. Unter "If the following condition is met" klickt Ihr "Always" an und dann die rechte Maustaste, in diesen Untermenü wählt Ihr dann "Replace with a New Condition", darunter Binary Statemend with Query -> Objects wählen und dann [Object] is a player weapon.

Weiter zu "Then perform the following actions". Zum ersten muss der Laser rein, das geht wie folgt. Klickt zu nächst "DO NOTHING" an und dann mit der rechten Maustaste darauf. Wählt "Replace with a Different Action -> Objects -> Fire weapon [WeaponName] from gun number [GunNum] of object [Object]"

Jetzt müssen wir noch unseren Ball zum drehen bringen, sobald auf ihn geschossen wird. Klickt mit der rechten Maustaste "Then perform the following actions" und wählt "Add a New Action/Objects/Set movement type for [Object] to [MovementType]", ist das selbe, was schon im zweiten Bild der vorigen Seite zu sehen ist, aber unter "MovementType" stellt auf "Physics" ein. Klickt jetzt auf die Schaltfläche **Save**.

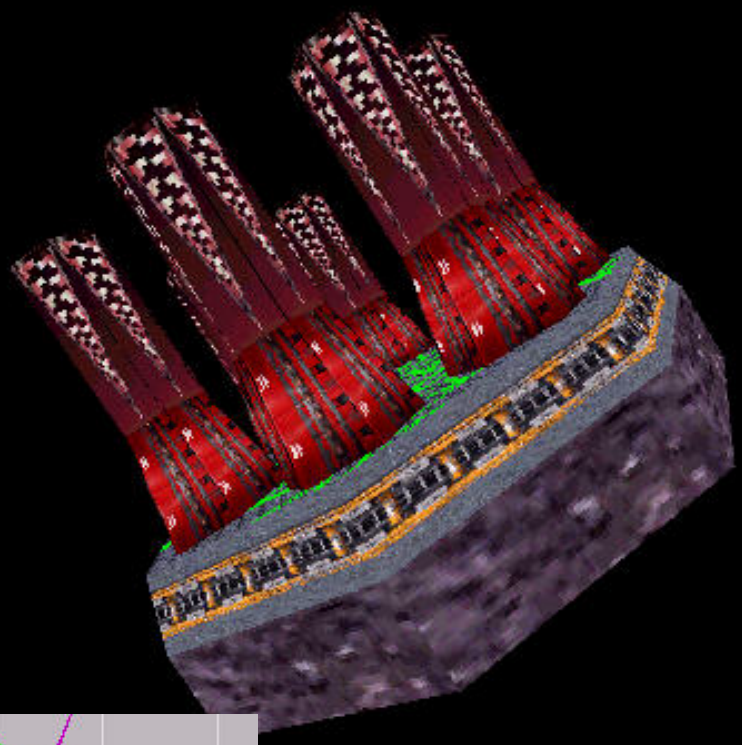
Speichert Eure Level und fügt im MN3 - Packer alle Dateien hinzu. (bei mir sind das: `objekt.gam`, `ball01.orf`, `objekt.dll` und `objekt.msg` ,die letzten beiden Dateien habe ich in `C:\Games\Descent3\Dallasscript\osiris` liegen, kommt darauf an ob es bei Euch der selbe Pfad ist)

076 - Eigene Objekte 3

Fischlein, inspiriert von Papacat

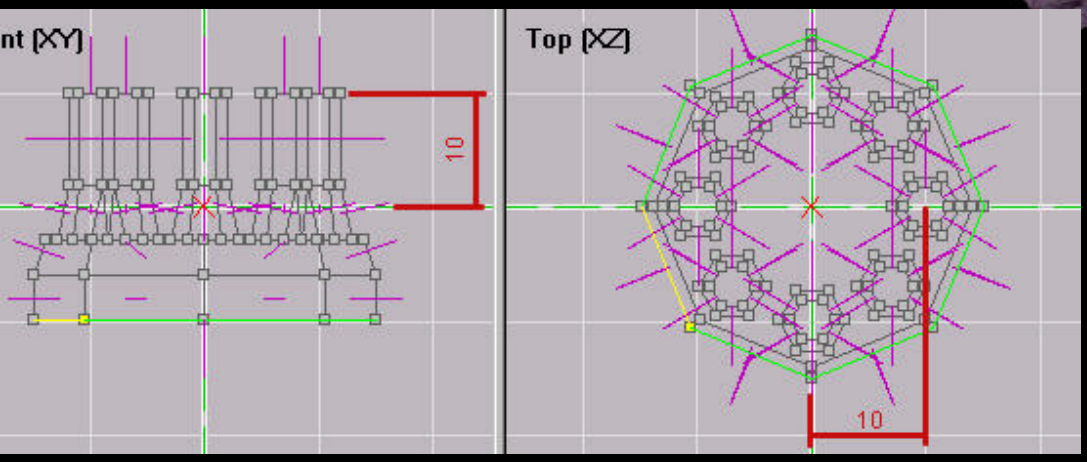
Nachdem Ihr nun wisst, wie man Objekte erstellt und in den Level einbindet, fehlen eigentlich nur noch ein paar schöne Effekte. Nun denn, ich werde hier versuchen Euch zu zeigen, wie man an einem Objekt einen "Gun Point" hinzufügt, positioniert und auch zum "Leben" erweckt. Für dieses Thema braucht Ihr das **D3Tool** (V1.4).

Ich habe mir erst einmal ein Objekt erstellt, das im Bild rechts zu sehen ist.



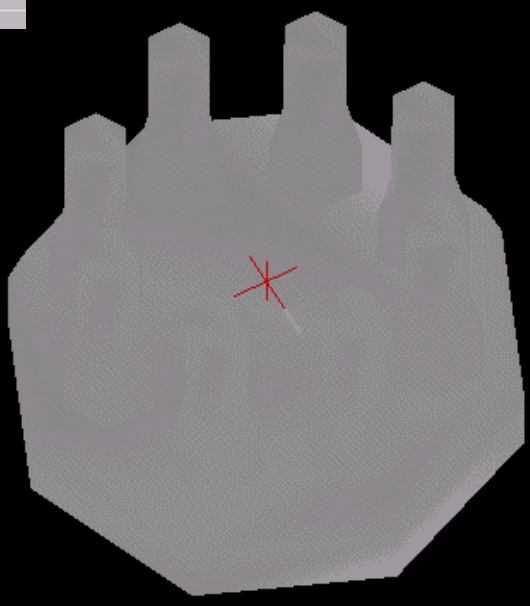
Wie Ihr das erstellen könnt, ist in den Themen **Eigene Objekte1** und **Eigene Objekte2** erklärt.

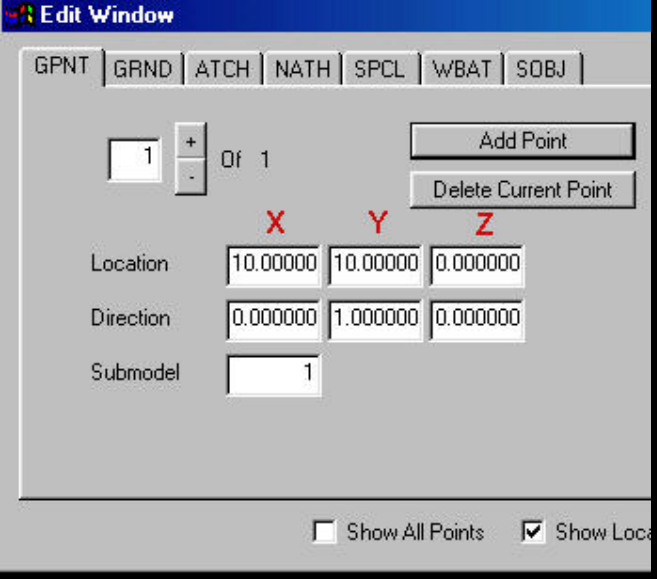
Um zu wissen, wo wir unsere Gun Points im D3Tool positionieren müssen, öffnen wir D3Edit und den Raum, den wir für das Objekt erstellt haben. Wir brauchen zwei Werte, welche seht Ihr im Bild unten.



Öffnet nun das D3Tool und wählt File -> Import... ORF. Wählt nun Euren ORF aus.

Geht nun in das Menü Tool -> Edit Window, die gewählte Tab ist GPNT. Klickt hier auf die Schaltfläche **Add Point**, setzt ein Häkchen, unten bei Show All Points. Jetzt seht Ihr ein rotes Kreuz (Bild rechts) im Zentrum Eures Objektes, an diesem Kreuz ist an einer Seite ein hellerer Strich zu sehen, das ist die Richtung in die geschossen wird.





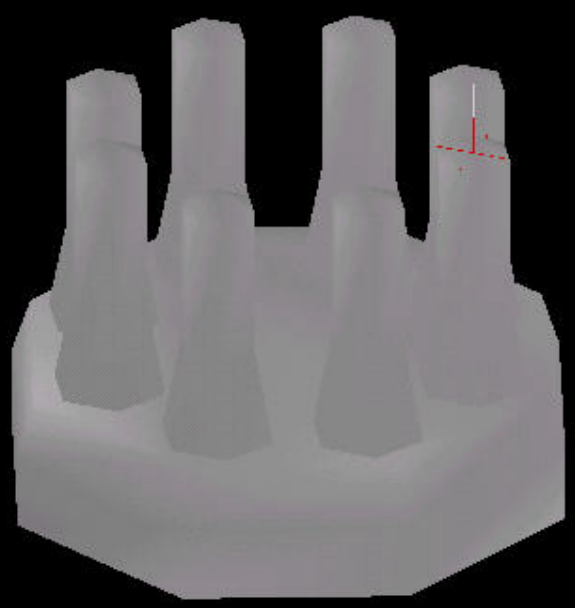
Im Bild links seht Ihr was einzustellen ist, um den Gun Point zu positionieren (X= +10, Y=+10, Z=0).

Das + braucht man nicht mitzuschreiben, nur das - (minus).

Als Direction habe ich die Y Achse gewählt. (einfach eine 1 rein, 0 ist aus 1 ist ein)

Hier einmal die Übersicht über die Positionen meiner Gun Points, ich brauche 8:

Z= 0
Z= 0
Z= +10
Z= -10
Z= +7
Z= -7
Z= -7
Z= +7



Anmerkung: Achtet darauf, für welchen Point Ihr die Position einstellt. (1 of 8 oder 5 of 8) Des weiteren ist mir aufgefallen, das man die 3D Ansicht nicht bewegen sollte, denn bei mir haben die Gun Points die Position verändert, aber nur als ich die ersten zwei gesetzt habe. Ich würde Euch empfehlen, alle Points zu setzten, das Objekt zu speichern, danach noch einmal öffnen und nachsehen, ob alles richtig ist.

Zum Speichern, geht in das Menü File ->Export... Oof. Erstellt jetzt das Gam File mit folgenden Einstellungen:

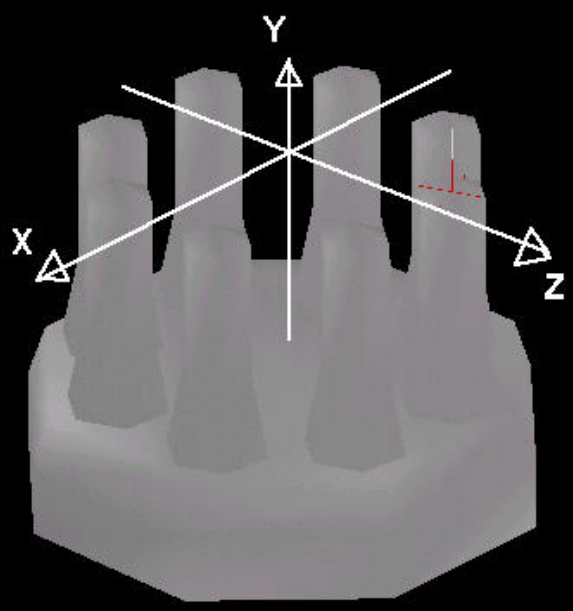
Physics chunk/Initial Rotation Velocity: Y=90000
Physics chunk/Flags/Fixed rot Velocity - ankreuzen
Physics chunk/Flags/Lock: X, Y und Z

Speichert das Gamfile unter dem selben Namen wie Eure Level heisst!

Öffnet nun D3Edit, geht in das Menü Data -> Tablefile Manager und wählt Euer Gamfile aus.

Anmerkung: Um Eure eigenen Objekte auch in der 3D Vorschau sehn zu können, erstellt ein fertiges MN3 File, mit allen was rein gehört, auch *.GAM und *.OOF. Wenn Ihr nun D3Edit öffnet, wechselt in das Menü Data -> Configure A Mission Hog. Im folgenden Dialogfeld klickt auf **Change** und wählt Eure MN3 aus und klickt anschliessend auf **OK** und im nächsten noch mal **OK**. Jetzt wieder Data -> Tablefile Manager und klickt auf **Hog Browse** bei Mission Tablefile.

Update:
Bei neueren D3Edits ist es ausreichend, das `.oof` dort liegen zu haben wohin in der `.gam` verwiesen wird. Beachtet hier die 34-Zeichen-Grenze für Pfad und Dateiname!



Öffnet nun Eure Level *.D3L und schon in der Vorschau (Objekte einfügen) seht Ihr es jetzt. Fügt es in Eure Level ein und schaut es Euch an. Das ist hilfreich, wenn man Objekte ausrichten muss. Nun müsst Ihr Euer Objekt noch benennen. Drückt die Tasten **STRG+G** um in den Objektmodus zu wechseln, nun klickt Ihr in der Raumansicht Euer Objekt an, so das Ihr oben in der Objektbar Euer Objekt seht. Unter dieser Vorschau befindet sich ein Button, auf dem der Name des Objektes zu sehen ist. Klickt auf diesen und tragt in dem folgenden Dialogfeld einen Namen Eurer Wahl ein. Ich habe das Objekt "waffe" genant. Das muss man machen, um über Script auf das Objekt zugreifen zukönnen.

Skript zu diesem Thema

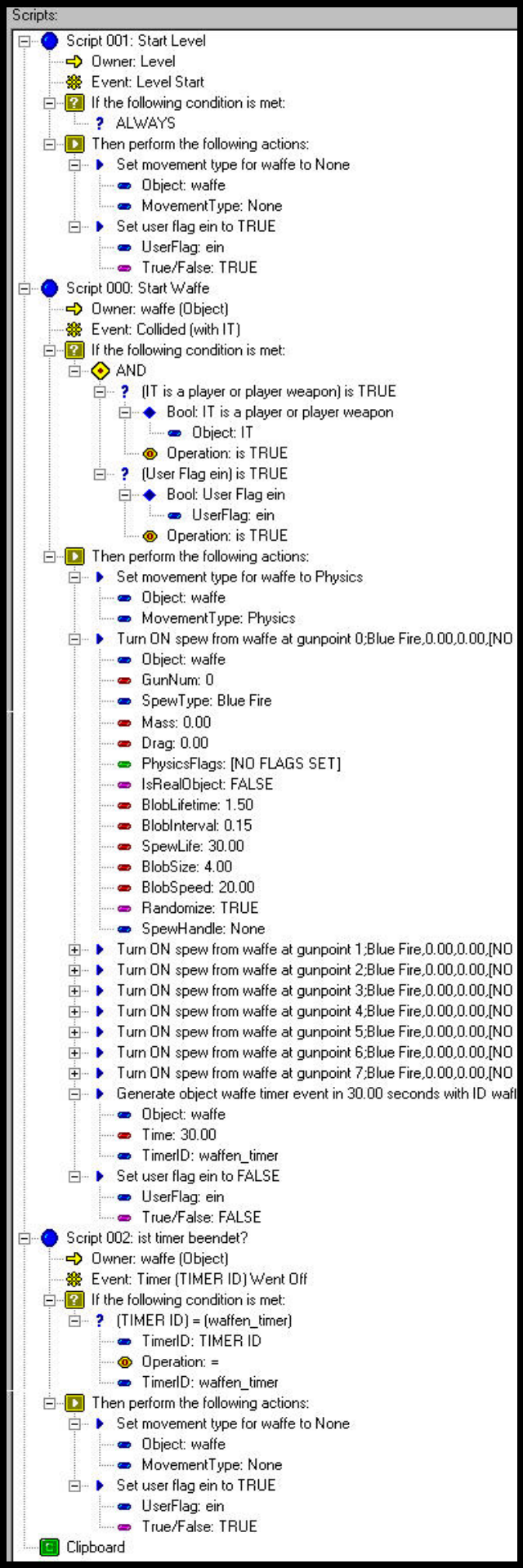
Wie Ihr den "**Dallas Graphical Script Editor**" einrichtet, wird unter Der Dallas Graphical Script Editor erklärt.

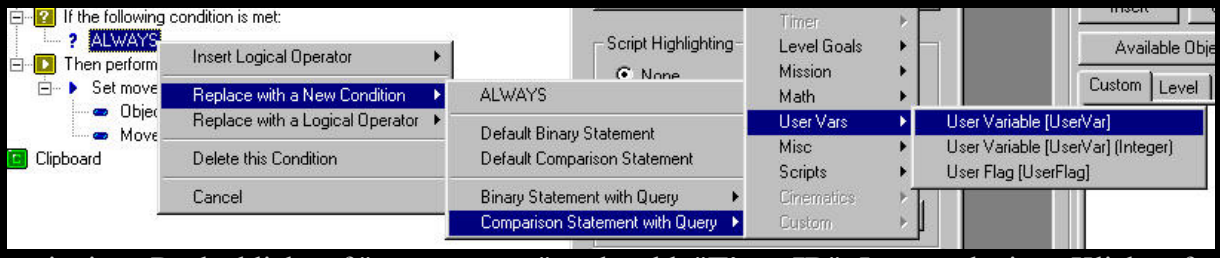
Wechselt in die World View Ansicht und dann in das Menü File -> Dallas Graphical Script Editor. Klickt nun auf **New Script** und benennt es im folgenden Dialogfeld mit **Start Level**. Den Rest könnt Ihr aus dem Script im rechten Bild entnehmen.

Die User Flag bekommt Ihr so: Klickt **User Types Workshop**, wählt bei Select the User Type to edit: UserFlag, dann klickt **Add New Value** und vergebt im darauffolgenden Dialog einen Namen (hier „ein“).

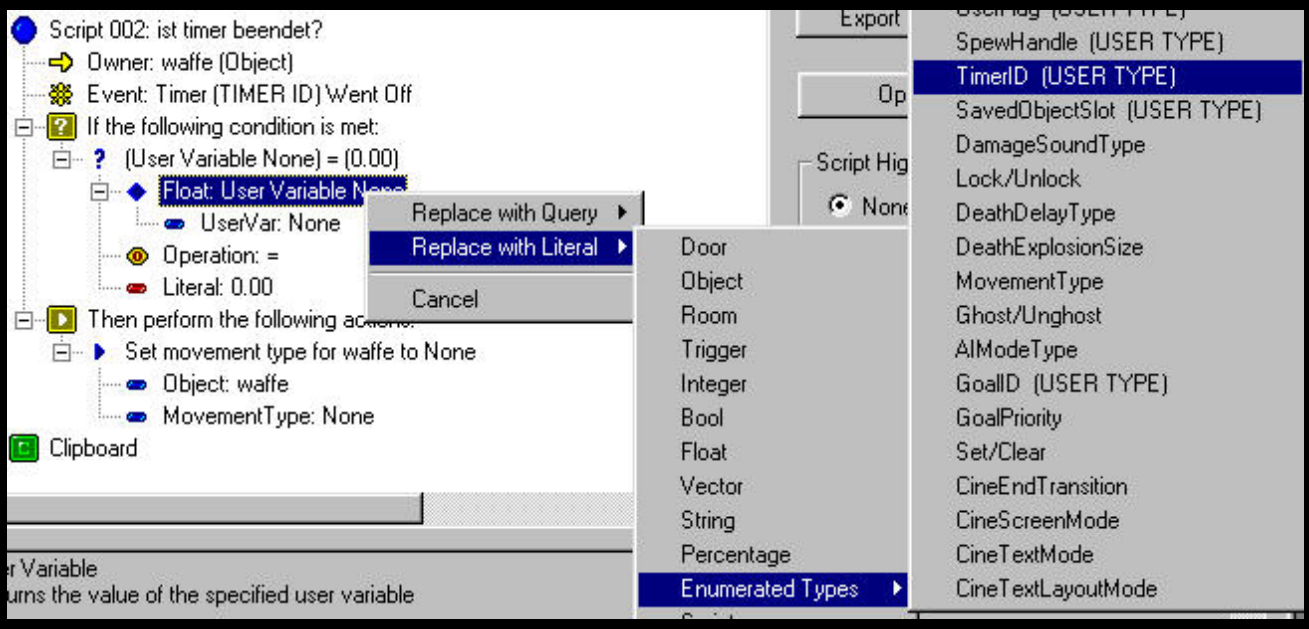
Wenn Ihr das getan habt, klickt wieder auf **New Script** und nennt es **Start Waffe**. Übernehmt wieder die Einstellungen aus dem rechten Bild.

Erklärung: Unter Then perform the following actions steht alles, was ausgeführt werden soll. "Set movement type for waffe to Physics" sagt, das die Parameter, die im Gamfile unter Physics stehen, gültig sind. Das "Turn ON spew...", werde ich in einem extra Thema erklären, da es zu umfangreich ist.
"Generate objekt...", startet einen Timer für 30 Sekunden und gibt diesen die ID "**waffen_timer**". Diese ID erstellt Ihr wie folgt: Im Script Editor gibt es rechts eine Schaltfläche auf der steht, "**Open the User Types Workshop**" klickt diese an und wählt unter "Select the User Type to Edit", "TimerID", jetzt klickt auf "**Add New Value**" und wählt einen beliebigen Namen aus, der sich auf den Timer bezieht. Ich habe ihn "**waffen_timer**" genannt.
Jetzt zm Script "ist timer beendet?".
"If the following condition is met:" Um das in den umfangreichen Funktionen zu finden, habe ich die folgenden beiden Bilder vorbereitet:



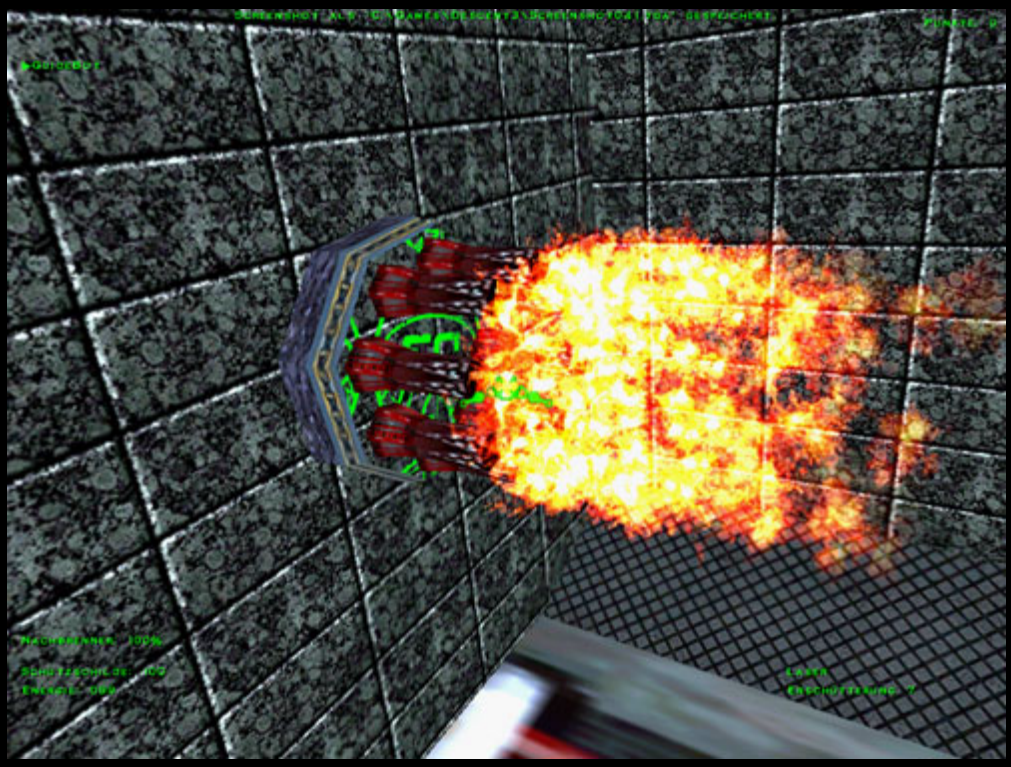


Nun mit einen Rechtsklick auf "TimerID:None" und wählt "**TimerID**". Jetzt noch einen Klick auf "TimerID:None" (unten) und Euren "User Typ" wählen. (bei mir „waffen_timer“).



Jetzt zu "Then perform the following actions:". Hier wird der "movement type" für das Objekt "**waffe**" auf "None" gesetzt. Das sagt, das sich das Objekt nicht weiter drehen soll.

Ok, das war es. Script ist nicht einfach zu erklären, darum habe ich viele Bilder gemacht. Speichert es nun, indem Ihr auf die Schaltfläche "**Save**" klickt. Erstellt das MN3 File, in das Ihr alles einfügt. (`Euer.GAM` , `Euer.ORF` , `Euer.DLL` , `Euer.MSG`)



077 - Custom Robots Part 1

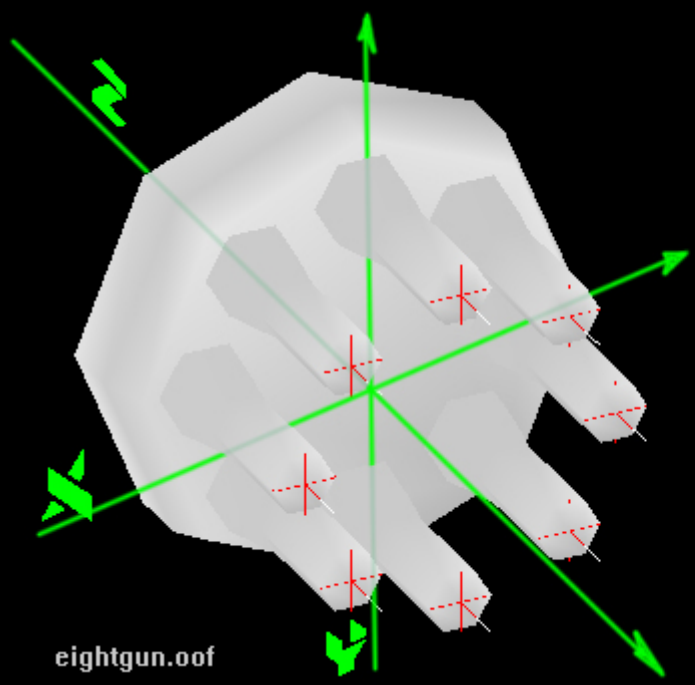
Papacat

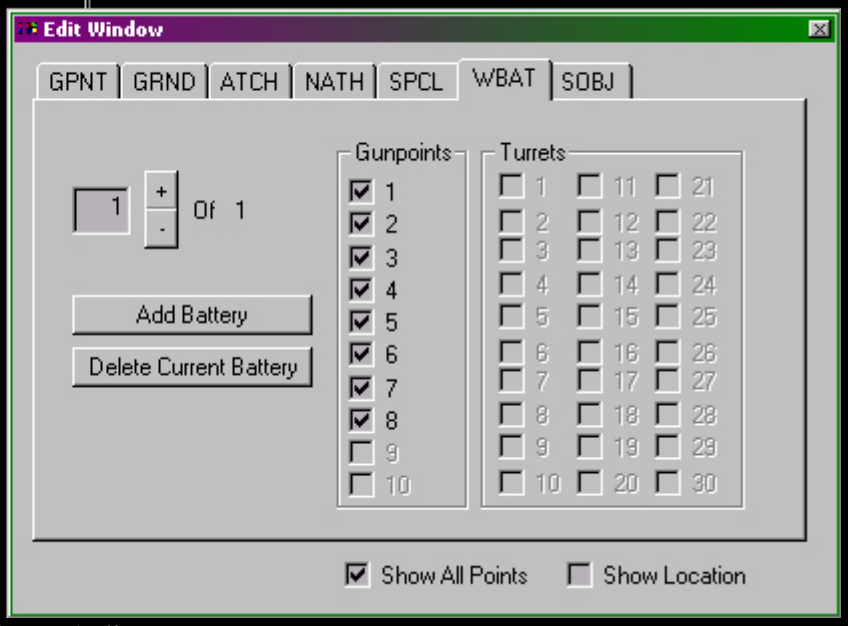
Hier gehts weiter, wo Objekte 3 geendet hat.

Beim Bauen eines Robots (oder Schiffes) ist es das Wichtigste daran zu denken welche Orientierung das Objekt hat. Stelle sicher, daß es in Richtung Z-Achse sieht: wenn Du in die Side View in D3Edit schaust, sollte Dein Objekt nach rechts schauen.

Wir verwenden das Ding aus dem vorigen Tutorial. Als erstes mußte ich die Orientierung ändern. Dazu bin ich ins D3Edit gegangen, habe die `.orf` geöffnet und gedreht. Dann habe ich die zurück ins D3Tool gebracht und die Gunpoints wieder draufgemacht - siehe voriges Tut.
Hast Du erstmal Deine Gunpoints drinnen mußt Du Batterien erstellen (Gruppen von Gunpoints).

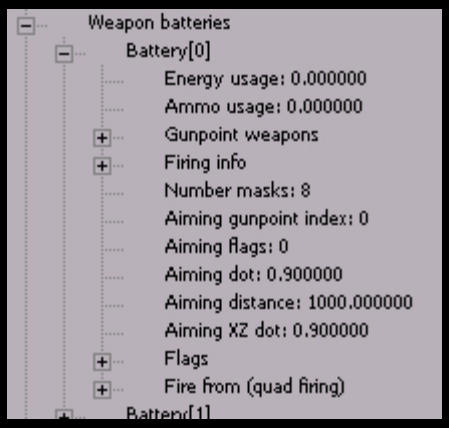
Im D3Tool gehe nach Tools -> Edit (gleiche Dialogbox wie für die Gunpoints) und wähle die WBAT-Tab (Weapon Batteries). Unter Gunpoints solltest Du nun für jeden dieser eine aktive Checkbox haben.





Ich habe hier alle acht angehakt. Speichere (Export) die `.oof`.

Als nächstes machen wir die .gam-Seite für einen Bot, dessen Verhalten für uns ein guter Startpunkt ist. Ich habe die .gam-Seite von einem ‚RAS1 Light Security Flyer' kopiert und als neue Datei verspeichert. Als erstes habe ich für den Bot Flags/Uses Physics gesetzt, sodaß er diese neben seiner normalen AI zum Rotieren verwenden wird, also wird der Bot beides verwenden, AI und Physik. AI verwendet er zum Zielen, Feuern, Manövrieren.



Als nächstes müssen wir die Weapon Batteries konfigurieren. Dort sagen wir dem Bot was er verschießen soll, von wo und wieviel oder oft. Öffne den Zweig Weapon batteries. Laß die Aiming-Einstellungen in Ruhe. Wir müssen die erste (und in unserem Fall einzige) Batterie einstellen, also öffne den Zweig Battery[0].

1	Vauss
3	Plasma ((43)Particle)
4	Fusion
6	Mass Driver
7	Napalm
9	Omega
10	Concussion
11	homing
12	Impact Mortar
13	Smart
13	Smart
14	Mega
15	Frag
16	Guided
20	Yellow flare
40	Raygun microwave
46	EMDblob
48	BlackShark
51	Cyclone Pack
53	NapalmRocket
57	Laser Level 1 - Red
58	Laser Level 2 - Blue
61	Laser Level 5 -Yellow

- Gunpoint weapons
  - Weapon index for gunpoint[0]: 53
  - Weapon index for gunpoint[1]: 53
  - Weapon index for gunpoint[2]: 53
  - Weapon index for gunpoint[3]: 53
  - Weapon index for gunpoint[4]: 53
  - Weapon index for gunpoint[5]: 53
  - Weapon index for gunpoint[6]: 53
  - Weapon index for gunpoint[7]: 53
- Firing info
  - Firing info[0]
    - Fire from
      - [x] Gunpoint 0
      - [x] Gunpoint 1
      - [x] Gunpoint 2
      - [x] Gunpoint 3
      - [x] Gunpoint 4
      - [x] Gunpoint 5
      - [x] Gunpoint 6
      - [x] Gunpoint 7
    - Wait time: 1.000000
    - Animation time: 0.000000
    - Animation start frame: 0.000000
    - Animation fire frame: 0.000000
    - Animation end frame: 0.000000
  - Firing info[1]

Als erstes öffne den Zweig Gunpoint Weapons. Hier weisen wir jedem Gunpoint eine Waffe zu, wozu wir die Waffenindexnummern verwenden. (Tabelle links außen)

Wo hat er die bloß her?

Firing info arbeitet in Beziehung zu den Number Masks. Diese sind wie eine Feuersequenz-Schleife. 8 Masken bedeutet, daß in jeder Schleife 8 Salven sind.

Denke von jedem Firing info-Eintrag als einer Salve. Um dies zu demonstrieren, werden wir die erste der acht Salven von allen acht Gunpoints abschießen, indem wir die Firing info[0] wie im Screenie links setzen.

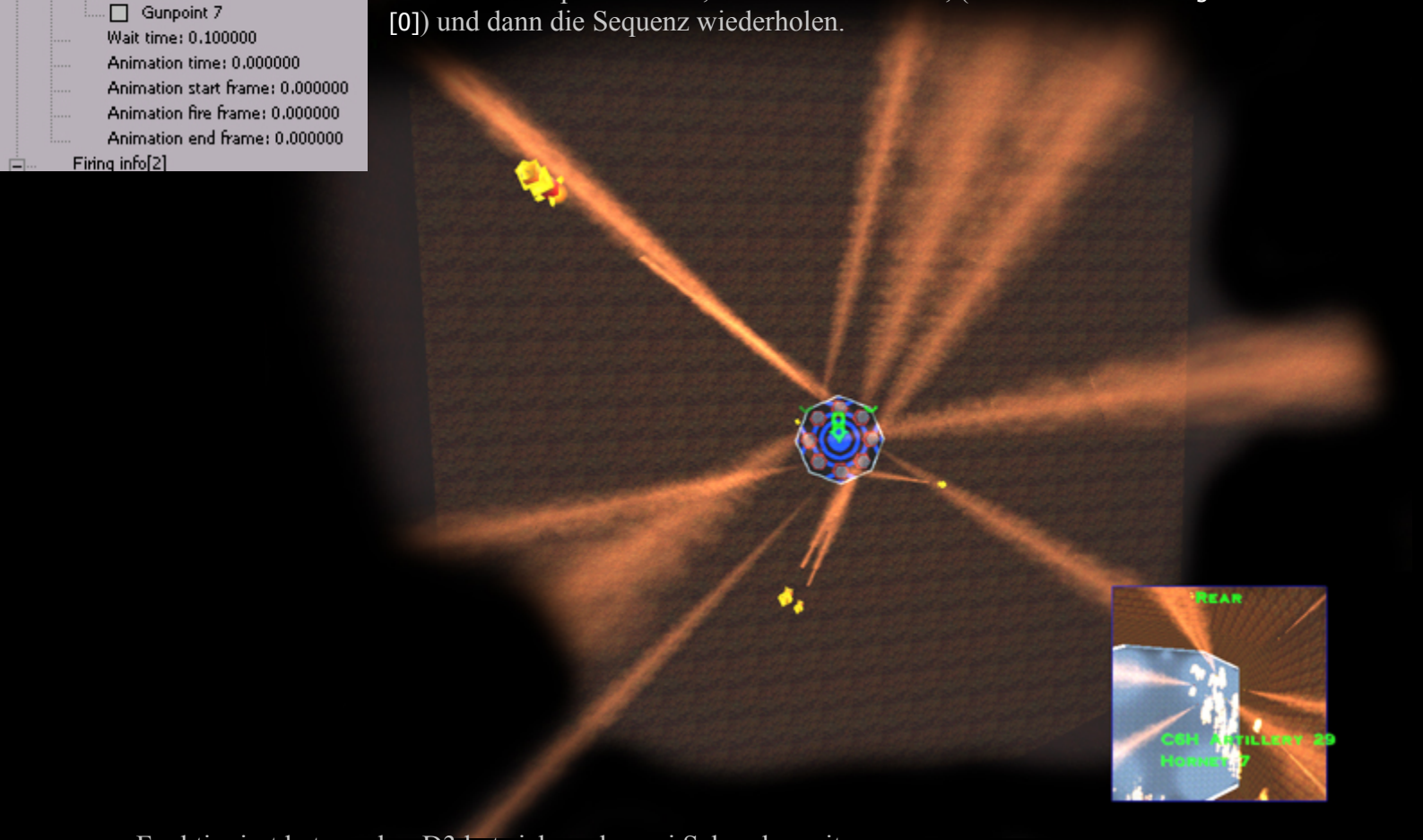
- Weapon batteries
- Weapons
  - Battery[0]
    - Weapon on gunpoint[0]: NapalmRocket
    - Weapon on gunpoint[1]: NapalmRocket
    - Weapon on gunpoint[2]: NapalmRocket
    - Weapon on gunpoint[3]: NapalmRocket
    - Weapon on gunpoint[4]: NapalmRocket
    - Weapon on gunpoint[5]: NapalmRocket
    - Weapon on gunpoint[6]: NapalmRocket
    - Weapon on gunpoint[7]: NapalmRocket
  - Battery[1]

Update:
Man muß auch noch unter Weapons/Battery[0] die Waffen für jeden Gunpoint als Text eintragen, auch nach der Tabelle links außen.

- Firing info[1]
  - Fire from
    - [ ] Gunpoint 0
    - [x] Gunpoint 1
    - [ ] Gunpoint 2
    - [ ] Gunpoint 3
    - [ ] Gunpoint 4
    - [ ] Gunpoint 5
    - [ ] Gunpoint 6
    - [ ] Gunpoint 7
  - Wait time: 0.100000
  - Animation time: 0.000000
  - Animation start frame: 0.000000
  - Animation fire frame: 0.000000
  - Animation end frame: 0.000000
- Firing info[2]

Dann setze Firing info[1] bis Firing info[8] wie im Screenie links, mit einer Wait time von 0.1 (1/10 Sekunde). Laß [1] vom Gunpoint 1, [2] vom Gunpoint 2... bis [7] vom Gunpoint 7 schießen.

Diese Einstellung wird eine Acht-Napalmraketen-Salve abfeuern, gefolgt von 7 einzelnen Napalmraketen, eine Sekunde Pause, (Wait time in der Firing info [0]) und dann die Sequenz wiederholen.



Funktioniert hat es, aber D3 hat sich nach zwei Sekunden mit einem Speicherfehler verabschiedet 😬 Ihr müßt also (wieder mal) selbst herumprobieren.

CUSTOM WEAPONS WORKSHOP

BY PAPACAT

Die Schnellsieder-Version:

1) Erzeuge ein `.oof` für Dein Powerup
2) Erzeuge ein `.oof`, .ogf oder .oaf zur Verwendung als Schuß
3) Erzeuge ein 64x64 und ein 32x32 `.ogf` für Dein HUD-Bild und -Icon
4) Kopiere die GAM-Seite für die Waffe und ihr Powerup die Du auszutauschen beabsichtigst
5) Kopiere die Schiffs-GAMs wenn Du Änderungen an den Gunpoints, Feuer-Details oder den Nutzdaten vornehmen willst.
6) Ändere die GAM des Powerups derart um daß sie Dein `.oof` verwendet. Vielleicht willst Du auch den Ammo Count verändern, wenn anwendbar.
7) Ändere die .gam der Waffe daß sie Dein Feuer-Bild, HUD-Bild und Icon verwendet. Hier kannst Du auch ihr Verhalten, ihre Beschädigungs-, Sound-, und visuelle Effekte ändern.
8) Packe mit **Add Files** die Dateien in Dein `.mn3`

Fertig.

Was Du hier lernen wirst:

1) Die Struktur der Ereignisse die eine Waffe betreffen
2) Die verschiedenen benötigten Dateien um eine Waffe zu produzieren
3) Wie unterschiedliche Einstellungen das Verhalten der Waffe beeinflussen
4) die GAM-Einstellungen welche beinahe alle Aspekte des Verhaltens Deiner Waffe kontrollieren.

Was Du hier NICHT lernen wirst:

Erzeugen von: OOF, OGF, OAF
Modding-Arbeit

Ich konnte Papacat's originale Objektdateien nicht mehr auftreiben. Daher habe ich meine eigenen erzeugt, und bin von den Originalen etwas abgewichen.

078 - Einführung & Übersicht

Papacat

Ich persönlich denke, daß viele Leute ‚Customizing' oder das Verwenden von Custom-Elementen mit ‚Modding' verwechseln. Mods schließen C++-Skripting ein. Es gibt aber eine Fülle was wir durch Erzeugung von .oof-, .ogf-, .oaf-Dateien und der GAM-Datei machen können

Dieser Workshop ist fünfteilig. Der erste Part (dieser hier) wird die Übersicht abdecken, was eine Waffe ausmacht. Der zweite wird sich mit der Powerup-`.gam` beschäftigen. Der dritte wird eine Übersicht über die `weapon.gam`. Im vierten Teil werden wir das alles zusammenbringen, um eine einfache Primärwaffe zu erzeugen. Schließlich werden wir die Schiffs-`.gam` untersuchen, wo wir die Feuer-Details der Waffe einstellen werden. Ich werde nicht auf Countermeasures eingehen, weil sie nicht von einem Schiff abgefeuert werden, was sie sehr limitiert. Die meisten Gegenmaßnahmen (Minen, Sucher und Gunboys) sind eigentlich Robots, welche von unsichtbaren Waffen spawnen (das wird nach dem Durchnehmen der `weapon.gam` mehr Sinn ergeben)

Eine kurze Übersicht über die Basis-Definitionen:

`.oof`	Outrage Object File	Physikalisches Objektmodell wie ein Powerup, eine Rakete, Fässer, Kisten, etc... (siehe vorige Tuts)
`.ogf`	Outrage Graphics File	Eine statische Textur (Bild). Beachte, daß ein `.ogf` ‚gleiten' kann, aber das macht es nicht zu einem ‚echten' `.oaf`.
`.oaf`	Outrage Animation File	Ein Animations-Clip, ähnlich einem animierten `.gif`. Der Unterschied zwischen einem `.oaf` und einem gleitendem `.ogf` ist daß bei letzterem EIN Bild bewegt (verschoben) wird, bei ersterem verschiedene Bilder (Frames) die Animation erzeugen.
`.gam`	Game Table File	Die GAM-Datei enthält die Seiten für jedes Element, sie beschreiben deren Eigenschaften und Verhalten.

Du mußt die `Table.gam` (die Haupt-GAM-Datei von D3) und die `.oofs`, `.ogfs` und `.oafs` durchsehen, die bei Waffen alle involviert sind. Zum Extrahieren der einzelnen Dateien aus der `d3.hog` verwende Hog2WorkShop.

Du wirst folgende Tools brauchen (Nebst D3Edit, versteht sich):

Hog2WorkShop	Extrahieren der Sub-Dateien
gamTool0.61	Bearbeiten der .gam-Datei(en)
ModelView32	Zum bearbeiten von `.oof`-Dateien
OOFEditor	

Hast Du die Tools erstmal installiert kanns losgehn.

Als erstes bedenke, daß wir ohne Modding mit C++ eine existierende Waffe ERSETZEN müssen. Der größte Nachteil dessen ist, daß wir die originalen Namen der Originalwaffe und ihrem Powerup behalten müssen, so daß ein Spieler sie verwenden kann. Wenn Du jemals Schplurg's Earthshaker 2001 oder mein Level, Vortex 2, gespielt hast wirst Du bemerkt haben daß wenn Du eine Earthshaker aufnimmst dort noch immer ‚Blackshark' steht. Sogar wenn Du eine Hornet auf Deinem HUD siehst, steht darüber immer noch ‚Cyclone' Abgesehen davon können wir immer noch jede Menge Fun haben.

Hier ist eine schnelle Übersicht der zugrundeliegenden Dateien die eine Custom-Waffe ausmachen:

1) ein `.oof` für das Powerup
2) ein `.oof`, `.ogf` oder `.oaf` für das Feuer-Bild (den Schuß)
3) ein 64x64 `.ogf` für das Bild im HUD (Cockpit geschlossen)
4) ein 32x32 `.ogf` für das Icon im offenen Cockpit
5) eine `.gam`, welche folgende Speicherseiten enthält:
   a) eine PAGETYPE_GENERIC für das Powerup
   b) eine PAGETYPE_WEAPON für die Waffe selbst
   c) die PAGETYPE_SHIPs um die Feuerdetails und Nutzdaten einzustellen, so anwendbar (optional)

Natürlich brauchst Du kein Powerup oder die HUD-Bilder, wenn die Waffe für einen Robot ist.

Um zu verstehen was diese Dateien machen, gehen wir durch den Lebenszyklus einer Waffe. Dies trifft mehr auf Primär- und Sekundärwaffen zu, Countermeasures sind etwas andere Kreaturen.

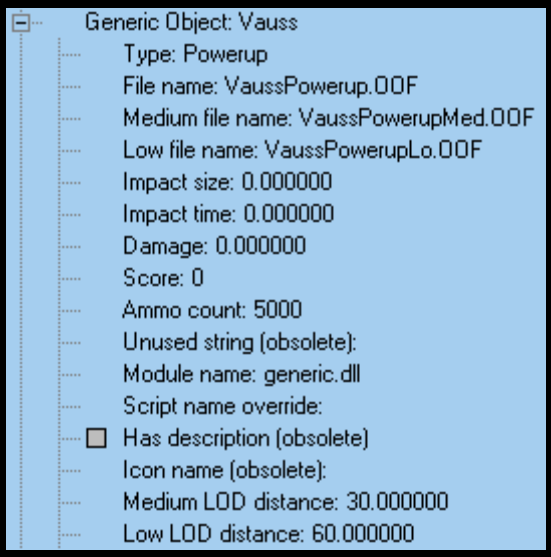
1) Der Spieler nimmt das Powerup auf.
   Dies macht die Waffe für das Spielerschiff verwendbar. Abhängig von der Natur der Waffe, kann sie auch die Munitionsmenge setzen. Diese kommt aus der .gam des Powerups.
2) Der Spieler wählt die Waffe aus.
   Wenn das Cockpit an ist, wird das 64x64 `.ogf` HUD-Bild angezeigt, wenn das Cockpit aus ist das 32x32 `.ogf` Icon. Beide Bilder werden in der GAM der Waffe spezifiziert.
3) Der Spieler feuert die Waffe ab.
   Ein ‚Schußbild' (der Schuß) wird von der Waffe generiert. Das Bild selbst ist entweder ein `.oof`, ein `.ogf` oder ein `.oaf`. Bedenke, es kann einfach nur ein Bild sein. Das Objekt erzeugt nicht selbst den Schaden, die Waffe macht das. Munition oder Energie wird dem Spieler abgezogen.
4) Die Waffe schlägt in einen Spieler ein.
   Drei Dinge passieren:
   a) eine Explosion – Der Effekt des Treffers der Waffe.
   b) Beschädigung an Spieler oder Objekt – Schild wird vom getroffenen Objekt abgezogen.
   c) Einschlag, der Treffer – Die physikalische Reaktion vom Treffer (Beispiel: MD wirft Dich durch den Raum)
5) Dem Spieler geht die Munition oder Energie aus,
   was die Waffe inoperabel macht bis der Spieler mehr Munition oder Energie besorgt, wenn es eine Primary ist. Wenn es eine Secondary ist, ist sie nicht mehr verfügbar und re-spawnt.
6) Der Spieler wird gekillt
   Das Powerup wird vom Schiff ausgespuckt. Immer bei Primaries, nur wenn noch welche über sind bei Secondaries.

Wir haben nun offiziell die Oberfläche angekratzt. Recht langweilig bis jetzt, aber nun hast Du eine Ahnung was Du alles bedenken mußt. Nun werden wir in die GAM's schauen, als erstes fürs Powerup.

079 - Die Powerup GAM

Papacat

Schauen wir was wir in der Seite für das Powerup einstellen müssen. Die GAM ist vom Typ PAGETYPE_GENERIC, es ist keine Waffe. Wenn ein Spieler sie aufnimmt (damit kollidiert) wird die Waffe die es repräsentiert verfügbar, und sofern anwendbar, wird Munition zur Nutzlast des Spielers hinzugefügt. Um einen Überblick zu kriegen, sehen wir uns den GAM-Eintrag für die Vauss an.



Generic Object: Der Objekt-Name
Type: Powerup, Robot oder Clutter
File name: Die Haupt-`.oof`, voll detailliert; so wie man es von Nahe sieht (erforderlich)
Medium file name: das bei einer mittleren Distanz gezeigte `.oof`
Low file name: Bei weiten Distanzen angezeigtes `.oof`
Ammo Count: Menge an Munition die zur Nutzlast hinzukommt
Module name: Das D3-Skript, welches das Verhalten der Waffe kontrolliert (nicht anfassen!)
Medium LOD distance: Die Entfernung vom Powerup, bei der D3 die mittel detaillierte .oof anstatt der Haupt-`.oof` verwendet.
Low LOD distance: Ab wann D3 das am wenigsten detaillierte `.oof` anzeigt.

Wenn Du die GAM-Seite der Vauss öffnest, wirst Du sehen daß da noch viel mehr drinnen ist. Der oben gezeigte Teil ist alles um was Du dich kümmern mußt. Manche Leute glauben, um das Powerup rotieren zu lassen muß man den Physik-Block einrichten. Nicht Wahr! Powerups rotieren indem man die Eigenschaft `$rotate` zur `.oof` selbst hinzufügt.

Oben siehst Du, daß die GAM drei verschiedene `.oof`s angibt. Um die Spielperformance zu maximieren, dachte sich Outrage daß Du einen bestimmten Detailgrad aus einer bestimmten Entfernung nicht mehr sehen kannst, also warum das Spiel das zeichnen lassen. Wenn Du nahe bist, siehst Du das voll detaillierte .oof. Basierend auf der oben gezeigten Medium LOD- distance, siehst Du das Powerup zwischen 30 und 60 Einheiten Entfernung als weniger detaillierteres .oof. Wenn Du 60 Einheiten und weiter weg bist, zeigt Dir D3 das niedrig detaillierte .oof.

Du kannst ein einzelnes `.oof` verwenden und die Namensfelder für Medium und Low leer lassen.

Weiter zur Weapon-GAM.

080 - Die Weapon GAM

Papacat

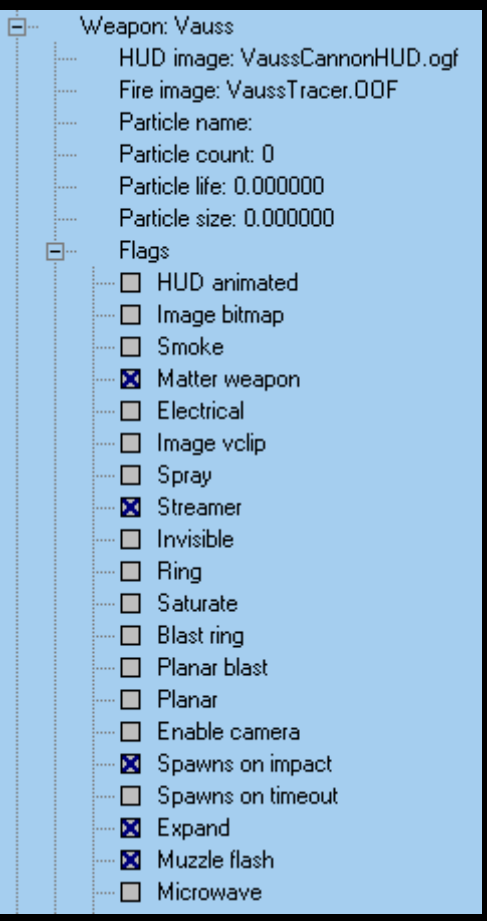
Diese Sektion ist eine Referenz, um die GAM-Seite für eine Waffe zu verstehen. Sie ist eine Kulmination von Information aus der Outrage Tablefile specification (ab Seite 494) und verschiedener Readme-Files von Aldel, Dark Knight und Peter Runge. Zusätzlicher Input und Verdeutlichungen sind von SuperSheep und BlackPanther gekommen. Dank an alle.

Ich habe hier versucht das ganze durch Beispiele verständlicher zu machen, wo sie helfen könnten. Wenn ein Eintrag leer ist habe ich KEINE Information. Wenn da ein Fragezeichen steht ist es zwar Info, aber ich verstehe es nicht und würde ein Beispiel schätzen daß die Verwendung verdeutlicht.

Eine Sache die verwirrt ist wenn der Name von etwas wie dem Schuß-Bild oder Particle name erforderlich ist. Manchesmal, wie im Fall von Fire image, erfordert es den Namen eines `.oof`, `.ogf` oder `.oaf`. In anderen Fällen, wie bei Particle name, geht es um den Namen eines GAM-Eintrages in der `.gam`. Ich werde versuchen aufzuzeigen, welche.

Beachte daß Energieverbrauch und Feuerrate hier nicht behandelt werden. Das wird im Zweig Weapons in der Schiffs-GAM spezifiziert.

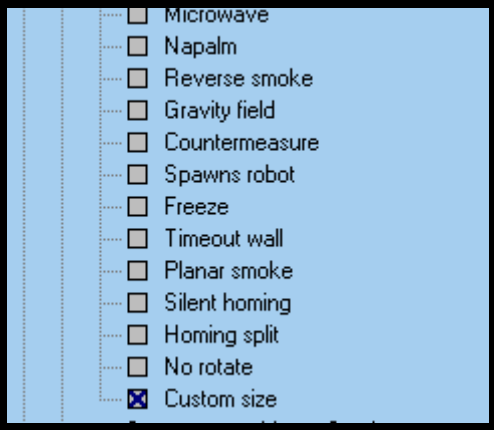
Hud Image: Gezeigtes Waffenbild (64x64), wenn Cockpit an ist
Fire Image: Der Schuß der das Schiff verläßt (`oof`. `ogf`, oder `oaf`)
Particle name: Partikel, die hinter dem Schuß-Bild herkommen (gewöhnlich eine Waffe) Beispiel: die ‚Plasma-Funken' hinter den homenden Plasmablobs einer Smart.



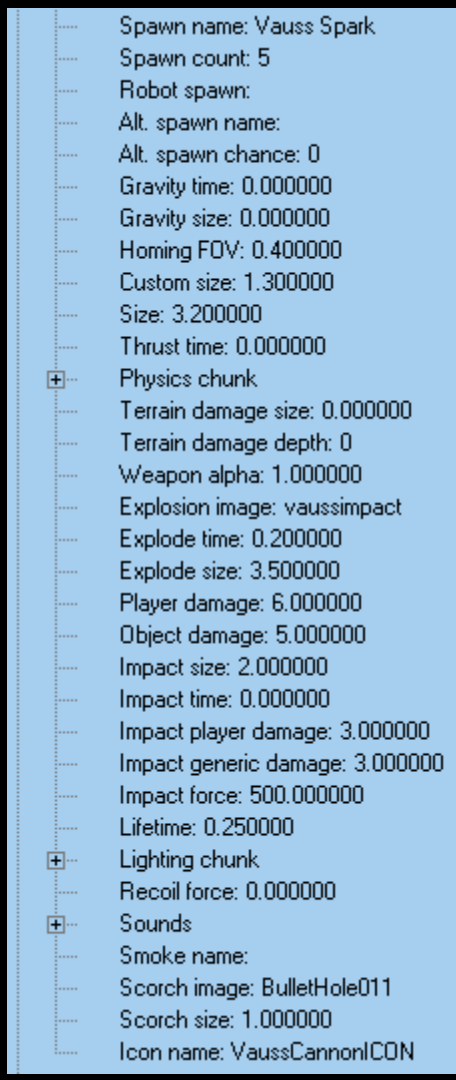
Particle Life: Lebensdauer des Partikels.
Particle size: ?

Flags

Hud animated: ?
Image bitmap: Das Schußbild ist ein Bitmap (`.ogf`). Beispiel: Die Omega
Smoke: Rauch ist hinter dem Schußbild. Beispiel: Concussion Missile
Matter Weapon: Die Waffe ist ein physisches Objekt. Beispiel Vauss, Concussion (erfordert Munition)
Electrical: Electrical Storm weapon (Omega)
Image Clip: Das Schußbild ist ein Videoclip (`.oaf`) Beispiel: Flammenwerfer (die Napalm)
Spray: Die Waffe ist ein Spray (😬) Beispiel Flammenwerfer
Streamer: ‚Band'-Effect (Streifen) hinter der Waffe. Beispiel: Streifen hinter den Partikeln einer Frag Missile
Invisible: Unsichtbare Waffe. Für einen Gunboy wird eine unsichtbare Waffe abgeworfen die den Gunboy spawnt.
Ring: Weapon is drawn ring style?
Saturate: Saturate this bitmap weapon?
Blast Ring: Creates a Blast Ring upon explosion?
Planar Blast: Explosionsbild bleibt immer parallel zur Wand (abschalten = das Explosionsbild sieht immer zu Dir hin)
Planar: Die Waffe sieht nicht immer zu Dir.
Enable Camera: Kamerasicht von der Waffe. Beispiel: Guided Missile
Spawns on impact: Spawnt eine Waffe wenn sie ein Objekt trifft. Beispiel: Smart Missle
Spawns on timeout: Spawnt eine Waffe, wenn die Lebensdauer abläuft. Beispiel: Cyclone
Expand: Das Explosionsbild der Waffe vergrößert sich
Muzzle Flash: Mündungsfeuer wird am Gunpoint angezeigt, von dem geschossen wird
Microwave: Waffe erzeugt den ‚Mikrowellen-Effekt' bei Objekten

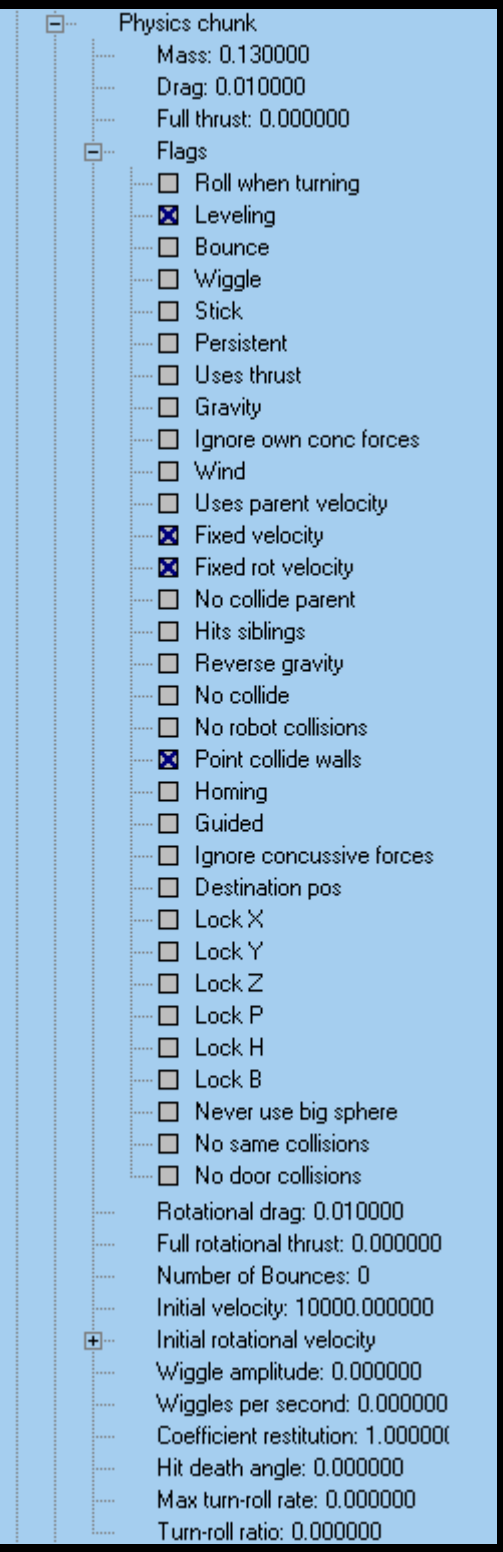


Napalm: Napalm-Effekt bei Objekten
Reverse Smoke: Rauchspur wird kleiner. Beispiel: schrumpfende Ringe der Microwave
Gravity Field: Erzeugt ein Gravitationsfeld (BlackShark)
Countermeasure: Waffe ist eine Gegenmaßnahme
Spawns Robot: Spawnt einen Bot anstatt einer Waffe. Beispiel: Gunboy
Freeze: Verlangsamt den getroffenen Spieler.
Timeout wall: Ein Wandtreffer endet wie ein Time out.
Planar Smoke: Planare Rauchspur anstatt Blobs. Beispiel: Homers und Smarties
No Rotate: This weapon does not rotate as a bitmap?
Custom Size: Uses Custom Size?



Spawn Name: Name der bei Einschlag spawnenden Waffe
Spawn Count: Menge an Spawns (in diesem Fall spawnen fünf Funken beim Einschlag
Alt. Spawn Name: Eine alternative Waffe, die zu spawnen ist.
Alt. Spawn Chance: Chance (0 - 100%) daß die alternative Waffe spawnt.
Gravity Time: Wie lange soll das Gravitationsfeld andauern (-> BlackShark)
Gravity Size: Größe des Gravitationsfeld-Radius.
Homing FOV: Cosinus des Sichtfeldes für Homing-Waffen.
Custom Size: Custom-Größe der Waffe (Polymodel-Größe überschreiben)
Size: 'Blob'-Größe der Waffe.
Thrust Time: Wie lange die Schubkraft der Waffe andauert.
Physics Chunk: Physik-Einstellugen (ich werde das separat erklären)
Terrain Damage Size: Radius des Schadens, den die Waffe an Terrain verursacht.
Terrain Damage Depth: Wie tief dringt der Schaden ins Terrian.
Weapon Alpha: Durchsichtigkeit der Waffe (0 to 1.0, 0=voll transparent)
Explosion Image: D3-Texturname (GAM) der verwendeten Textur, wenn die Waffe explodiert.
Explode Time: Wie lange das Explosionsbild existiert.
Explode Size: Größe der Explosion
Player Damage: Schaden, den die Waffe an Spielern verursacht.
Object Damage: Schaden, den die Waffe an allgemeinen Objekten verursacht.
Impact Size: Radius der wirksamen Sphäre, wenn die Waffe einschlägt und explodiert.
Impact Time: Wie lange der Einschlagseffekt andauert.
Impact Player Damage: Wieviel Schaden der Einschlag an Spielern verursacht.
Impact Generic Damage: Wieviel Schaden der Einschlag an allgemeinen Objekten verursacht.
Impact Force: Betrag der Kraft, die durch die Einschlags-Explosion freigesetzt wird.
Lifetime: Wie lange die Waffe lebt. Wenn ihre Lebensdauer endet, macht sie ein "TIMES OUT"
Recoil Force: Rückstoß, der beim Abschuß frei wird (Mass Driver oder Mega)
Sound Name: D3 Sound-Name (GAM) des Sounds, der mit diesem Sound-Slot assoziiert ist.
Smoke Name: D3 Textur-Name (GAM) der Textur für das Rauchspur-Bild.
Scorch Image: D3 Textur-Name (GAM) der Textur, die für die Schuß-Spuren der Waffe verwendet wird.
Scorch Size: Größe der zu hinterlassenden Schuß-Spur
Icon name: Waffenbild das gezeigt wird wenn das Cockpit aus ist (32x32)

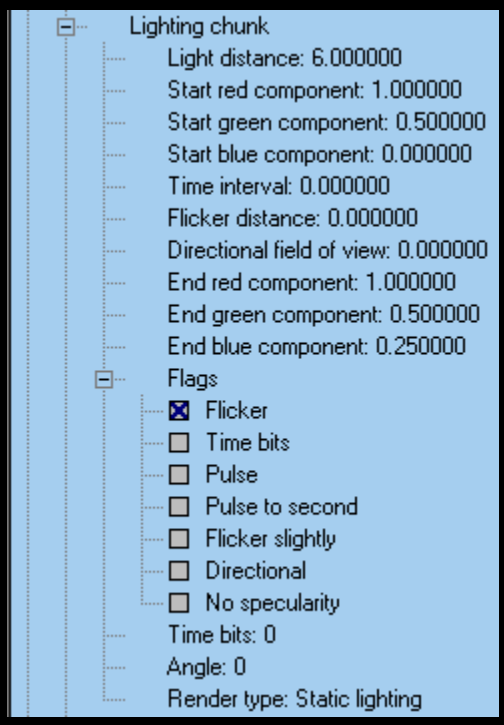
Physics Chunk geöffnet



Mass: Beeinflußt die Fähigkeit Geschwindigkeit ut Rotation zu bekommen.
Drag: Beeinflußt wie schnell sich die Bewegung eines Objektes verlangsamt (Widerstand versucht mal -100, hehe)
Full Thrust: Betrag der Schubkraft
Roll when turning: Rollt wenn es sich dreht
Leveling: Nivelliert das Objekt mit der nächsten Seite ?
Bounce: Prallt von Wänden ab
Wiggle: Bewegt sich auf und ab (wiggle) während des Fliegens
Stick: Bleibt an Wänden oder Objekten haften (prallt nicht ab oder timet nicht out)Beispiel: Napalm und Flares
Persistent: Objekt bewegt sich nach Kollision mit einem anderen weiter. Beispiel: Fusion
Uses Thrust: Benutzt seine Schubkraft
Gravity: Wird durch Gravitation beeinflußt. Beispiel: Napalm, Impact Mortar, Bouncing Betty
Ignores own conc forces: Ignoriert seine eigenen Concussion-Kräfte?
Wind: wird von Wind beeinflußt
Uses Parent Velocity:
Fixed Velocity: Konstante Geschwindigkeit
Fixed rot Velocity: Konstante Rotationsgeschwindigkeit
No Collide Parent: Kann nicht mit seinem Elternobjekt kollidieren
Hits siblings: Kann mit seinen Geschwistern kollidieren. Beispiel: proximity bomb
Reverse Gravity: Bewegt sich entgegen der Gravitation.
No Collide: "No collisions AND NO RELINKS -- DANGEROUS TO USE if not used correctly" ?
No robot collide: Kollidiert nicht mit Robots
Point collide walls: Bei Kollisionen mit der Wand rechne den Radius mit 0. Das heißt, der einzige physikalische Kontaktpunkt ist das Zentrum des Objekts, nicht seine Seiten.
Homing: Dieses Objekt (Waffe) ist zielsuchend (homing)
Guided: Dieses objekt ist geleitet; Guided Missiles
Never use big sphere:
No same collision:
No door collision: Collision won't open a door?
Rotational drag:
Full rotation thrust: Beeinflußt die Rotationsgeschwindigkeit
Number of Bounces: Anzahl an Abprallern vor einem Einschlags-TimeOut (explodiert oder stirbt). Beispiel: Mortar (2 Abpraller)
Initial Velocity: Geschwindigkeit vor Release oder Spawn bevor Schubkraft einwirkt
Initial rotional velocity: Anfängliche Rotationsgeschwindigkeit. Stelle die Geschwindigkeit für die Rotation bei jeder Achse ein. Öffnen um Zugriff auf X, Y und Z zu bekommen
Wiggle amplitude: Betrag dieser Bewegung.
Wiggles per second: Wie oft pro Sekunde.

Lighting Chunk geöffnet

Kontrolliert das Licht, daß ein Objekt abgibt. Beispiel: eine Cyclone einen dunklen Tunnel lang geschossen. Sie erleuchtet den Tunnel purpurn. Auch für rotierende Lichter.



Light distance: Entfernung, die das Licht noch erreicht? (need to test)
Start red component: Betrag Roten Lichtes (0-1.0)
Start Green light: Betrag Grünen Lichtes (0-1.0)
Start blue lighting: Betrag Blauen Lichtes (0-1.0)
Time Interval:
Flicker Distance: Entfernung, die flackerndes Licht noch erreicht? (need to test)
Directional Field: Gerichtetes Sichtfeld: Lichtkonusbreite 0 - 1.0 (Cosinus des Winkels?):
End red component: ? (für gewöhnlich die selbe wie Start)
End green component: ? (für gewöhnlich die selbe wie Start)
End blue component ? (für gewöhnlich die selbe wie Start)
Flicker: Licht flackert
Time bits:
Pulse: Licht pulsiert. Beispiel Not- oder Warnlichter
Pulse to second:
Directional: Licht ist ein Konus oder Strahl, Directional Field muß gesetzt sein
No Specurlarity: Does not have specular light cast upon it ?
Time Bits:
Angle: Unbenutzt??
Render Type: Wie das Licht gerendert werden soll, also wie es sich im Level ausbreitet und abzeichnet.

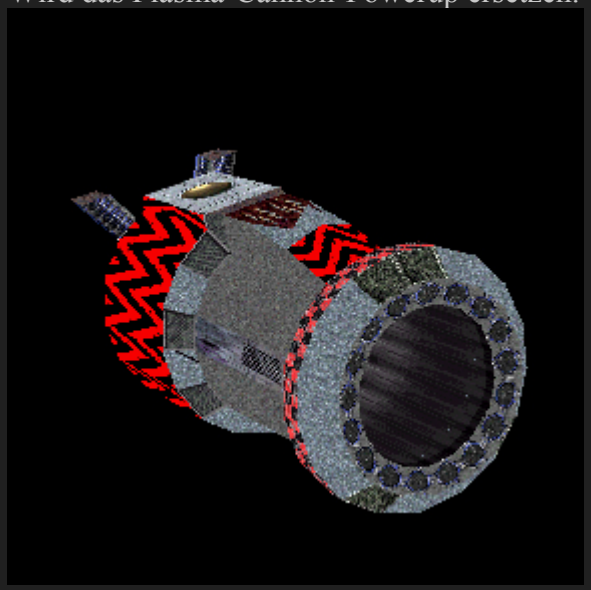
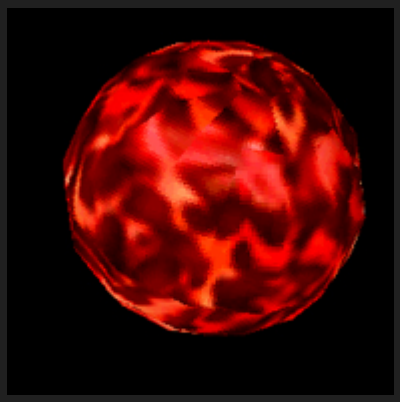
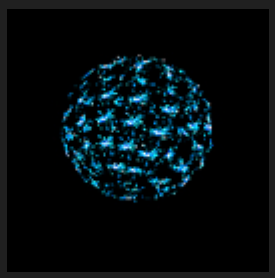
081 - Eine simple Primärwaffe

Papacat

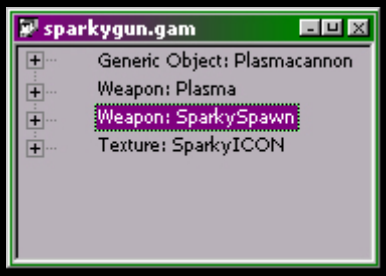
Für unsere Primärwaffe werden wir die Plasma Cannon (Waffenname: Plasma) ersetzen. Die dazu nötigen Dateien sind in der `.zip`, sodaß Du Dir das Selberbauen ersparst
Dieses Ding heißt Kugelblitzwerfer.

Wenn abgefeuert, siehst Du den Kugelblitz anstatt des Plasmablobs. Er wird auf drei Abpraller (=Bounces) eingestellt, bevor er bei einem Wandeinschlag explodiert. Bei Wandeinschlag spawnt er eine andere Waffe, den Sparkspawn; wir werden es so einstellen, daß drei spawnen. Die Sparkspawns werden wir auf vier Bounces bevor sie an einer Wand explodieren einstellen, oder daß sie nach 2,5 Sekunden outtimen.

Dies sind die fünf Objekte für die Waffe:

Sparkballgun.oof	Sparkball.oof	Sparkyspawn.oof	SparkyHUD.ogf
Wird das Plasma-Cannon-Powerup ersetzen. 	Das Schußbild beim Feuern. 	Schußbild der gespawnten Waffe 	64x64er-Bild für das HUD-Bild der Waffe
			SparkyICON.ogf 32x32er-Bild für das Bild der Waffe ohne Cockpit

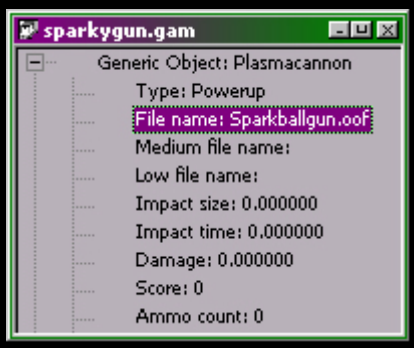
Wir müssen vier GAM-Seiten erzeugen:



Generic Object: Plasmacannon: Das Powerup.
Weapon: Plasma: Die Waffe, die wir ersetzen werden.
Weapon: SparkySpawn: Die zu spawnende Waffe. Beachte, da wir sie nicht vom Pyro aus abschießen, kann es ein beliebiger Name sein (Kopie von frag particles).
Texture: SparkyICON1: Die Textur für die Anzeige bei deaktiviertem Cockpit

Anmerkung: In den Screenshots hier seht Ihr aber überall den Eintrag ohne die ‚1'; doch dann funzt das ganze nicht. Haltet Euch also an den Text!

Generic Object: Plasmacannon



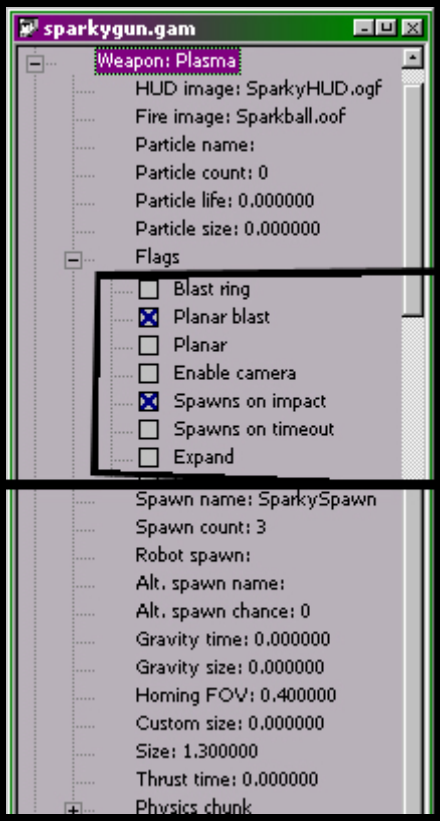
Wir müssen die Powerup-GAM der Plasma auf unser `.oof` umstellen.

Ich habe ‚`sparkballgun.oof`' als Dateinamen angegeben. Stelle sicher, daß Du die Einträge bei Medium und Low file name 'rauslöschst.

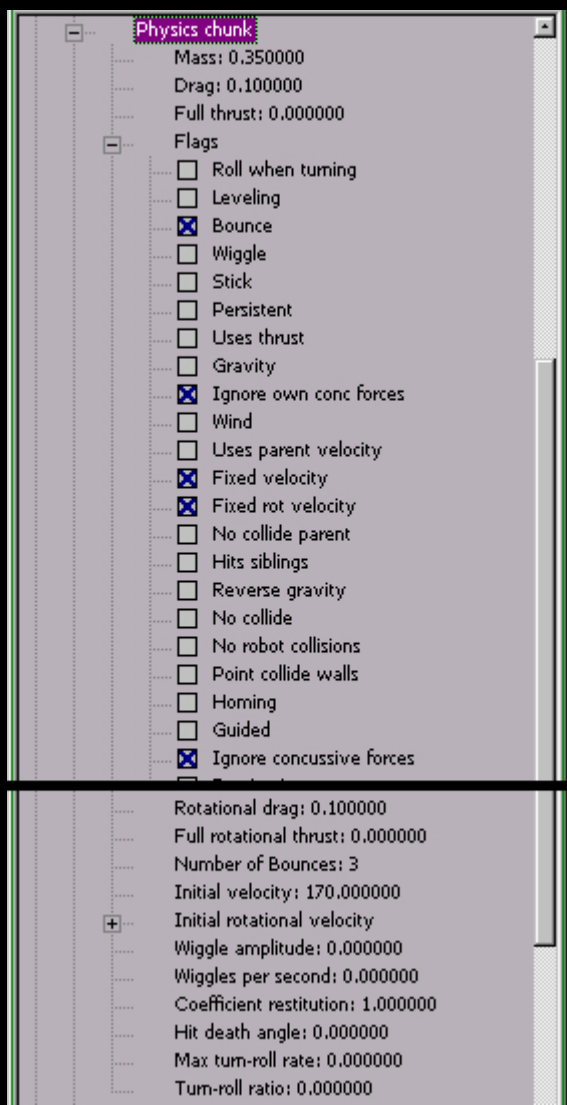
Nun sehen wir die Kugelblitzkanone anstatt dem Plasma-Cannon-Powerup.

Weapon: Plasma

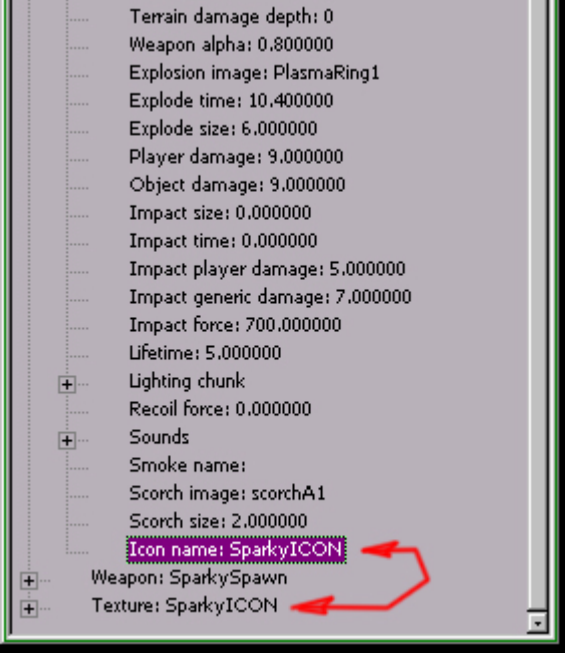
Einstellungen, die wir nicht behandeln habe ich bei diesen Grafiken ausgespart, um deren Größe zu reduzieren.



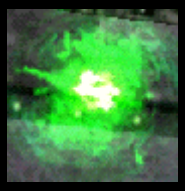
Stelle HUD image auf `sparkyHUD.ogf`; das ist die eigentliche `.ogf`-Datei, keine ‚Texture'.
Setze Fire image auf `sparkball.oof`
Den Particle name lösche raus (falls nicht leer), sodaß es keine Funkenspur gibt.
Planar Blast gesetzt lassen
Spawns on impact einschalten.
Setze den Spawn name auf Sparkyspawn. Bemerke: ich gebe hier den GAM-Namen an, nicht die `.oof`!
Setze Spawn count auf 3. Es werden drei der blauen Bälle spawnen, wenn der Schuß an der Wand einschlägt und explodiert.



Bounce: Der Schuß wird von der Wand eine bestimmte Anzahl oft abprallen, bevor er bei einem Wand-Einschlag explodiert.
Fixed velocity: Geschwindigkeit bleibt konstant.
Fixet rot velocity: Die Rotationsgeschwindigkeit wird sich nicht ändern.
Number of Bounces: Wie oft soll's von der Wand abprallen. Beim vierten Wand-Treffer wird's explodieren.
Initial velocity: Die Geschwindigkeit, mit der der Schuß die Waffe verläßt.

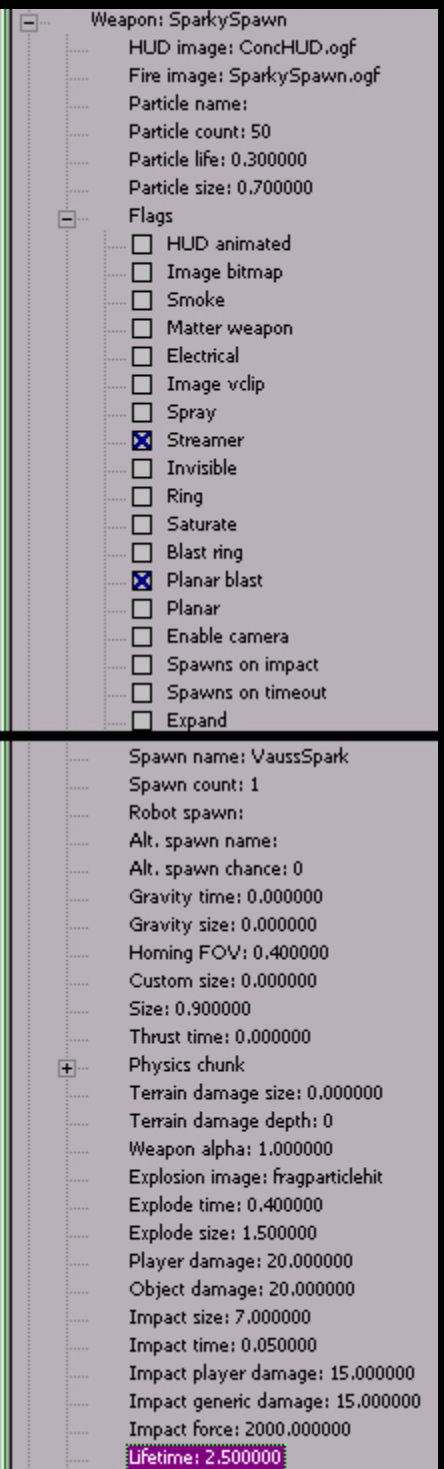


Die Explode time auf 10.4 erhöht, aus drei Gründen: Erstens kanns Du sehen, daß ein Planar Blast parallel zur Wand bleibt. Zweitens, wie die Einstellung das Explosionsbild sichtbar hält. Drittens, wie die gesetzte Flag Expand das Explosionsbild sich ausdehnen läßt.



Den Icon name setze auf den GAM-Eintrag Deiner Texture, SparkyICON1, der auf die eigentliche `.ogf` zeigt (Doppelpfeil). Die siehst Du, wenn Dein Cockpit ausgeschaltet ist.

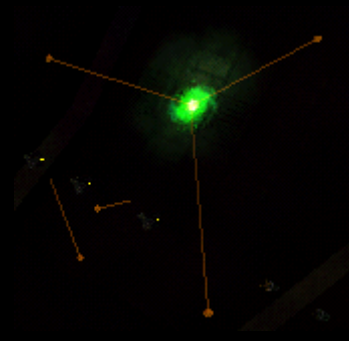
Weapon: SparkySpawn:



Ändere Fire image auf `SparkySpawn.oof`

Lösche Particle name. Ohne die Partikel sind die Streamer offensichtlich.

Streamer-Flag setzen. Rechts kannst Du eine streifenförmige Spur beim Schußbild sehen, das ist der Streamer.



Planar Blast anlassen. Merke, die grüne Explosion oben rechts ist die Explosion der Plasma, nicht das Sparkyspawn.

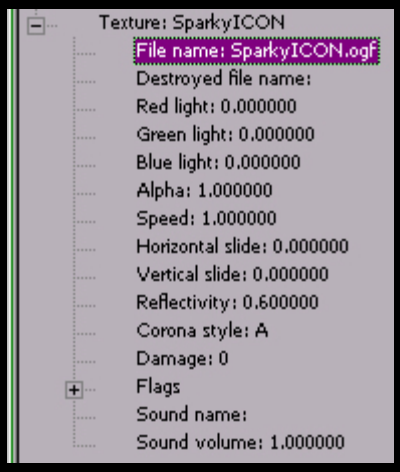
Spawns on impact und Spawns on timeout aus lassen, sodaß keine Waffen mehr erzeugt werden.

Beachte, daß Spawn name hier wurscht ist, da keine der Spawn-Flags an sind.

(Hier nicht gezeigt) im Physics Chunk setze das Flag Bounce und Number of Bounces auf 4.

Setze die Lifetime auf 2.5 sodaß Du Zeit hast, es abprallen zu sehen.

Texture: SparkyICON1:



Letztlich, um eine Anzeige am HUD ohne Cockpit zu haben mußt Du eine Textur erzeugen.

Gib bei File name den Namen des `.ogf`'s an.

Sonst ist hier nix einzustellen.

Das wars. Ausprobieren kannst Du das Ganze, indem Du die `.gam` und die fünf Objekte in ein Level einbaust, wo es eine Plasma Cannon gibt. Vergiß nicht, daß für die `.mn3` und die `.gam` Namensgleichheit gegben sein muß. Wenn der Level bereits eine eigene `.gam` hat, füge die vier GAM-Seiten einfach dort hinzu.

Boom!

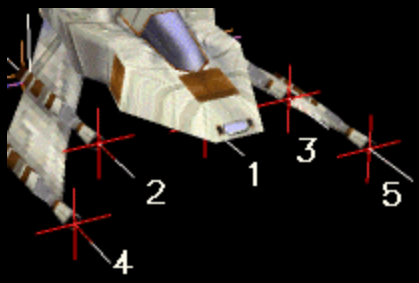
Natürlich kann man sich jetzt in die Details verlieren... eigene Waffensounds erstellen und einbinden, an den Einstellungen herumdrehen... etc...

082 - Die Schiffs-GAM

Papacat

Da wir jetzt wissen wie man die Waffe kontrolliert müssen wir schauen wie man ihr eigentliches Feuern vom Schiff kontrolliert. Die Schiffs-GAM-Seiten sind der Ort wo wir wählen von welchem Gunpoint aus gefeuert wird, die Feuerrate, maximale Zuladung etc.

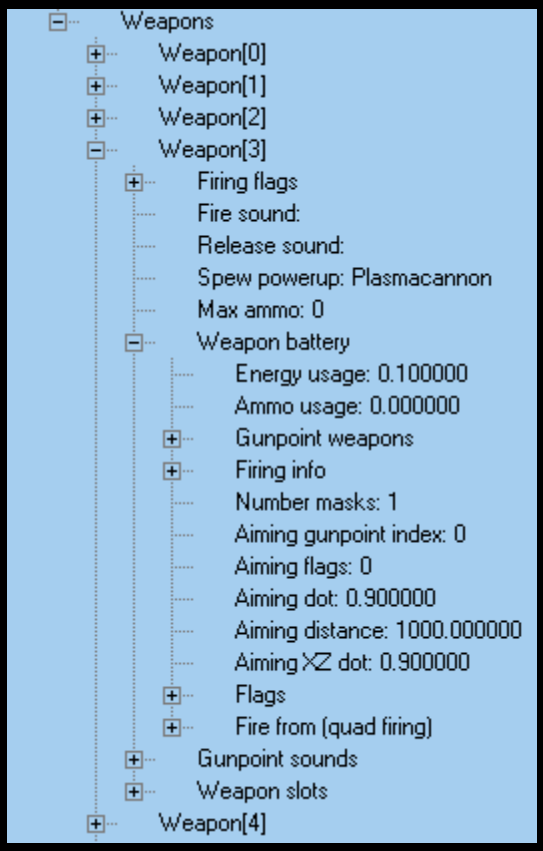
Die Waffen-Setups bei Schiffen sind ähnlich denen der Robots. Allerdings gibt es einen GROSSEN Unterschied, Schiffe verwenden nur eine Battery, also wird alles in den Waffeneinträgen spezifiziert.



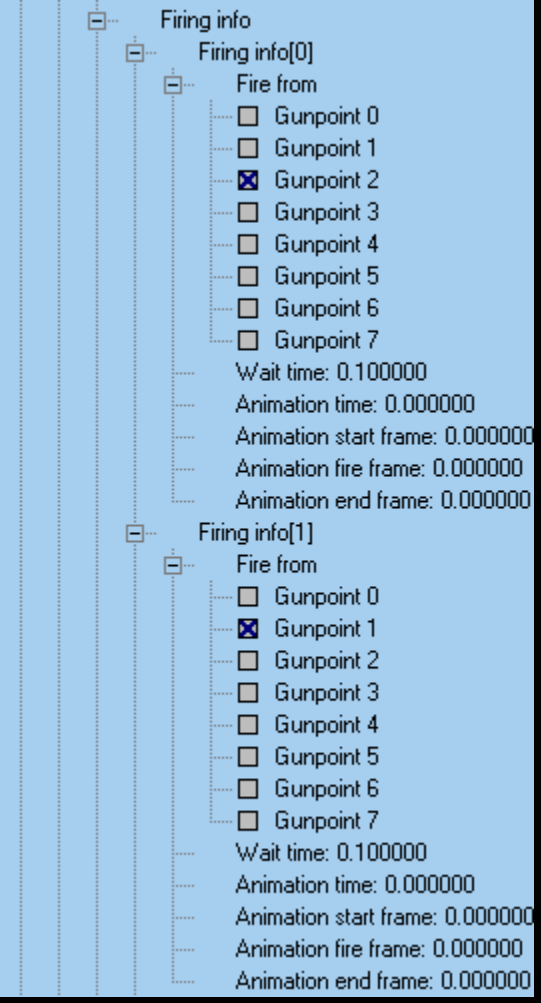
Da sich vieles hier auf Gunpoints bezieht, habe ich diese Diagramme eines Pyro reingetan.

Jede vom Schiff verwendete Waffe ist unter Weapons eingeragen. Weapon[0] bis Weapon[4] sind die ersten fünf Primärwaffen, [5]...[9] deren Zweitselektion (2=Vauss, 6=MD). [10] bis [14] und [15] bis [19] sind die Secondaries, [20] wird nicht verwendet. Gegenmaßnahmen sind geskriptet.

Wie Du sehen kannst, verwenden nicht alle Waffen Fire sound. Die meisten verwenden Gunpoint sound, was bewirkt daß der Sound beim Feuern vom Gunpoint kommt (normalerweise nur einer, auch wenn von mehreren geschossen wird). Fire und Release sound werden für Effekte wie Laden und Releasen der Fusion oder Lock und Release des MD verwendet.

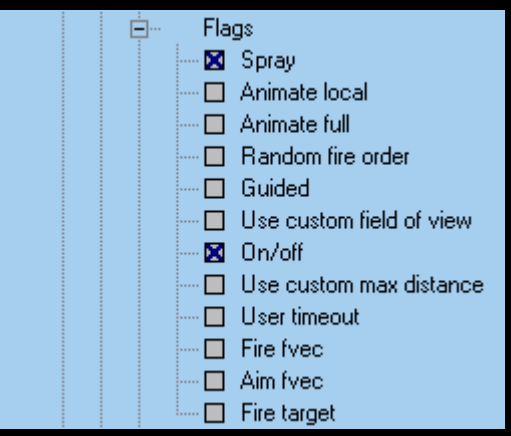


Firing Flags: Öffnen, um Fusion- oder MD-Spezialeffekte zu machen. Hier kann man auch einstellen, daß Munition in Zehnteln gezählt wird.
Spew powerup: Das Powerup daß der gekillte Player von sich gibt.
Max Ammo: Maximale Munitionszuladung der Waffe (in der Weapon battery [Vauss, MD, alle Raketen])
Energy Usage: Wieviel Energie die Waffe vom Spieler zieht.
Ammo Usage: Munitionsverbrauch bei Waffen die Muni verwenden.
Gunpoint Weapons: Hier ordnet man eine Waffe durch ihre Indexnummer einem Gunpoint zu. Dies macht sie nur zur Verwendung ‚verfügbar'. Anmerkung! Wolf On Air hat aufgezeigt, daß dies eventuell ein obsoleter Bereich ist, der zu setzen nicht erforderlich ist. TESTEN!
Firing info: Wo das eigentliche Feuern eingestellt ist (Siehe geöffnete Firing info).
Number masks: Denke von Masken als Salven. Das setzt die Anzahl an Salven, die pro wiederholter Feuersequenz stattfinden.
Aiming flags, -dot, -distance, -XZ dot: weiß nicht ☺
Flags: (siehe geöffnete Flags)
Fire From (quad firing): Hier stellt man ein, von wo gefeuert ist, wenn der QuadLaser aufgenommen wurde. Betrifft nur Weapon[0] außer Du machst ein wenig schweres Skripting.
Gunpoint sounds: Hier setzt man den Sound, den eine Waffe beim Abfeuern von einem bestimmten Gunpoint aus macht. Wird dieselbe Waffe von zwei Gunpoints gleichzeitig abgefeuert, weise nur einem der Gunpoints einen Sound zu.



Weapon slots: Hier weist man eine Waffe per Namen zu einem Gunpoint zu. Dies macht sie nur ‚verfügbar' NOTWENDIG!
Hier stellst Du die Release-Eigenschaften für eine Maske (eine Salve) ein. Du mußt für jede Maske eine Firing info einstellen. Links sind zwei Firing info's die beim Abfeuern der Vauss verwendet werden.
Fire from: Hier stellst Du ein, von welchem Gunpoint diese eine spezielle Salve abgeht.
Wait time: Der Betrag der Zeit der vergeht bis die nächste Salve losgelassen wird. Wenn es die letzte Salve/Maske ist, ist es die Zeitspanne bis zur Wiederholung der Sequenz. Dies verhindert auch, daß die Waffe erneut abgefeuert wird bevor die Zeit abgelaufen ist (Beispiel: MD).
Animations-...: Info für die Animation des Schiffes. Die Animationszeit beginnt mit Abfeuern der Salve. Der Animation fire frame (an welchem Punkt der Animation die Waffe abgefeuert wird) funktioniert NICHT bei Schiffen (aber bei Robots). Egal was Du machst (außer du skriptest entsprechend) wird IMMER bei Frame #1 gefeuert, sogar wenn Frame #1 garnicht in den Animationsframes ist.
Beim Betrachten der Fire info der Vauss sehen wir, daß sie von Gunpoint #2 feuert, 0,1 Sekunden wartet, dann vom Gunpoint #1, dann das ganze wiederholt.

So bekommt der Black Pyro seine Plasma-‚Bursts'. Er verwendet acht Masken mit einer kürzeren Wartezeit als der Standard-Pyro. Die letzte Maske hat die Wartezeit beträchtlich länger eingestellt. Dies bringt ihn dazu acht schnelle Salven, Pause und wieder acht schnelle Salven zu verschießen.



Ich habe dieses Gebiet der Battery nicht getestet, aber wenn man in den Napalmwerfer oder die Omega schaut, bekommt man eine Ahnung um was es geht. Die meisten nicht-animierten Waffen verwenden diese Flags nicht, außer, natürlich, die Guided Missile.

Ich hoffe, diese Tuts haben Dir eine gute Grundlage verschafft, um Deine eigenen Custom-Waffen zu bauen. Man kann mit schwerem Skripting noch mehr machen, aber das ist etwas wo ich bezweifle daß ich jemals Zeit haben werde das zu erkunden.

Viel Glück!

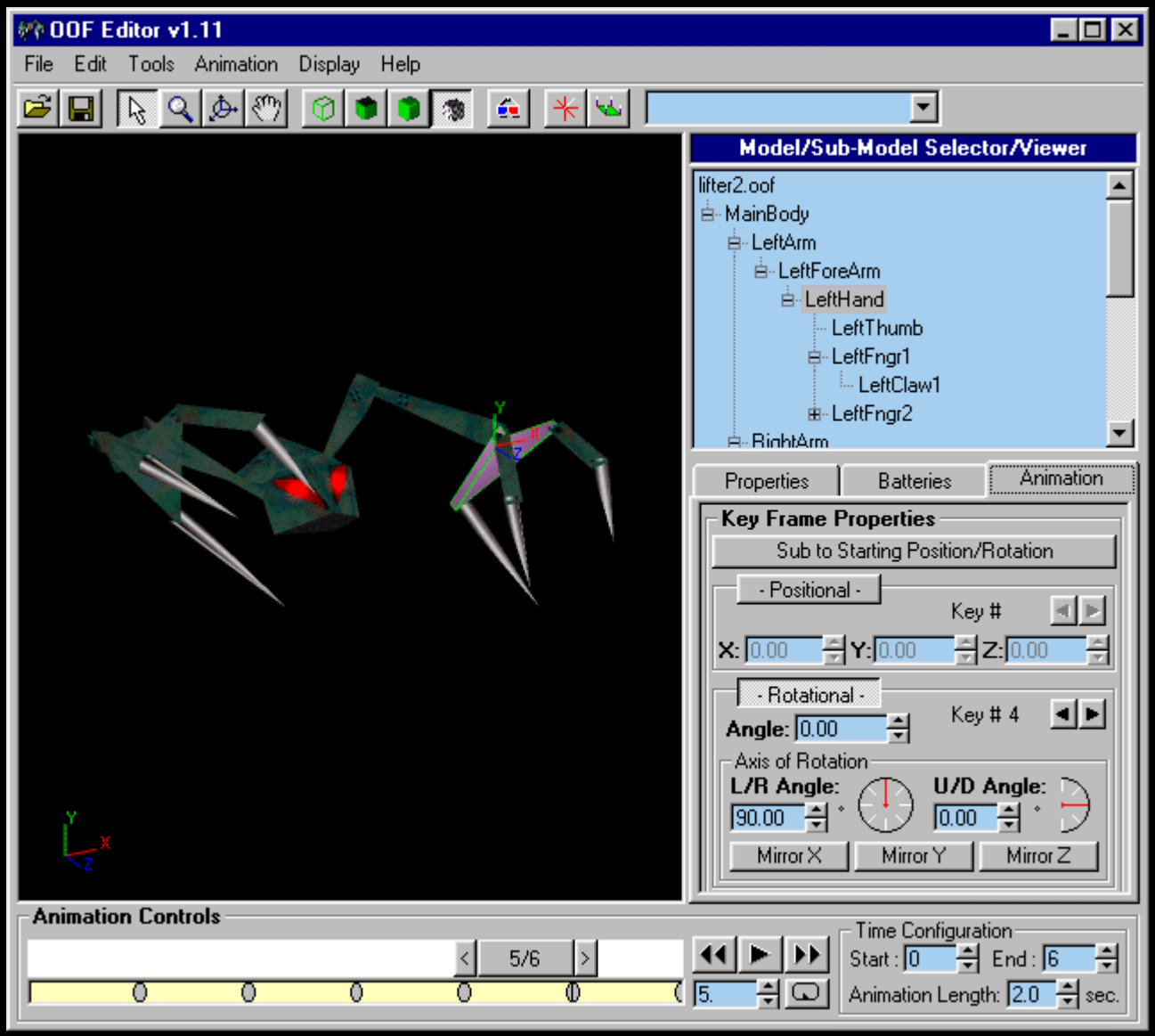
EXKURS

BY PAPAGAT

OOFEditor macht es möglich, komplexe Objekte, inklusive Sub-Models in D3Edit zu bauen. Du brauchst kein teures Modelling-Programm. OOFEdit konvertiert das `.orf` von D3Edit, wo Du das komplette Modell erstellst nach `.oof`, komplett mit den Submodels und in weniger als einer Minute (habe mitgestoppt) bist Du soweit daß Du Eigenschaften einstellen, Gunpoints setzen, Punkte anfügen, Waffenbatterien definieren und das .oof voll animieren kannst. OOFEdit zeigt Dir dein `.oof` auch mit Custom- und animierten Texturen an.

Der OOFEditor gibt Dir nicht nur die Macht Leveldesign an neue Grenzen zu bringen, SuperSheep hat dieses Programm auch mit intuitiven Tools beladen, was es einfach macht sie zu verstehen, lernen und zu verwenden. Denke daran, wenn Du neu beim 3d-Modelling/Animation bist, gibt es Konzepte die Du lernen müssen wirst. Einige der fortgeschritteneren Animationskonzepte können ein wenig trickreich sein. Glücklicherweise macht es der OOFEditor leichter, zu verstehen was man macht und warum.

Hier ist ein Screenie von OOFEditor mit meinem Tierchen, den Lifter ‚Max', als ich ihn animiert habe. SuperSheep war so nett und hat Max als Logo für den Editor genommen.



Von dem Moment an wo ich Max' `.orf` im OOFEditor geöffnet habe, hat es weniger als 20 Sekunden gebraucht um die Sub-Model-Hierarchie komplett zu strukturieren. Weniger als zwei Minuten um alle 16 Pivot-Punkte zu relokalisieren, genau dorthin wo ich sie haben wollte. Die glühenden Augen brauchten jeweils zehn Sekunden - inklusive Farbzuweisung und Adjustierung deren Radii.
Ich war nach 20 Minuten mit der Animation von Max fertig (6 Frames, jeweils 10 Keyframes).

Dach der Installation muß OOFEdit nur wissen, wo Du die `d3.hog` liegen hast. Ist das erst erledigt, kannst Du über die Dropdown-Box rechts oben im Programm jedes beliebige `.oof` von D3 betrachten. Das Programm kann direkt aus der `d3.hog` und `.mn3`-Dateien lesen und extrahieren. Nein, Du kannst nicht direkt in eine `.hog` oder `.mn3` zurückspeichern - ein wenig Arbeit mußt Du schon selbst machen.

Wir werden mit einem einfachen ein-modelligem .oof beginnen, einem Powerup, das die `$rotation`-Eigenschaft verwendet um sich in D3 zu drehen. Danach ein animierter Schalter. Ab da - Robots mit Gunpoints, glühenden Augen und (Thrusters), und soviel Animation wie die Fusion Engine von D3 erlauben wird.

(Papacat rät hier, ein .oof mit SuperSheep's OOFtoORF in ein `.orf` zu wandeln, obwohl OOFEdit ein `.oof` als `.orf` exportieren kann.)

083 - OOFEdit: Terminologie

Papacat

OOF: Outrage Object File - Diese Datei enthält alle Submodelle, Eigenschaften und Animationsinformationen für ein komplettes Modell oder ‚Szene' wie manche 3d-Programme das nennen.

Sub-Model: Ein individueller Teil eines Models wie etwa der Griff eines Schalters oder die Hand an einem Arm. Beachte daß ein Submodell einfach ein einzelnes Face sein kann. In Fakt, Glows und Gunpoints sind solche; ein-facige Submodelle mit speziellen Eigenschaften die ihnen zugewiesen wurden.

Pivot Point: Der Ursprungspunkt eines Submodels. Von diesem Punkt wird die Bewegung bestimmt wenn ein Translations- (Verschiebung) oder Rotationskey zugewiesen wird. Wenn Du einen Arm am Ellbogen abbiegst, ist der Ellbogen der Pivot-Punkt des Unterarms. Das Handgelenk wäre der Pivot-Punkt der Hand. Du kannst den Pivot-Punkt in Relation zu seinem Sub-Model beliebig setzen.

Parents, **Children** und **Silblings**: So beziehen wir uns auf die Relationen unter den Sub-Models die zusammengehängt sind (logisch, nicht physikalisch) Ein Arm ist das einfachste Beispiel: Eine Hand ist Child zum Unterarm. Dieser ist Parent zur Hand. Finger sind individuelle Children der Hand und sind Silblings zu den anderen Fingern dieser Hand (gleicher Parent).

Frame: Animationsinkrement.

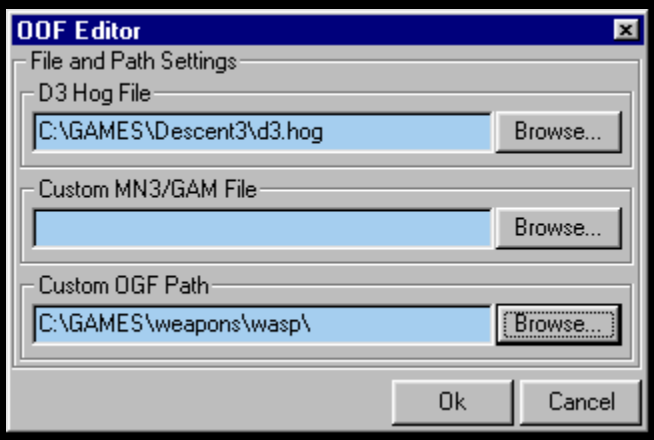
Key: Eine Beschreibung einer Bewegung welche ein Sub-Model innerhalb eines Frames vollführt. Es gibt zwei Typen von Keys: **Translation** und **Rotation**. **Translation** ist die Bewegung von einer X,Y,Z-Koordinate zu einer Anderen, etwa wie ein Kolben der sich nach oben bewegt. Es würde noch einen Translations-Key brauchen, um ihn sich wieder zurückbewegen zu lassen. Ein **Rotation**s-Key rotiert ein Sub-Model um eine Achse um einen bestimmten Betrag an Winkelgraden. Beispiele wären eine Tür die sich in ihren Angeln dreht, ein rotierendes Powerup, oder ein Arm der am Ellbogen abgebogen wird.

GunPoint: Ein Punkt, von dem aus eine Waffe abgefeuert wird oder ein Objekt ausgespuckt wird. Er wird erzeugt, indem man ein Submodel, daß aus nur einem Face besteht zu einem Gunpoint konvertiert.

Glow: Ein Punkt, von dem ein Lichtschein ausgeht.

084 - OOFEdit: Features

Papacat

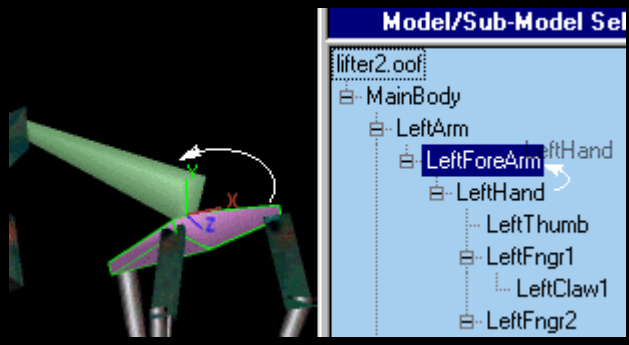


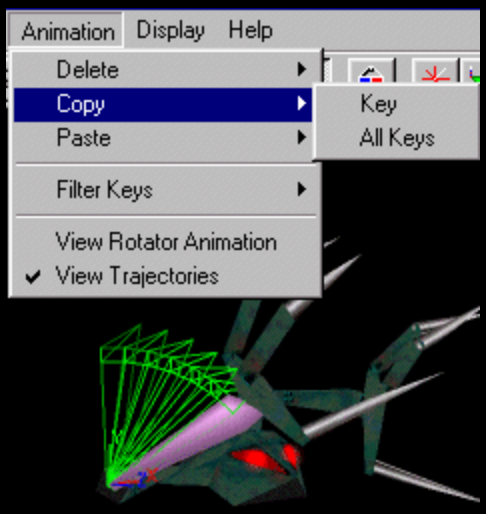
Zeigt man OOFEdit erst einmal wo die d3.hog liegt, zeigt er die enthaltenen Objekte nicht nur mit den D3-Texturen an, sondern Du kannst auf jede .oof aus der d3.hog zugreifen, ihre Animation abspielen oder exportieren.

Wenn du eine Custom MN3 und eine .gam spezifizierst, kannst Du auch auf Objekte zugreifen, die diese Dateien enthalten.

Oder zeige auf einen Ordner der die .ogf's enthält um sie im OOFEditor zur Verfügung zu haben.

Bereite schnell und einfach die Animation Deines .oof vor, indem Du die Sub-Model-Hierarchie per Drag-n-Drop einrichtest. Ziehe einfach das Child auf das beabsichtigte Parent. Du kannst das im Model/Submodel-Selektor/Viewer machen, oder gleich im Display. Hervorhebungen machen es einfach zu bestimmen, was man macht. Das angewählte Child ist purpurn. Das potentielle Parent grün.

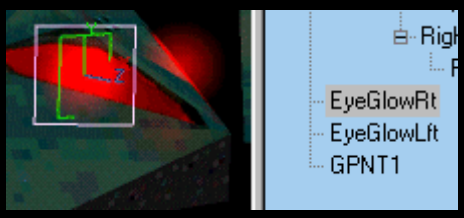




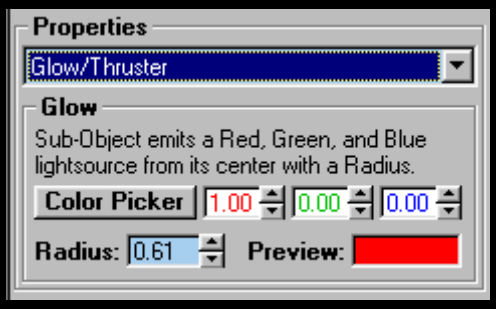
Du kannst Keys eines Sub-Models von einem Frame zu einem Anderen Frame kopieren.
Du kannst sogar alle Keys eines Submodels auf einmal kopieren und sie in ein anderes .oof einfügen.
Schalte View Rotator Animation ein um Rotationseigenschaftseinstellungen zu überprüfen. View Trajectories zeigt Dir den Pfad, den ein Sub-Model in einem Frame nimmt.

Hier ist die Animationskontrollen-Toolbar. Setze Time Configuration für die Frames die Du animieren willst und drücke den Play-Button. Du verwendest sie auch um Frames hinzuzufügen. Der Key-Indikator zeigt Dir ob das gewählte Submodel einen Animations-Key in diesem Frame hat oder ob nicht und wird auch beim Copy/Paste-Prozeß verwendet.

Glows sind nur ein-facige Sub-Modelle mit der Glow-Eigenschaft.



Wähle das Sub-Model und Wähle Glow/Thruster aus der Properties-Liste. Du kannst die Farbe durch Justierung der RGB-Werte einstellen, oder über einen Farbwähler aus einer Farbpalette auswählen. Verwende die Radius, um die Größe zu bestimmen.



085 - OOFEdit: Erste Schritte

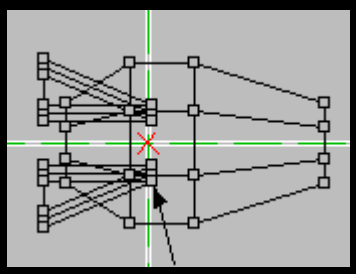
Papacat

Du mußt das GESAMTE Model in D3Edit bauen, bevor Du es in den OOFEdit importierst. Dies inkludiert Faces, die zu Punkten konvertiert werden sollen (Gunpoints, Attach Points, etc...) Aye, dies bedeutet wenn Du später etwas hinzufügen willst, mußt Du das in D3Edit machen und den gesamten `.orf` wieder re-importieren. Die gute Nachricht ist, daß OOFEdit derart einfach ist, daß Du schnell wieder dorthinkommst wo Du 'rausgegangen bist. Ein Hauptpunkt bei der Verwendung von OOFEdit ist gute Vorarbeit in D3Edit.

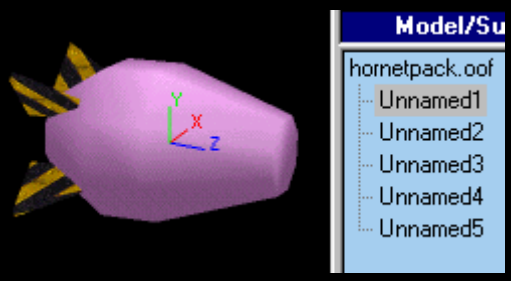
Ein ORF vorbereiten:

1.) OOFEditor bestimmt basierend auf der Verbindung von Vertices, welche Faces ein einzelnes Submodel ausmachen . Dh., wenn Faces an Vertices verbunden sind, werden diese Teil desselben Submodels sein. Das macht es einfach, in D3Edit Submodels zu konstruieren - verbinde sie einfach nicht. Ein gutes Beispiel sind die Finnen auf meinem Hornet-Powerup. Die Finnen wurden seitlich gebaut und dann durch den Hauptkörper gestoßen. Keins der Finnen-Verts ist mit einem aus dem Körper verbunden, sodaß ich insgesamt fünf Submodels bekomme, wenn ich es in OOFEdit öffne. Keine Sorge – wenn Du willst daß sie ein Model sein sollen, kannst Du sie im OOFEditor verkoppeln.

Hier ist die Hornet in D3Edit. Beachte, daß die Verts der Finne nicht mit den Verts des Körpers verbunden sind.

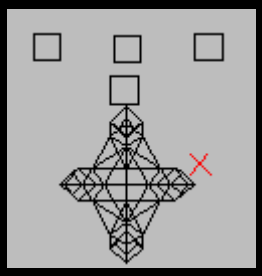


Hier ist die Hornet im OOFEditor. Du kannst fünf Sub-Modelle im Model/Sub-Model-Baum sehen.

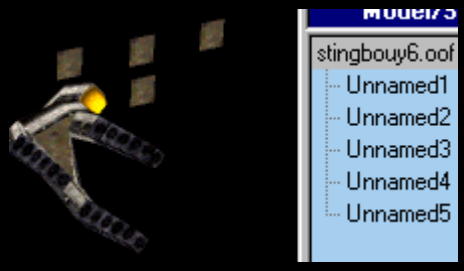


2.) Stelle Sicher daß Du für jeden GP, Glow, etc. die dein `.oof` enthalten soll, unabhängige Faces erstellst.

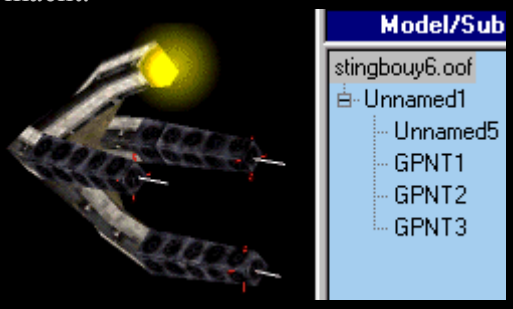
Hier ist die Wasp. Ich habe ein Face für ein Glow und drei für Gunpoints hinzugefügt.



Geöffnet im OOFEdit. Beachte die fünf Submodelle.



Fertiggestellt im OOFEdit. Es hat fünf Minuten gedauert. Ich zeige Dir später, wie man Gunpoints und Glows macht.



3.) Stelle sicher, Das die GESAMTE Texturierung und Texturausrichtung in D3Edit fertiggestellt wurde

4.) Zentriere das Model auf 0,0,0 so wie Du es in OOFEdit willst. Dies beeinflußt den Kollisions-Radius des Objektes. Kein Problem wenn Du's vergessen hast, Du kannst es immer aus OOFEdit heraus zentriern. Du kannst auch die Radiusgröße ändern, wenn Du erstmal im OOFEdit bist.

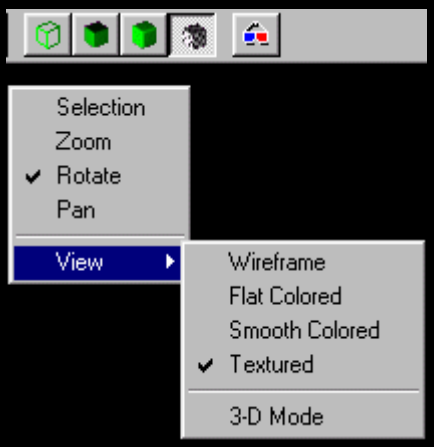
Den .orf in den OOFEditor bekommen

1.) Starte OOFEdit, gehe nach File -> Open OOF/ORF und wähle dein `.orf`.
2.) Verwende File -> Save oder File -> Save as um deinen Raum als `.oof` zu verspeichern. Save erzeugt eine `.oof`-Datei mit demselben Namen wie dein `.orf` im Ordner deines `.orf`'s. Save As läßt Dich noch einen Namen vergeben und den Speicherort bestimmen.
Das wars – Congrats – es ist ein OOF.

Navigation

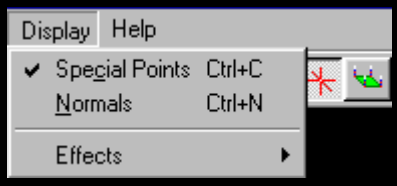
Mit Zoom, Rotate und Pan manövriert man herum. Man wählt sie entweder von der Toolbar oder per Rechtsklick in die Anzeige und Auswahl im folgenden Popup-Menü (von mir bevorzugt). Beides hier rechts.

Ansichtsoptionen



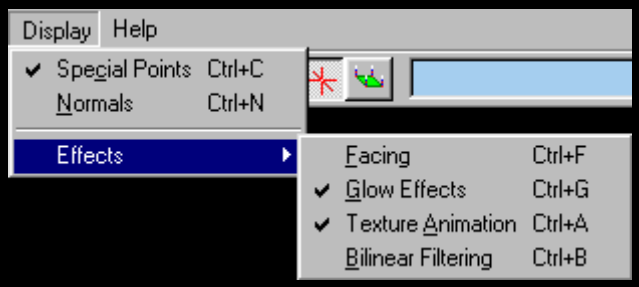
Du kannst die Ansicht als Drahtmodell, Solid, geglättet oder texturiert wählen indem Du den entsprechenden Knopf in der Toolbar drückst oder die View-Option im Kontextuntermenü wählst. Wenn Du Anaglyphengläser hast, klicke auf die Brille.

Schalte ‚Special Points' (Gunpoints, etc...) und Normals an oder ab indem Du das Display-Menü oder die Toolbar verwendest.

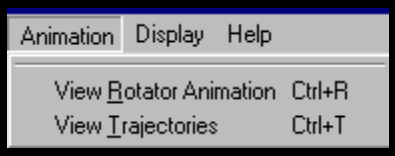


Effekte

Aktiviere deren Anzeige über eine Auswahl unter Display -> Effects.

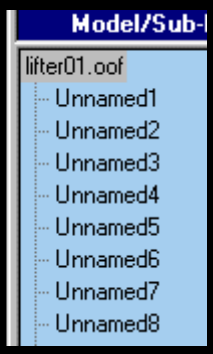


Um die Rotation von Objekten mit der $rotation-Eigenschaft zu sehen gehe nach Animation -> View Rotator Animation.



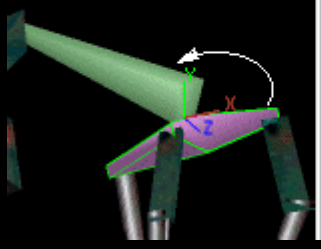
Parent/Child-Beziehungen und Vereinigung

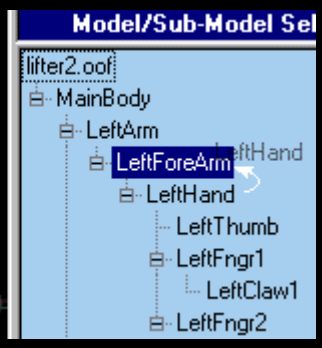
Verwechsle das nicht miteinander. Obwohl logisch miteinander verbunden, sind Parents und Children immer noch separate Sub-Models. Wenn Du zwei Sub-Models vereinigst, werden sie zu einem Submodel. Es gibt kein Undo für Combine, also speichere oft.



Wenn die `.orf` frisch importiert wurde, sind die Sub-Models unabhängig voneinander.

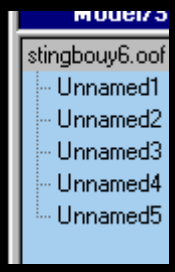
Drag/Drop ein Submodel von dem Du willst daß es ein Child wird auf jenes, dessen Kind es werden soll. Dies geht im Model-Tree als auch in der Anzeige. Hervorhebungen machen es einfach zu kontrollieren was man tut. Das gewählte Kind ist purpurn, das potentielle Parent grün.





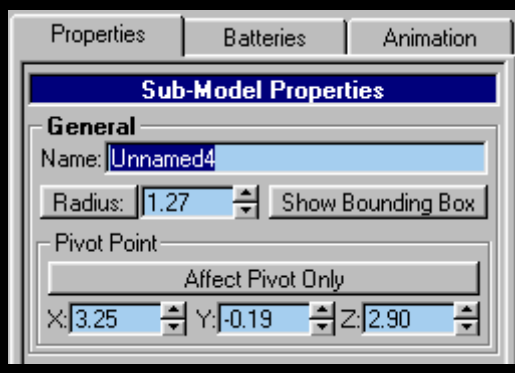
Um Submodels zu vereinigen, halte Shift während des Drag/Drop.

Gunpoints

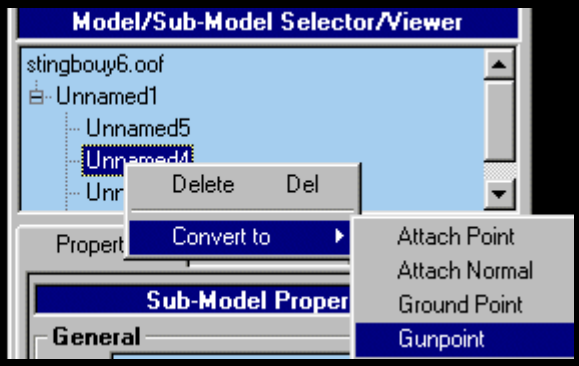


Gib Deinem `.orf` unabhängige Faces zum konvertieren in Gunpoints mit.

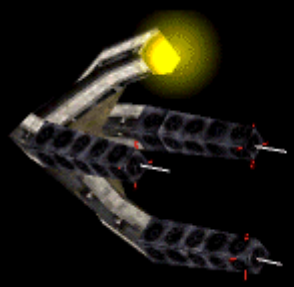
Verwende die X, Y und Z-Wähler um sie in Position zu bringen. Du kannst sie allerdings auch in D3Edit positionieren.



Um konvertiert zu werden, muß ein Sub-Model erst Child eines Anderen sein. Verwende die zuvor beschriebene Methode dafür.
Um ein Submodel zu konvertieren, rechtsklicke auf es in der Baumansicht und wähle Convert to -> Gunpoint aus dem Menü.



Fertig.
So einfach ist das.

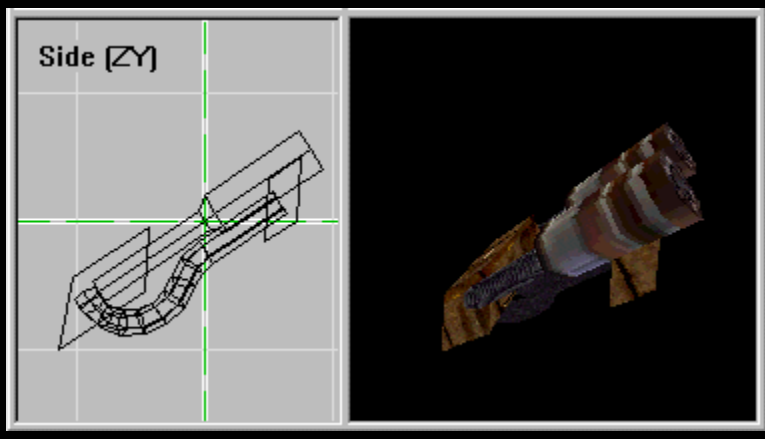


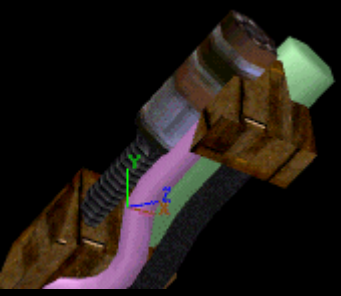
Vergiß nicht, daß D3 einen Gunpoint erst verwenden kann, wenn er zu einer Battery gehört. Gehe in die Batteries-Tab. Klicke **Add Battery**, die verfügbaren Gunpoints werden aufgelistet. Markiere einen Gunpoint auf der Available-Seite und klicke den >-Button. Genauso würde man auch Batteries für Turrets machen.

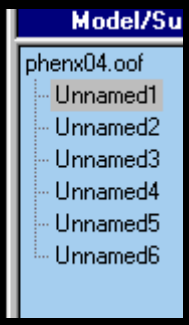
086 - OOFEdit: Ein Powerup

Papacat

Hier ist eine Kanone, die ich als Powerup verwende.
Ich habe die Modellierarbeit in D3Edit abgeschlossen, Texturierung eingeschlossen. Das Modell besteht aus sechs verschiedenen Teilen, die zusammengebracht aber nicht an Verts verbunden sind.
Ich habe es geneigt und so ziemlich zentriert wie ich es will. Ich hätte es auch in OOFEdit adjustieren können, aber das geht auch.

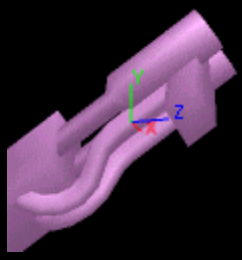


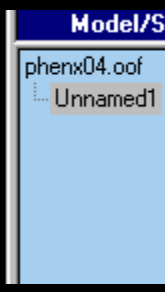




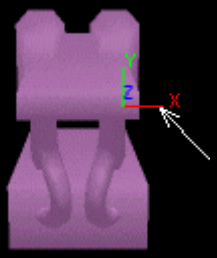
Wenn ich es öffne sind da sechs Sub-Models. Halte **Shift** nieder, während Du alle Submodelle auf ‚Unnamed1' drag-dropst.

Dies vereinigt sie in ein einziges Submodel. (rechts)

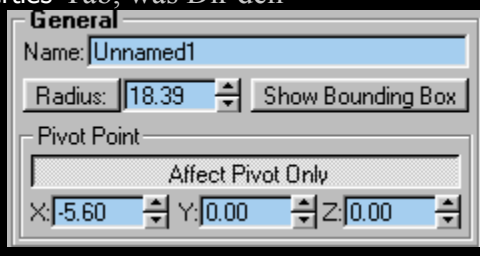




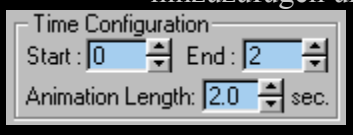
Es ist jetzt ein guter Zeitpunkt zu SPEICHERN. Es gibt kein Undo. Deine beste Verteidigung vor selbstzugefügten Verletzungen ist's oft zu speichern und multiple Kopien anzulegen.



Ich bemerkte, daß ich das `.orf` nicht korrekt zentriert hatte bevor ich es importiert habe. Links siehst Du den Pivot-Punkt für das Submodel, hier entspringen die Achsen um die das Objekt rotiert. Er ist einfach zu verschieben. Wähle die Properties-Tab, was Dir den Sub-Model-Eigenschaften-Dialog anzeigt. Klicke auf **Affect Pivot Only**. Wenn Du das nicht machst, wird sich das gesamte Submodel bewegen. Justiere die X-Achse indem Du auf den Spinner klickst. Du kannst einen der dreieckigen Pfeile klicken/halten, oder zusätzlich die Maus rauf oder runter bewegen.

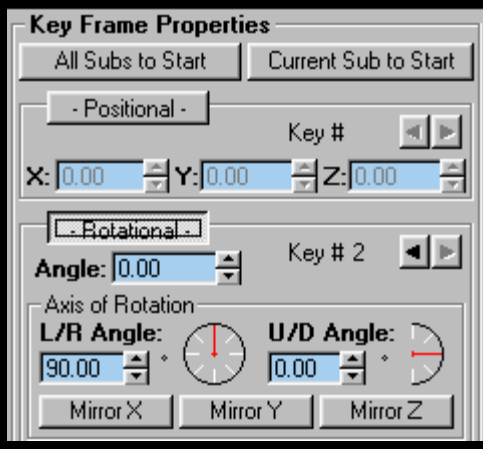


Wenn Du erst einmal den Pivot-Punkt dort hast wo Du ihn haben willst, ist es Zeit Frames hinzuzufügen und diesen Keys zuzuweisen.

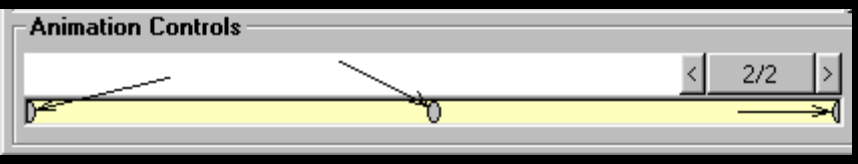


Zuerst brauchen wir Frames, welche die Keys enthalten werden. In der unteren rechten Ecke des Programms siehst Du die Time Configuration, die Zeitablaufkontrolle. Laß Start auf 0 und ändere End auf 2. Du wirst sehen, daß der Frame-Marker nun 0/2 anzeigt. 0 ist der aktive Frame, die 2 ist die Gesamtzahl an vorhandenen Frames.
Nun füge Animations-Keys zu den Frames hinzu. Diese sagen, wie sich das Submodel bewegen soll. Es gibt zwei Arten: Rotational und Positional. Auch Rotate und Translate genannt. Wir werden Rotational Keys verwenden.

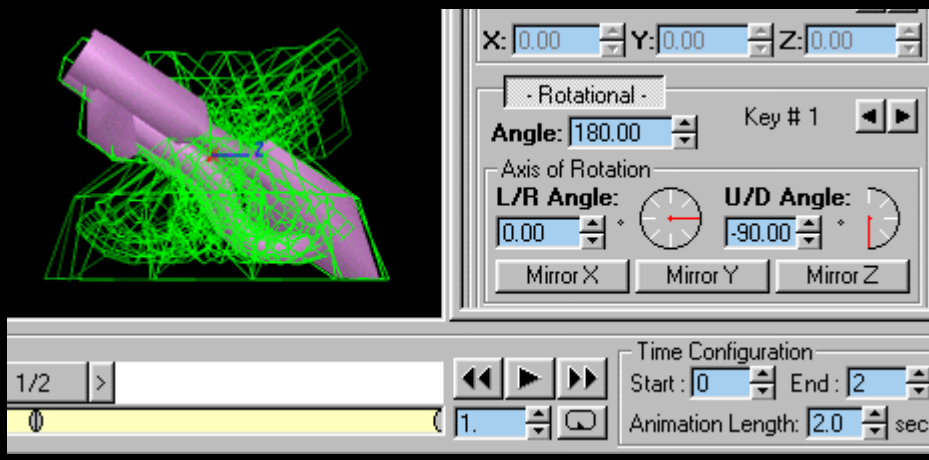
Als erstes gehe nach Key Frame Properties in der Animation-Tab. Wähle das Submodel, während der Framemarker auf 0 steht. Klicke **Rotational**.



Du siehst wie ein Oval in die Framebar eingefügt wird. Das heißt, das ausgewählte Sub-Model hat einen Animations-Key in diesem Frame. Bewege Dich zu Frame 1 und 2 und füge einen Rotational Key zu jedem Frame hinzu.



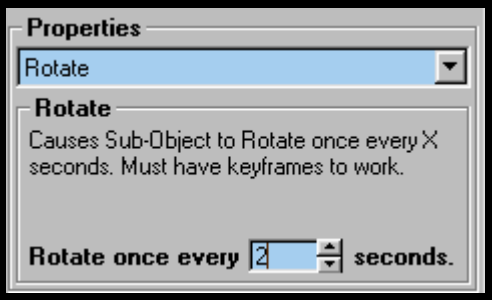
Bevor Du irgendwas anderes machst, gehe nach Animation im Progammenü und schalte View Trajectory ein. Setze den Framemarker auf 1. Der Angle (=Winkel) ist der Betrag der Rotation, den das Objekt vollführen wird. L/R (Links/Rechts) und U/D (Rauf/Runter)-Angle setzen die Achse, um die das Objekt rotieren wird. Du kannst L/R und U/D entweder über die Spinner oder durch ziehen der Winkelanzeiger (rote ‚Zeiger') justieren. Wenn Du **Shift** während des Ziehens betätigst, bewegen sie sich in 45°-Inkrementen. Du kannst auch direkt einen Wert eingeben. Setze die Winkel wie rechts gezeigt.



Schiebe den Framemarker weiter zu Frame 2 und gib die selben Einstellungen an. Der grüne Anriß den Du siehst ist die Trajektorie. Er zeigt Dir den Weg den das Submodel beim Bewegen nehmen wird. Speichere, damit nichts verloren geht.

Spielen wir die Animation ab, um zu sehen was unser `.oof` bis jetzt macht. Drücke den Knopf mit der gezeichneten Schleife (gleich links neben der Time Configuration), das ist die Loop-Kontrolle. Drückst Du dann Play, wird die Animation kontinuierlich wiederholt. Drücke Play. Dein `.oof` sollte sich drehen, während die Trajektorienspur angezeigt wird. Du kannst diese im Animation-Menü abschalten wenn Du willst.

Während das Submodel noch immer selektiert ist, gehe in die Properties-Tab und wähle Rotate aus der Dropdown-Box. Wenn Du das machst, siehst Du darunter Information über Rotate und eine Stelle, wo Du die Rotationsgeschwindigkeit einstellen kannst. Setze sie auf zwei Sekunden.



Speichern

Du hast fertig.

Mache eine `.gam`, die dieses `.oof` als Powerup verwendet. Durch die Eigenschaft Rotate mußt Du nicht DALLAS skripten oder Initial rotation in der `.gam` setzen. Setze nur die Use physics-Flag in der `.gam`. Wenn wir die Rotate-Eigenschaft nicht setzen, würdest Du ein Event skripten müssen um die Animation abzuspielen.

File: 087-OOFEditTurret.zip

087 - OOFEdit: Turrets

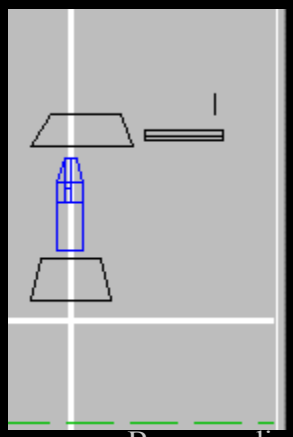
Papacat

Auch hier war die originale .zip mit den Übungsdateien nicht mehr auffindbar. Der von mir nachgebaute Turret ist ein wenig unterschiedlich von dem hier gezeigen, aber es funktioniert alles gleich.

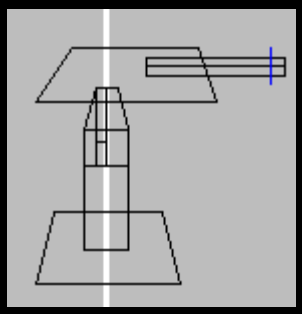
Geschütztürme sind die am leichtesten zu bauenden/animierenden ‚animierten Bots'. Dies deswegen, weil Du die Animation nicht erstellen mußt. Alles was Du zu machen hast, ist die Turret-Eigenschaft zuzuweisen und ihre Einstellungen zu setzen. Ein paar Regeln müssen beim Konfigurieren der Submodels befolgt werden. Hier lernst Du, wie Du Dein `.oof` erfolgreich dazu bringst die Turret-Eigenschaft zu verwenden.

Hier ist ein simpler Turret, in D3Edit gebaut:

Ich habe fünf separate Teile. Der Kopf und der Ständer werden die zwei sich bewegenden Teile sein. Ich werde den Lauf separat halten, für den Fall daß ich eine Animation für den Rückstoß einbauen will (werden wir hier nicht machen). Hier ist ein unabhängiges Face, daß ich zum Gunpoint konvertieren werde (Denke daran - plane voraus).

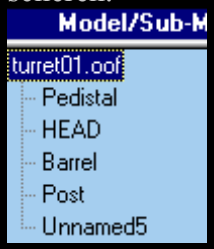


In D3Edit stelle ich sicher, daß das Modell so zusammengesetzt ist wie ich das will.

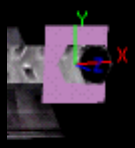


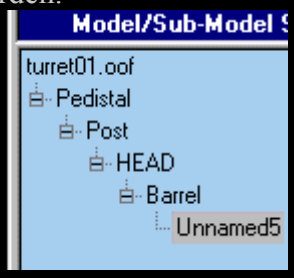
Beim Öffen in OOFEdit siehst Du die fünf Submodels im Tree.

Benenne die Submodels; um jenes das zum Gunpoint wird, brauchst Du dich nicht scheren.

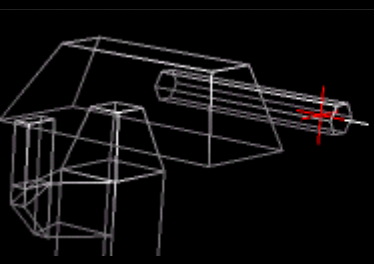


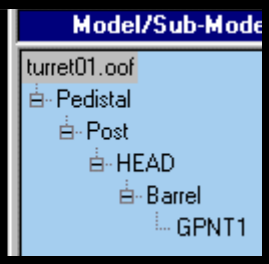
Konfiguriere die Submodel-Hierarche.
Rechsklicke auf das Face und konvertiere es in einen Gunpoint. Gunpoints bekommen eine Enumeration mit GPNT#, in der Reihenfolge wie sie erstellt werden.

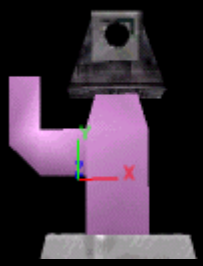


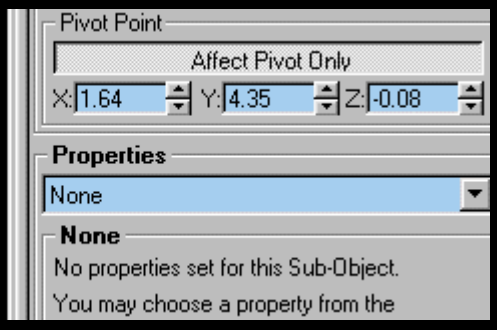


Dieser Baum zeigt einen der wichtigsten Punkte beim Bau eines Geschützturmes. GPNT1 ist nicht nur das was die Waffe feuern läßt, sondern auch der ‚Aiming Gunpoint' (in der `.gam` spezifiziert.) Er MUSS hierarchisch unter den Submodels liegen, die sich bewegen. Wenn GPNT1 zu zielen versucht, sagt er den Submodels DARÜBER daß sie sich bewegen sollen. Er kann Child zum Lauf oder Kopf sein, solange er dem Kopf untergeordnet ist.

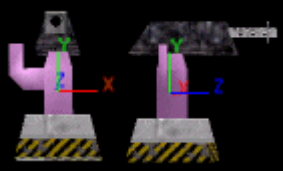




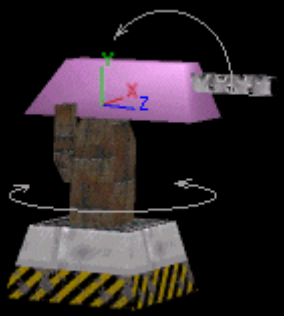




Ein Sub-Model rotiert um individuelle Achsen, welche von dessen Pivotpunkt entspringen. Stelle sicher, daß diese Pivot-Punkte dort sind, wo Du sie willst.
Um diese zu bewegen wähle das Submodel, klicke auf **Affect Pivot Only** und justiere seine X/Y/Z-Werte.

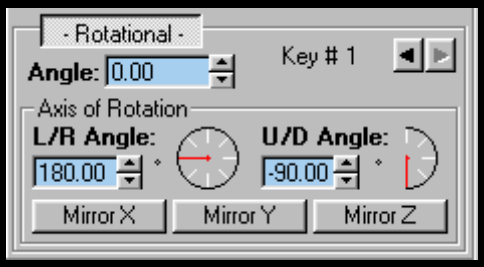


Anfangs sagte ich, daß wir keine Animation machen müssen. Damit habe ich ‚eigentliche' Bewegung gemeint. Wir werden den Subs NICHT sagen daß sie sich bewegen, wir werden nur die Keys für D3 und die Turret-Eigenschaft, die sie verwenden, konfigurieren.

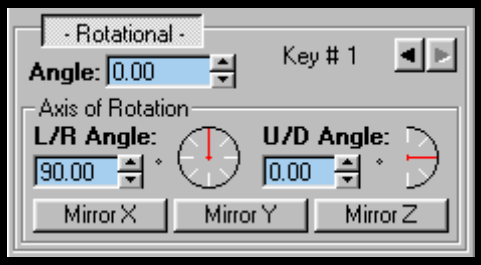


Der Kopf wird rauf/runterrotieren (um die X-Achse). Der Ständer wird links/rechts rotieren (um die Y-Achse). Um die Turret-Eigenschaft zu verwenden, müssen sie einen Rotations-Key in den Frames 0 und 1 haben. D3 guckt in den Frame 1 um zu erfahren um welche Achse rotiert wird.
Gehe in die Animaton-Tab während die Sub hervorgehoben ist und der Framemarker bei 0. Setze den **Rotation**-Button. Daraufhin erscheint ein Oval in der Animationbar. Mache das für die Frames 0 und 1, für den Kopf und den Ständer.

Wähle den Ständer und setze den Framemarker auf 1/1. Setze die Rotationsachse des Rotational-Keys wie unten gezeigt. Hier sieht die Turret-Eigenschaft nach der Achse. Diese Einstellungen machen die Achse vertikal (=parallel zur Y-Achse.)

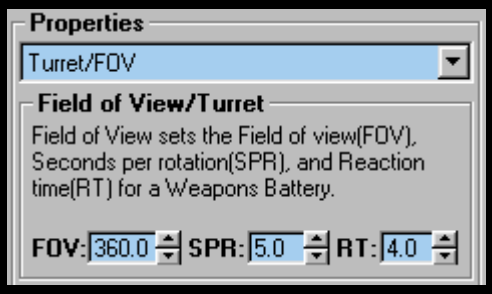


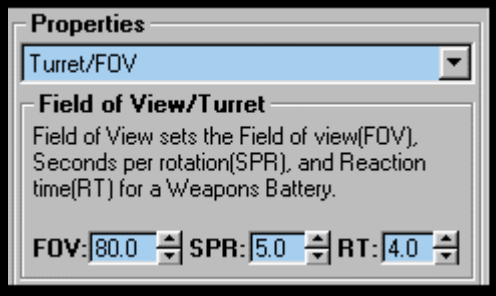
Setze den Rotation-Key für den Kopf wie hier gezeigt. Dies macht seine Achse parallel zur X-Achse.

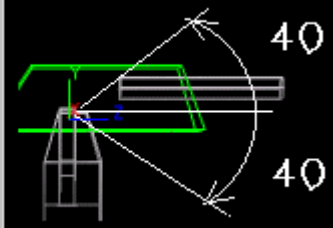


Beachte, daß wir Angle nicht angerührt haben. Wenn wir das würden, hätten wir eine ‚eigentliche' Bewegung definiert.

Während der Ständer hervorgehoben ist, gehe in die Properties-Tab. Setze seine Properties auf Turret/FOV. Die FOV setze auf 360; das ist eigentlich der Bereich der Bewegung. Durch Setzen auf 360 wird er dazu fähig sein, einen kompletten Kreis zu beschreiben. Er wird sich ebenso weiterdrehen, anstatt sich zurückdrehen zu müssen.







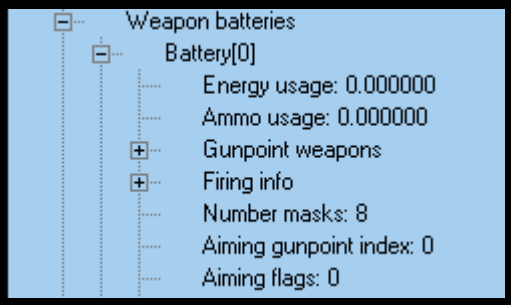
Setze die Eigenschaften für den Kopf so wie hier links gezeigt. Die 80 FOV limitieren den Bewegungsbereich auf 80°, 40° in jede Richtung.

Nun die Konfiguration der Batterie:

Gehe in die Batteries-Tab. Klicke **Add Battery** um eine solche zu erzeugen. Linksseitig sind alle verfügbaren Gunpoints und Submodels.

Während Current Weapons Battery auf 1 gesetzt ist wähle jedes Submodel und den Gunpoint, uund füge sie zu Assigned hinzu. Die Submodels und der Gunpoint müssen alle in der selben Battery sein. Speichern, und Du bist mit der `.oof` fertig.

Nun erzeuge eine `.gam`-Datei. Da sind ein Haufen Einstellungen mit denen Du das Verhalten des Turmes kontrollieren kannst. Zwei Haupteinstellunge werde ich herausstreichen, damit dein Turret ordentlich funktioniert. Der erste ist unter AI Settings. Stelle sicher, daß dort Type auf Stationary Turret gesetzt ist.



Der andere findet sich in Weapon batteries. Erinnere Dich daran, daß die .gam bei 0 zu nummerieren beginnt, nicht 1. Gehe also in die Battery[0].
Setze den Aiming gunpoint index auf 0. Im OOFEditor war das GPNT1. Dieser ist der Gunpoint, der zum zielen verwendet werden wird. Er muß teil der Battery sein, und wie bereits erwähnt, er MUSS hierarchisch unter den Submodels mit der Eigenschaft Turret stehen. Für mehr Info über das Konfigurieren einer Roboter-Waffenbatterie schaue unter ‚Custom Robots Part 1' nach.

Der einfachste Weg eine .gam aufzusetzen, ist die GAM-Seite eines der Original-Türme von D3 zu kopieren. Wenn Du das machst, sind da einige Sachen auf die Du aufpassen mußt. Hauptsächlich achte darauf daß die GAM nicht auf Gunpoints referenziert, die in DEINEM `.oof` nicht existent sind, D3 wird beim Versuch einen nicht-existenten Gunpoint zu verwenden abstürzen. Gehe auch durch die Firing info deiner Weapon battery und prüfe, daß keine Animation verwendet wird, wie etwa vom Lance-Turret (der mit den Homings).

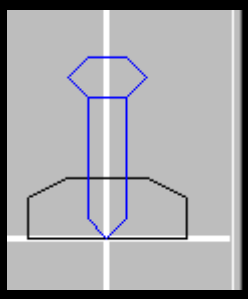
Das war's. Bau den Turm in Dein `.d3l`, dann mache die `.mn3` und vergiß nicht die `.gam` und das `.oof` einzubinden.

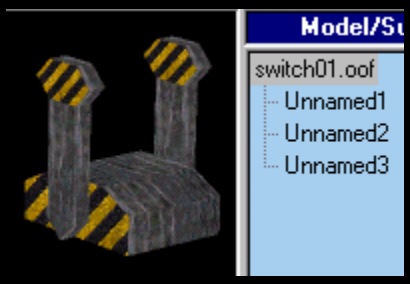
Dank an Garner1 der mir beim Anfangen mit Turrets geholfen hat.

088 - OOFEdit: Animation

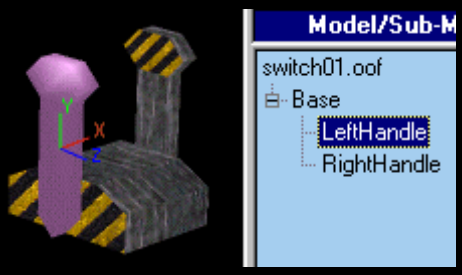
(Papacat)

Dieses Objekt wird ein Schalter mit zwei beweglichen Griffen sein (die `switch1.orf` ist in der `.zip`). Ich habe eine Basis und zwei Schaltgriffe in D3Edit gemacht. Vergiß nicht, Du kannst in OOFEdit keine Teile HINZUFÜGEN, also stelle sicher daß alles was Du brauchst im `.orf` ist.

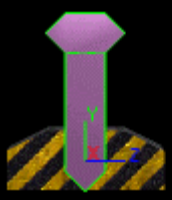


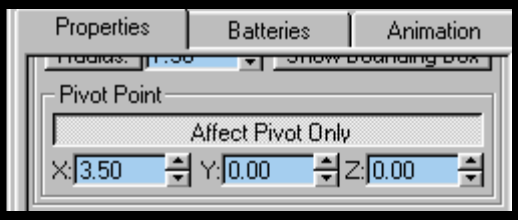


Ich öffne die .orf im OOFEditor und die drei Teile werden aufgelistet. Du kannst sie unbenannt lassen, aber es ist leichter wenn man sie benennt. Dazu wähle es an und ändere in der Properties-Tab den Wert im Name-Feld. Dann mache die Griffe zu Childs der Basis. Es ist zwar für das Tut nicht nötig, aber es ist gute Praxis.



Das Koordinatenkreuz ist der Pivot-Punkt. Wir müssen den Pivot-Punkt von LeftHandle dorthin schieben, wo wir die Rotationsachse entspringen lassen wollen. Während LeftHandle ausgewählt ist, gehe in die Properties-Tab und klicke **Affect Pivot Only**, dann justiere die Y-Einstellung sodaß der Pivot dorthinkommt wo Du willst. Nun ist ein guter Zeitpunkt zu speichern.





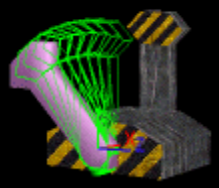
Um die Animierung eines Objektes zu verstehen mußt Du vier Begriffe verstehen: Frame, Key, Rotation und Translate. Gegebenenfalls schaue unter ‚feld gibts no net' nach.

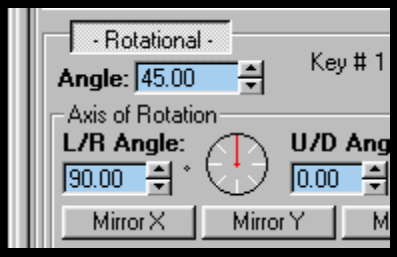
Zuerst brauchen wir Frames, wo unsere Animation stattfinden kann. Unten im rechten Eck, bei der Time Configuration, lasse Start auf 0 und setze End auf 3. Du wirst sehen wie sich der Framemarker auf 0/3 ändert. Das bedeutet, der Marker steht auf 0 und der letzte verfügbare Frame ist 3. Du könntest denken , daß dies vier Frames ergibt. Denke so davon: Ein gleicher Betrag an Zeit existiert zwischen 0-1, 1-2 und 2-3. Wenn wir mehr Keys hinzufügen, wird das klarer werden.

Bei selektierter LeftHandle und dem Framemarker bei 0 gehe in die Animation-Tab und wähle **Rotational**. Dies sagt der `.oof`, daß dieses Submodel eine Rotationsanimation bei Frame 0 hat. Du siehst links in der Animation-Bar ein Oval auftauchen. Dieses sagt Dir, daß das selektierte Submodel hier einen Animation-Key hat. Führe den Framemarker mit > (In der Animation-Tab) weiter, sodaß er nun 1/3 anzeigt. Setze hier einen Rotational Key. Schließe die LeftHandle damit ab, daß Du in den Frames 2/3 und 3/3 auch Rotational Keys setzt. Bei selektiertem LeftHandle sollte Deine Animations-Kontrolle nun so aussehen:

Ein wichtiger Punkt ist, daß ein Key nur für ein Submodel gilt. Selektiere RightHandle: all die Ovale verschwinden. Sie repräsentieren die existierenden Keys für das *aktive Submodel*. Weise Rotational-Keys an allen vier Stellen zu, so wie wir es für LeftHandle gemacht haben. Speichern.

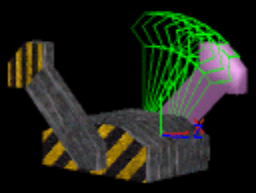
Zeit, die Submodels zu bewegen. Als erstes, damit wir den Weg sehen können den die Submodels zurücklegen werden, schalte View Trajectory entweder über die Animation-Tab oder über das Menü an.
Markiere LeftHandle und setze den Marker auf 1/3. Wenn Du ein Submodel an einem Keyframe bewegst, sagst Du wo es sich am Ende dieses Frames hinbewegt haben soll. Die Bewegung beginnt bei 0/3 und erreicht die Position, die wir bei 1/3 einstellen.
Angle ist der Betrag der Rotation in Grad, um den Du das Submodel gedreht haben willst. Unter Angle siehst Du Axis of Rotation. Indem Du den L/R Angle (Links/Recht) und den U/D Angle (Rauf/Runter) setzt, definierst Du eine Achse, welche dem Pivot-Punkt entspringt. Es ist diese definierte Achse, nicht die X-, Y- oder Z-Achse um die Dein Submodel sich drehen wird.

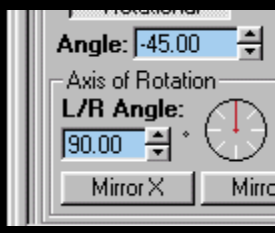


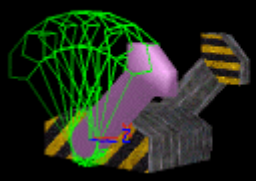


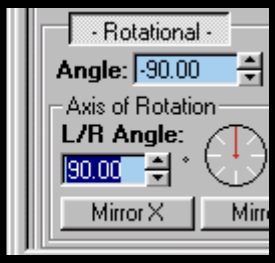
Bei L/R=0 und U/D=0 liegt die Rotationsachse auf der Z-Achse.
Wir wollen die Rotationsachse parallel zur X-Achse. Sie um 90° zu verdrehen erledigt das. Nun Rotiere das Model um 45°. Schreibe es rein und **Enter** oder klicke die Spinner-Pfeile und ziehe mit der Maus. Wenn Du die Methode mit dem Ziehen verwendest, siehst Du wie es sich bewegt und die Trajektorie nachgezeichnet wird. Es sollte so aussehen wie hier oben.

Während der Framemarker noch auf 1/3 ist wähle den RightHandle an und rotiere ihn um -45° (minus 45°).



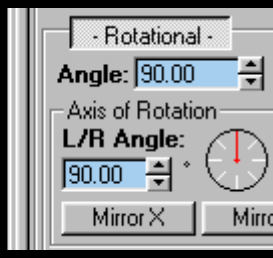






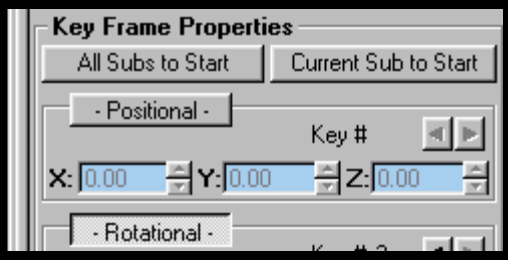
Bewege den Framemarker nach 2/3.

Wähle LeftHandle (links), rotiere es mit -90°.
Wähle RightHandle und rotiere mit 90° (rechts).



Jetzt werden wir beide Griffe wieder in ihre Ausgangspositionen bewegen. In diesem Beispiel ist das recht einfach, aber stelle Dir vor wir hätten drei- oder viermal bewegt und jedesmal den Achsenwinkel verändert. Glaubst Du, daß Du den exakten Winkel der Achse und den Betrag der Rotation herausfinden könntest, um zum Anfang zurückzugelangen? Vielleicht nach einer halben Stunde mathematischer Projektionen. Glücklicherweise hat SuperSheep dem OOFEditor ein Werkzeug mitgegeben, das die ganze Arbeit für Dich erledigt.

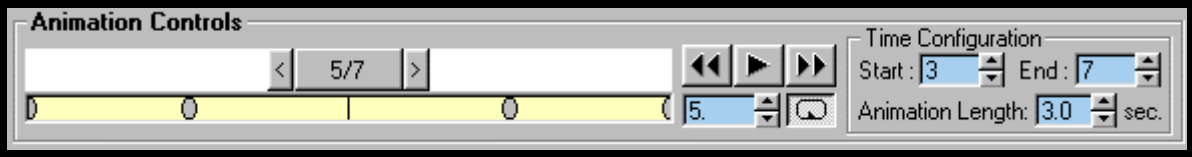
Mit dem Framemarker auf 3/3 klicke auf **All Subs to Start**. OOFEditor berechnet alle nötigen Keys und wendet diese auf das Submodel an, um alles wieder auf Originalposition zu bringen. An und für sich müssen wir keine Keys in 3/3 setzen. Die **All Subs to Start** erledigt auch das.
Speichern!



Testen wir die Animation. In der Time Configuration setze die Animationslänge auf drei Sekunden. Start und End sollten noch immer 0 und 3 sein. Bewege den Framemarker zurück auf 0 und klicke den Play-Button.
Hast Du bemerkt daß sich die Griffe im zweiten Frame (1-2) schneller bewegt haben? Alle Frames sind gleich lang. Im zweiten Frame müssen sich die Griffe doppelt soviele Grade in der selben Zeit drehen, sodaß sie sich schneller drehen um ihre Bewegung in der zugeteilten Zeit zu vollführen. Der bis jetzt fertiggestellte Schalter ist `switch01.oof`.

Nun werden wir die Animation so konfigurieren, daß di eGriffe sich mit gleichbleibender Geschwindigkeit bewegen. Zur selben Zeit werden wir sehen, wie Frames und Keys zueinander in Beziehung stehen.
Ein offensichtlicher Weg wäre die Griffe sich in die erste Position bewegen zu lassen, dann zurück zum Anfang, dann zur zweiten Position: immer 45° per Frame. Dies würde einen Key in jedem Frame erfordern, und da es offensichtlich ist würdest Du nichts lernen. Laß uns was anderes versuchen.

Als erstes gehe in die Time Configuration und setze Start auf 3 und End auf 7. Steht der Framemarker ganz links, sollte da 3/7 draufstehen. Wähle LeftHandle. Du siehst den Farker immer noch auf 3, ganz links. Füge einen Rotational Key für beide Griffe in den Frames 4 und 6 hinzu. KEINEN auf 5. Bei 4 setze die Animationen beider Submodels auf 45 und -45 Grad, gleich wie zuvor. Überspringe 5. Bei 6 setze beide Submodels auf -90 und 90, ebenso wie zuvor. Bei 7, wo noch keine Keys sind, wähle **All Subs to Start**. Ein Key wird für beide Submodels hinzugefügt und deren Animation von OOFEdit gesetzt. Stelle Deinen Framemarker auf 5 und Deine Animations-Bar sollte so wie hier aussehen. Es sollte KEIN Key bei 5 sein. Stelle den Framemarker auf 3, setze den Loop-Button und spiele die Animation ab.



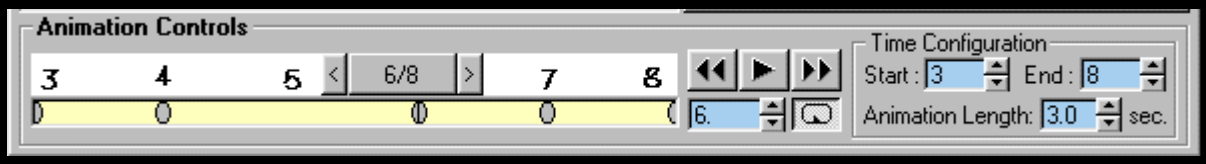
Nun ändert sich die Geschwindigkeit der Griffe nicht mehr. Vielleicht ist es für Dich einfacher zu erkennen, wenn Du View Trajectory momentan ausschaltest. Der fertige Schalter ist `switch02.oof`.

Diese Beispiel zeigt uns zwei Dinge:
1) Animation wird von Key zu Key berechnet, nicht über einen Frame. Dieser ist nur ein Zeitinkrement.
2) Animation und Animationsschleifen können bei jedem beliebigen Frame beginnen und bei jedem beliebigen Frame enden.

Hier ist der Vorteil den wir aus #1 ziehen: Sagen wir Du hast einen Bot. Du willst daß sich sein Kopf um 180° dreht. Während dieser sich dreht, willst Du daß der Bot seinen Arm ausstreckt, etwas greift und zurückzieht. Dies passiert während drei Frames (0 bis 3). Da Animation von Key zu Key berechnet wird, ist alles was wir für den Kopf machen Keys bei 0 und 3, bei 3 die Rotation auf 180° setzen. Dann gehen wir zurück und setzen Keys bei 0, 1, 2 und 3 für den Arm und setzen für jeden die Bewegung. Wenn die Animation bei jedem Frame berechnet werden würde, würden wir Keys bei den Frames 1, 2 und 3 für den Kopf benötigen, die ihn um jeweils 60° drehen.

Darum brauchen wir auch eine Key ganz am Anfang, sodaß es weiß wann mit der Bewegung zu beginnen ist. Wir brauchen auch einen, um ein Submodel vom Bewegen abzuhalten. Ich meine nicht daß Du eine Key bei jedem Sub brauchst, um es stillstehen zu lassen. Was ich meine, ist daß Du einen Key brauchst um es sich erneut bewegen zu lassen (ein wenig verwirrend).

Hier ist ein Beispiel.
Wir verwenden diese Animation-Bar als Bezug:



Von 3 nach 4 soll unser Bot den Kopf um 30° drehen. von 4 nach 5 und 5 nach 6 soll er den Kopf stillhalten. von 6 nach 7 soll er seinen Kopf um 120° drehen. Um das zu erreichen, setzen wir einen Rotational Key bei 6 und lassen den Angle bei 0. Wir setzen einen Key bei 7 und stellen den Winkel auf 120° ein. Wenn die Animation bei 4 ankommt, schaut sie nach was sie auszuführen hat. Sie finded den Key bei 6, der ihr sagt daß sie um 0° drehen soll, daher bewegt sich der Kopf zwei Frames lang nicht. Kommt es zu Frame 6, schaut sie nach vorne und findet 7, welcher ihr sagt daß sie 120° rotieren soll und einen Frame dafür hat. Ohne den Key auf 6 hätte die 120°-Rotation schon bei 4 begonnen und sich über drei Frames erstreckt.

Achtung! Deswegen mußt Du auch einen Key haben der nix tut. Ohne einen Startpunkt passiert nichts.

Nun zu #2: Animation und Animationsschleifen können bei jedem verfügbaren Frame beginnen und enden.
Das heißt, ein Bot kann ein großes Repertoire an Verhaltensweisen haben. Ohne das würdest Du verschiedene `.oof`'s für jedes Verhalten brauchen. Öffne das `.oof` vom ‚Old Scratch' (`scratch.oof`), und laß seine Animation ein paar mal laufen. Du wirst bemerken, daß er immer wieder zur Startposition zurückkehrt. Scratch hat mehrere Verhaltensweisen, die bei verschiedenen Gelegenheiten ausgeführt werden. Um zu sehen wie sie verwendet werden, verwende gamTool um seine GAM-Seite zu öffnen. Suche nach Generic Object: Old Scratch.
Bist Du dort, öffne den AI Settings-Zweig und gehe runter zu Movement. Gehe nach Movement[0], Animation[0] und Du wirst sehen, daß Animation[0] über vier Sekunden von Frame 0 bis 11 läuft. Nun schau auf die `.oof`. Scratch beginnt ein Verhalten bei Frame 0 auszuführen und kommt in Frame 11 auf seine Startposition zurück. Zurück in der GAM, schau Dir Movement[1], Animation[0] an - eine Animation, die von 11 bis 17 geht. Movement[1], Animation[1] geht von 17 bis 21. Gehe in die `.oof`, und Du kannst die Korrelation sehen.

Natürlich kannst Du nicht einfach Bewegungen zur Table hinzufügen und erwarten daß die ausgeführt werden, außer Du bist ein Heavy-Duty-Modder. Du kannst allerdings einen existierenden Bot nehmen der Dinge tut wenn Du Deine haben willst und sein `.oof` mit Deinem ersetzen (stelle sicher, daß Du die Start- und Stop-Frames entsprechend setzt). Du kannst auch über DALLAS skripten, daß verschiedene Verhaltensweisen ausgeführt werden. Du kannst einen Bot dazu bringen, dich zu verhauen wenn Du einen Schlüssel nicht hast oder ihn sich vor Dir verbeugen lassen, wenn Du ihn hast.

Sicher, das war viel langweilige Info aber, hoffentlich, wird dir diese Information helfen Deine Animation zu planen und Vorteile aus einigen der Grundkonzepte zu ziehen.

089 - Ship Tutorial

Darkside Heartless

Ich dachte schon ich könnte dieses Tut nicht bringen, aber ich habs doch noch gefunden. Leider gibt es nirgendwo dieses schnieke Schiff, das Julian da gebaut hat. Ich mag diese trilaterale Symmetrie...
Die benötigten Tools dürften längst installiert sein.

überarbeitet

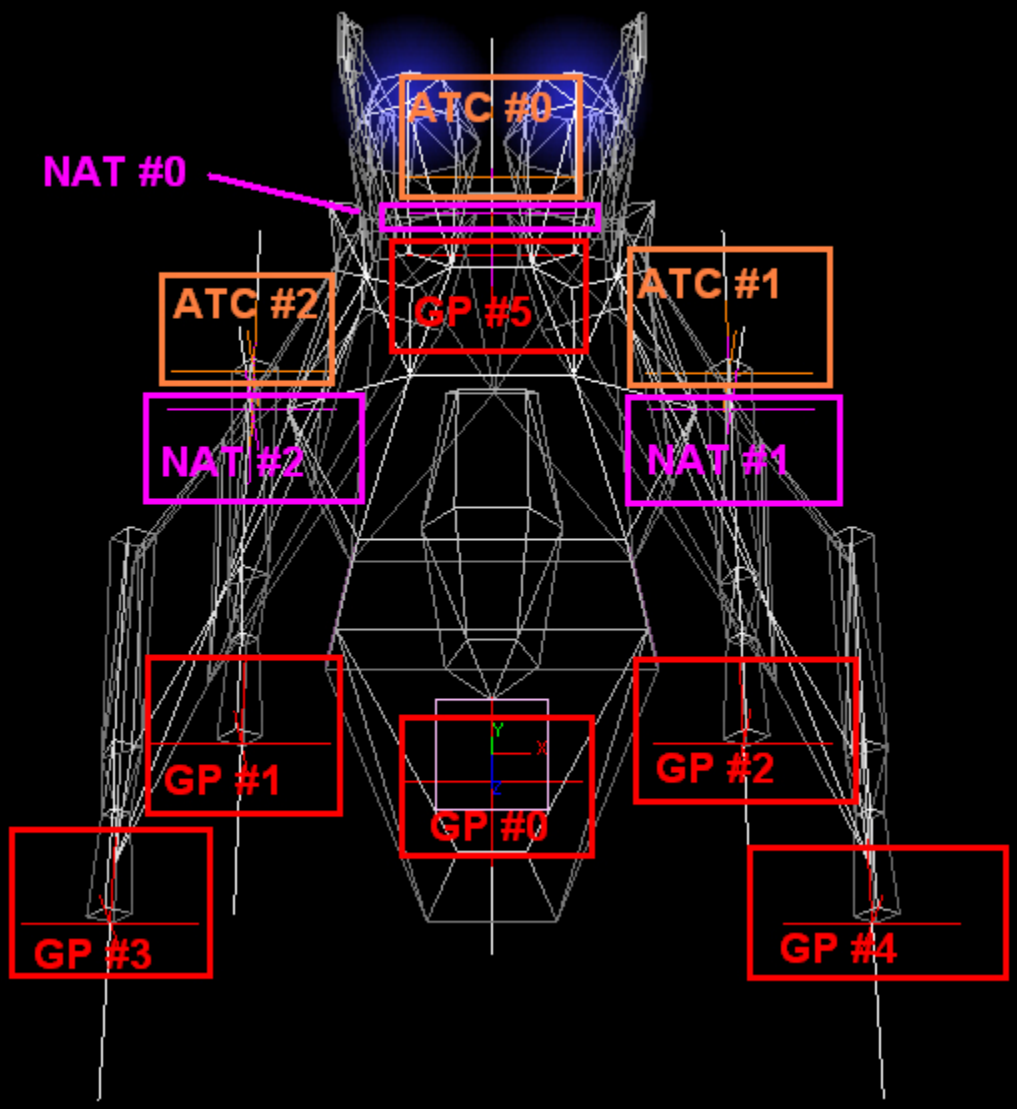
Anmerkung: Dieses Tut setzt voraus, daß Du bereits ein Schiff gebaut hast (ungeachtet des Maßstabs, da der einfach geändert werden kann), aber nicht die nötigen Faces für Gunpoints, Attach Points etc.

Öffne den OOFEditor und öffne Deine `.orf` mit Deinem Schiff.

Da wir keine Faces haben die als Attach Normal Point, Ship Logo etc. fungieren können, müssen wir zurückgehen und sie hinzufügen. Hier ist die volle Liste all der zuvor erwähnten Dinge, von denen alle notwendig sind um ein gutes, voll funktionales Schiffsmodell zu haben:

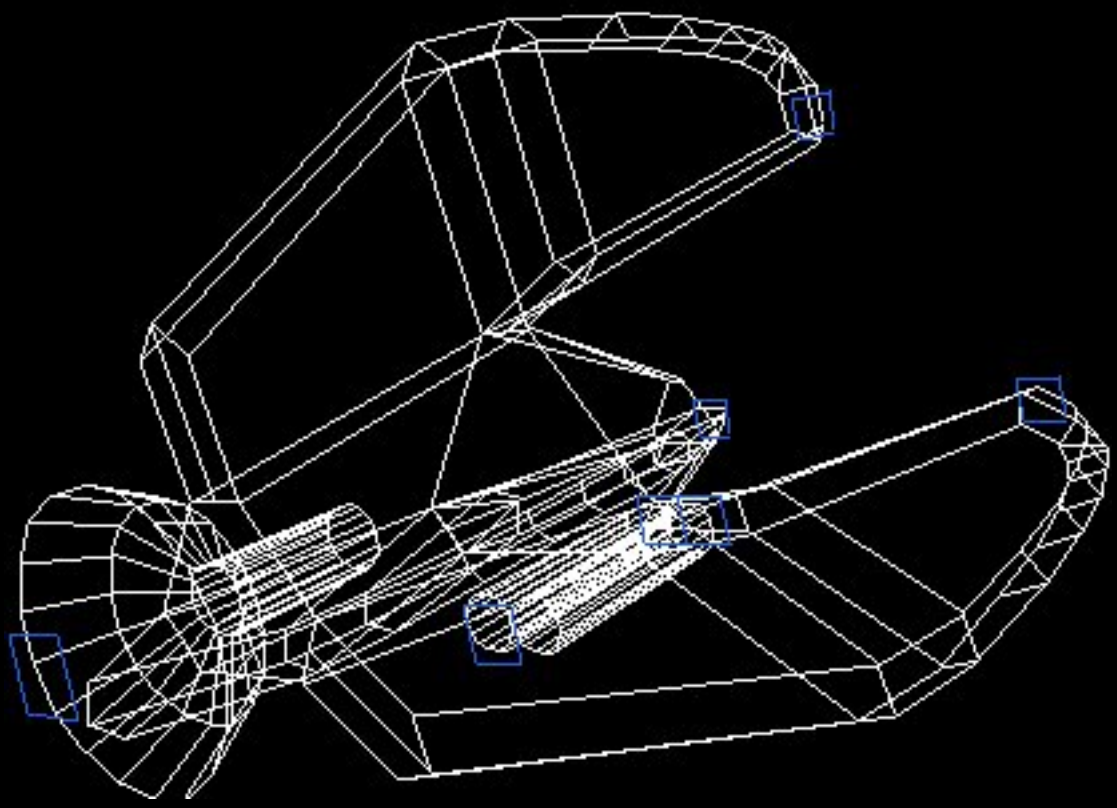
Name	Beschreibung
Gunpoints (GPNT)	Maximal acht Stück (Numeriert von 0 bis 7). Um Waffen abzufeuern. Anmerkung: Nachdem Du alle verfügbaren Gunpoints in der `.oof`-Datei verbraucht hast, werden die verbleibenden Gunpoints in der Tablefile vom letzten verfügbaren Gunpoint aus abgefeuert. Wenn Du nur einen machst, wird alles - ungeachtet dessen was in der GAM steht - von diesem einzelnen Gunpoint aus abgefeuert.
Attach Points (ATCH)	Verwendet, um Objekte ans Schiff zu hängen, wie etwa die Flag in CTF.
Attach Normal Points (NATH)	Bestimmen welche Richtung für die korrespondierenden ATCH ‚Oben'ist.
Glow Points	Erzeugen Lichtschein, typisch dafür der Engine Glow der Schiffe.
Ship Logo Faces	Sie werden zur anzeige der Multiplayer-Schiffslogos verwendet. Kann auch mehr als ein Face sein. Wenn dies so ist, müssen sie aber ein Submodel sein.

Um eine Ahnung zu bekommen wo Du Deine Punkte hinhaben wollen wirst, hier ist wo sie sich am Pyro-GL befinden. Die Punkte werden In der Reihenfolge numeriert wie sie erstellt wurden:

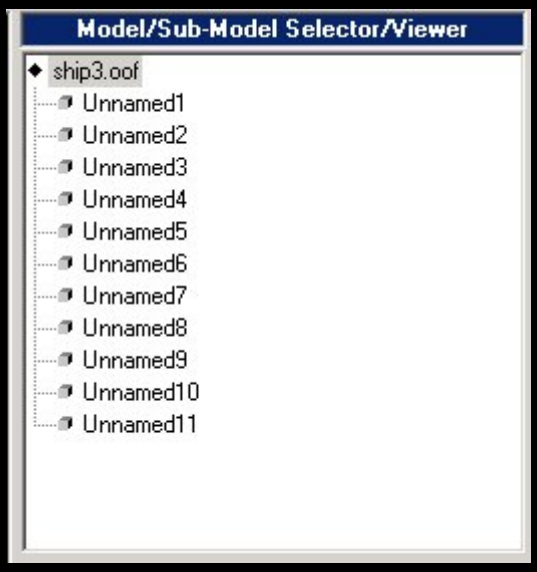


Wenn Du sie erstellst, verwende kleine quadratische oder dreieckige Faces. (Ich verwende 2x2-Faces, da sie sich leichter mit Kanonenläufen und Motoren ausrichten lassen.)

In diesem Bild sind die platzierten Gunpoint-Faces in Blau:

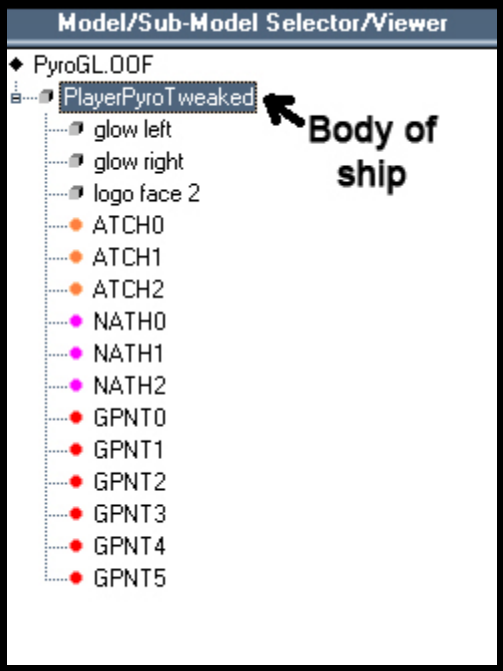


Gehe zurück in den OOFEditor und öffne Deine Schiffs-`.orf`. Du siehst zur rechten die ‚Teile-Liste':



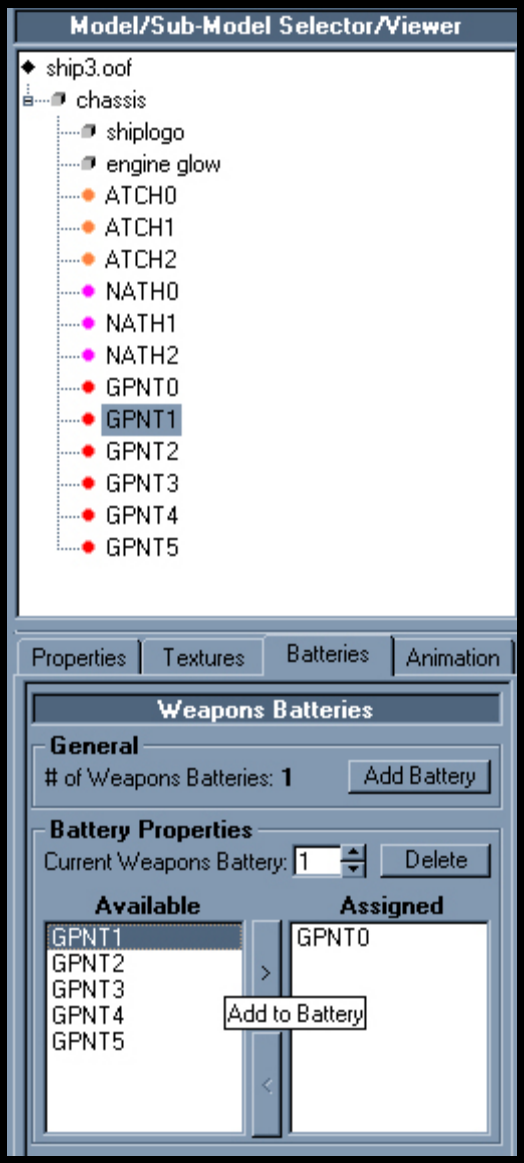
Finde den Hauptkörper und benenne ihn entsprechend. Ist zwar nicht notwendig aber sehr praktisch um den Überblick zu wahren.

Anmerkung: Wenn Du das Schiff nicht in einem Stück gemacht hast, dann vereinige alle Teile des Schiffes AUSSER die zukünftigen Gun-, Attach-, Normal-, Glowpoints und Logo-Faces indem Du Shift niederhältst während Du die Submodels ineinanderziehst. (Siehe dazu den OOFEdit-Exkurs).



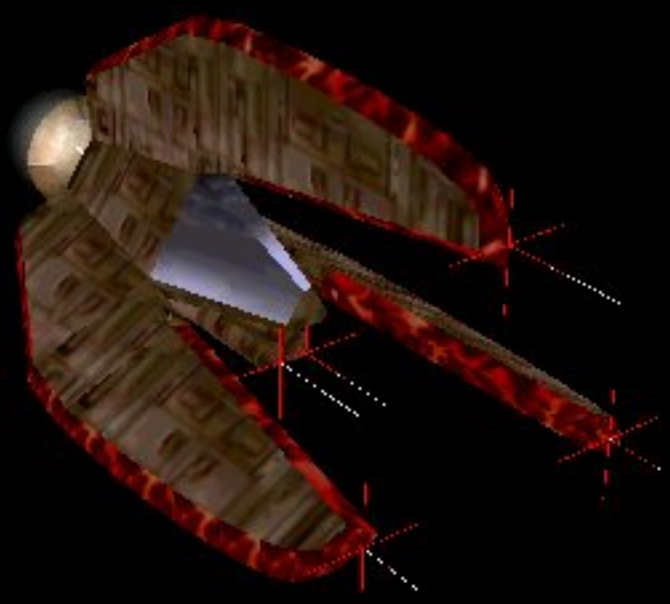
Wenn Du fertig bist, sollte nur noch ein ‚Körper'-Teil sein – Der Rest sind Punkte, Logos und Glows. Hier links ist die Hierarchie des Pyro-GL als ein Beispiel:

Zurück zu Deinem Modell; wähle eins der kleinen Würfelchen in der Tree-Ansicht und ziehe es aufs Chassis. Es taucht dann unter dem Chassis im Baum auf.
Das nennt man Zuweisen, mache das mit allen Faces, die später eine Funktion erfüllen sollen.



So, um anzufangen, klicke eines der kleinen Würfelchen, das der zukünftige Gunpoint[0] werden soll rechts an und wähle convert to -> Gunpoint. Es wird nun in der Ansicht als hellgelbes Kreuz dargestellt, mit einer weißen Linie die in Feuerrichtung weist (vergleiche mit dem Bild oben.). Ist der Gunpoint deselektiert, zeigt er sich als rotes Kreuz (hehe) mit weißer Linie. Mache das für die anderen Faces die Gunpoints werden sollen, dann wähle die Faces die Deine Attach- und Attach Normal Points werden sollen und konvertiere sie.
Dann wähle eins der Faces aus die für die Engine Glow gedacht sind, und gehe in die Properties-Tab und wähle dort wo ‚None' steht Glow/Thruster aus. Der Punkt scheint zu verschwinden, aber in der Properties-Tab ist eine Glow-Sektion aufgetaucht. Wähle dort die Farbe und die Größe die Du willst, und speichere erstmal das Schiff als `.oof` ab.
Gehe dann in die Batteries-Tab und klicke **Add Battery**; die verfügbaren Gunpoints zeigen sich dann in der Available-Box. Klicke jeden an und danach den >-Button, die markierten bewegen sich in die Assigned-Box. Mach das bis alle in der Assigned-Box sind, und speichere dann.

Gehe ins Tools-Menü und wähle Scale Model..., dort wähle Scale by Radius und gib dort einen Wert um die sechs ein, +/- 0,5 Units. Werte in diesem Bereich sind im Allgemeinen gut.



Öffne dann das gamTool und öffne entweder eine existierende oder erzeuge eine neue GAM: Edit -> New -> Player ship. Öffne den Zweig mit dem + und setze deine Schiffs-`oof` als Model ein. Achte auf die Medium und Low model names und lösche diese wenn Du keine entsprechenden Models gemacht hast.

Füge dann die `.gam` und das `.oof`-Model zum Level hinzu, beachte was es halt so zu beachten gibt (Namensgleicheit etc.).

Abschnitt I – Scripting

In Descent 3 kann man Objekten im Spiel ‚Leben' einhauchen, und dies geschieht mittels Skripten. Hier wird richtig programmiert, auch wenns vielleicht nicht so aussieht!

Ab hier muß Dallas laufen. Guckt gegebenenfalls in die entsprechende Sektion: 'Vorgriff: DALLAS zum Laufen kriegen'.
Außerdem ist es spätestens ab hier eine gute Idee, Block & Stift in Griffweite zu haben 😊

090	**Scripting: Hello World**	Die ersten zwei Skripten	Kyouryuu	330
091	**Scripting: Einführung**	Verschiedene Owner	Starkiller	333
092	**Neue DALLAS Suchfunktionen**	Ein Hilfsmittel beim Skripten	Atan	335
093	**Teleporter**	Von A nach B in Nullzeit	Fischlein	343
094	**Create Lighting**	Blitze, Strahlen und Anderes	Fischlein	344
095	**Wind skripten**	Wind per Skript machen	Papacat	346
096	**Musikregionen**	Ereignisgesteuerte Soundtracks	Kyouryuu	347
097	**Spew!**	Flammen, Rauch, Dampf...	Starkiller	350
098	**Glasbruch mit Scherben**	Mehrere Bedingungen	Papacat	352
099	**Einen Schalter skripten I**	Einen Schalter einsetzen & anwenden	WillyP	356
100	**Einen Schalter skripten II**	Das Schalter-Skript verfeinern	WillyP	359
101	**Scripten einer Inventory-Cloak**	Ein-/Ausschaltbare Tarnvorrichtung	Starkiller	362
102	**Reactor Gamma: Script Companion**	Erläuterung des Skriptes zum Level	Kyouryuu	366
Die verlorenen Tutorials Hier fehlt entweder das Tutorial oder das Beispiellevel.				
	kein tut	exit sequence		
	keine files	Hal9000 - scripting		

Toc

090 - Scripting: Hello World

Kyouryuu

Des Programmierers Anfang ist 'Hello World'. Wer bereits mit Programmierung zu tun hatte wird dies wiedererkennen ☺, und auch hier werden wir es so halten.

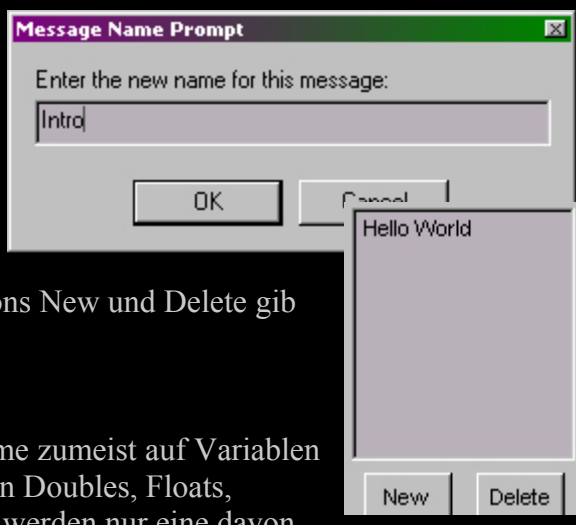
DALLAS muß installiert und eingerichtet sein, siehe 'Vorgriff: DALLAS zum Laufen kriegen'.

Die Anatomie eines Skriptes verstehen

Vorerst ein paar Worte zu den Datei-Extensionen. Wenn du in Dallas zu arbeiten beginnst, generiertst Du zwei Dateien - eine `.cpp`- und eine `.msg`-Datei (auch eine `.o`, aber die kannst Du [hier] ignorieren). Die `.cpp`-Datei ist eine einfache Textfile welche das gesamte Programmzeugs enthält, das Dallas ausspuckt. Tatsächlich ist Dallas ein nettes Frontend, welches das gesamte Coding für Dich übernimmt. Die `.cpp` ist das Ergebnis. Die `.msg`-Datei enthält Nachrichten, dazu später mehr. Wenn Du das Skript kompilierst, erzeugst Du eine `.dll`-Datei. In Wirklichkeit muß die finale .mn3 Deines Levels nur die `.dll` und die `.msg` enthalten. Doch die meisten Levelautoren schmeißen auch die `.cpp` dazu, für den Fall daß der User sehen will wie die Dinge geskriptet sind oder sie es auf einer anderen Plattform kompilieren wollen (z.B. Linux oder Mac)

Ein Beispiel-Skript

Laß uns annehmen, Du hast ein Level gebaut. Ich werde Dich durch eine sehr grundlegende Reihe von zwei Ereignis-Skripten führen. Ich möchte eine Nachricht auf das Spieler-HUD pflastern, die drei Sekunden nach Start des Levels erscheint. Klingt banal und langweilig genug, oder?
Gut, öffnen wir D3Edit und laden unser Level. In der World View gehe nach File -> Dallas Graphical Script Editor. Erstaunlicherweise geht ein neues Fenster auf, angefüllt mit guten Sachen. Die zwei Fehlermeldungen die besagen daß weder eine `.cpp` noch eine `.msg`-Datei geöffnet werden kann, kannst Du einfach wegklicken - es gibt sie ja noch nicht.



Die Message List

Zuerst müssen wir die Nachricht erzeugen, die wir auf das HUD bringen wollen. In der Box mit der Überschrift Message List klicke auf den Button **New**. Es geht eine kleine Dialogbox auf welche Dich danach fragt, wie die Nachricht heißen soll. Es ist nicht die eigentliche Nachricht - nur wie Dallas auf sie referenzieren wird. Gib `Intro` ein. In der Box genau über den Buttons New und Delete gib den Text der Nachricht ein. Darf ich vorschlagen, '`Hello World`'?

Eine Benutzervariable erzeugen

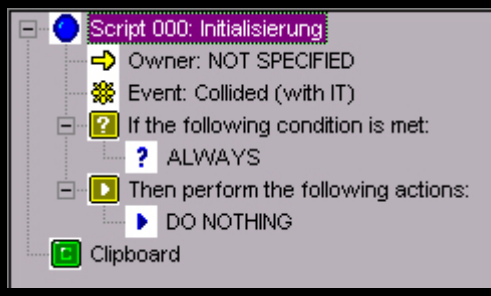
Wenn Du mit Programmieren vertraut bist, weißt Du daß Programme zumeist auf Variablen zurückgreifen, welche Du zu dessen Beginn deklarierst. Das können Doubles, Floats, Integers etc. sein. Dallas hat seine eigenen Variablentypen und wir werden nur eine davon in diesem Skript verwenden (eine TimerID-Variable).

Klicke auf den langen Button **User Types Workshop**. ein schmales Fenster geht auf. Wenn Du oben auf das Pulldown-Menü klickst (Wo GoalID steht) siehst Du eine Anzahl an verschiedenen Variablen-Typen die man deklarieren kann. Verschiedene Variablen dienen verschiedenen Zwecken. Zum Beispiel, wenn ich eine Variable will die grundsätzlich ein Integer ist den ich inkrementieren, dekrementieren oder sonswas kann, würde ich eine UserVar machen. In diesem Fall wollen wir eine Zeitgeber-Variable, also müssen wir eine TimerID-Variable hier verwenden. Wähle also TimerID. Klicke auf den Button **Add New Value** und benenne Deine Variable. Du kannst sie benennen wie Du möchtest (meine nannte ich `Nachrichtentimer`), versuche aber sie anschaulich zu halten. Danach klicke den Button **Return to Dallas**.

Skript 1 von zwei erzeugen

Nun, machen wir ein Skript, oder? Klicke auf den Button New bei Scripts. Ein Dialog geht auf, der Dich danach fragt wie Du das Skript benennen willst. Wähle wieder etwas Anschauliches hier. Meins habe ich Initialisierung genannnt.

Öffne es indem Du den Button **Expand** klickst oder über die +- Zeichen. Schauen wir mal was jedes dieser Felder bedeutet.

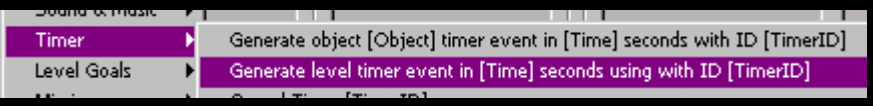


Owner - Dies bezieht sich darauf wen oder was das Skript betrifft. Zum Beispiel, wenn ich ein Skript schreiben wollte, wo sich etwas darauf bezieht daß ein Bot zerstört wird, würde ich hier den Namen des Bots spezifizieren. Wie bei allen Feldern, kann man hier eine Option durch **Rechtsklick** darauf auswählen. Also rechtsklicke den Owner. Wir wollen in diesem Skript daß es genau mit dem Level beginnt, also ist der Level der Owner.

Event - Was passiert dem Owner, das Descent3 mit dem Skript fortsetzen läßt? Wenn wir über unseren hypothetischen Bot sprechen, würde ich prüfen ob er zerstört wurde. Aber wir haben es hier mit einem Level zu tun. Und der Event ist, wie Du vielleicht erwarten wirst, ist der Start des Levels. Wähle Level Start, indem Du auf das Event-Feld rechtsklickst (falls es nicht bereits darauf gesetzt ist). Wie Du sehen kannst, ist da ein Haufen Zeugs das ausgegraut ist - Du kannst Dir die enormen Möglichkeiten vorstellen.

If the following condition is met: - Bei diesem Skript kannst Du es auf ALWAYS lassen. Dies deswegen, da wir wollen daß es bei jedem Levelstart ausgeführt wird. Weitere Bedingungen brauchen wir nicht. Wir werden aber, im zweiten Teil des Skriptes wenn wir irgendwas reintun wollen.

Then perform the following actions: - Klicke einfach rechts auf DO NOTHING und schau Dir die ganzen verschiedenen Sachen an die man hier machen kann! Nun, in unserem Fall wollen wir daß die Zeitgeber-Variable bis drei Sekunden zählt. Rechtsklicke auf DO NOTHING und halte die Maus über Replace with a Different Action. Gehe dann zur Option Timer, und dann klicke Generate Level Timer.... Ein paar neue Felder tauchen auf (wenn nicht, expandiere den Baum um sie zu sehen). Eins ist Time, ein anderes TimerID. Für Time spezifiziere drei Sekunden indem Du rechts draufklickst, und Enter Float Value auswählst, 3 eingibst und **OK** klickst.



Soweit wollen wir daß der TimerID, den wir als nächstes spezifizieren werden, zählen soll. Also rechtsklicke auf TimerID und gib Deine Variable an.

Also haben wir nun das Folgende:

"Wenn der Level beginnt, lasse die Variable TimerID bis drei Sekunden zählen."

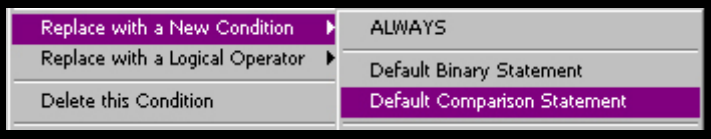
Aye, mag Viel für Wenig scheinen, aber Du wirst damit schneller werden.

Skript 2 von zwei erzeugen

Bis jetzt zeigen wir eigentlich die '`Intro`'-Nachricht nicht am HUD an. Das wird dieses Skript machen. Es wird Dich auch in etwas Neues einführen. Klicke auf **New** und gib einen Namen ein. Meins benannte ich '`zeigeNachricht`'.

Expandiere das Ganze und Du wirst bemerken, daß es gleich wie das letzte Skript aufgebaut ist. Unser Owner ist noch immer der Level. Keine Objekt- oder Spieler-Aktion ist hier involviert - nur der Level. Unter Event liegt die Sache etwas anders, wir wollen hier daß Descent3 nur dann in diesem Skript weitermacht, wenn der Intro-Timer die drei Sekunden durchgezählt hat. Also wähle Timer (TIMER ID) went off.

Jetzt, im Bereich If the following conditionis met, wollen wir Descent 3 erklären welche TimerID auf Ablauf zu prüfen ist. In anderen Worten, wir wollen daß es weiß daß wir daran interessiert sind wann Nachrichtentimer bei der dritten Sekunde angelangt ist und deswegen zu zählen aufgehört hat. Dazu werden wir eine Vergleichende Anweisung einsetzen - eine, die Ein Ding mit einem Anderen vergleicht. Rechsklicke auf das ALWAYS und gehe über Replace with a New Condition und klicke dann auf Default Comparison Statement. Ein paar neue Felder sollten erscheinen, wenn nicht expandier's.



Grundsätzlich, was wir hier haben sind zwei Variablen zwischen denen eine Operation ausgeführt wird. Ist nicht soo schwer zu verstehen - es ist sowas wie A=B oder A>B oder A<B. Also was sind A und B in diesem Fall? `Nachrichtentimer` und TIMER ID. Vielleicht bist Du genötigt, an diesem Punkt zu fragen 'wieso TIMER ID?'. Nun, stelle es Dir so vor: Nachrichtentimer zählt bis drei, aber TIMER ID weiß bereits, daß der bis drei geht. Also, indem wir sagen `Nachrichtentimer` = TIMER ID wird Descent 3 sagen "Hm, schaut so aus als wenn `Nachrichtentimer` = 3, und TIMER ID hat gesagt daß der bis dorthin zählt, also stimmt die Bedingung!"

Wie kriegt man das jetzt ins Dallas? Rechtsklicke auf das erste Literal und mache Replace with Literal, Enumerated Types und dann TimerID (USER TYPE). Nun sollte der If...-Bereich zwei TimerID-Variablen vergleichen, welche aber momentan auf (None) gesetzt sind. Ändere das, indem Du auf den Ersten rechtsklickst und und `Nachrichtentimer` angibst. Beim nächsten weise über rechtsklicken TIMER ID zu. Somit hast Du die Anweisung "Wenn `Nachrichtentimer` = TIMER ID" erzeugt.

Nun gibts für jedes 'Wenn' ein 'Dann'. Wenn `Nachrichtentimer` = TIMER ID, dann wollen wir daß unsere '`Intro`'-Nachricht am HUD angezeigt wird. Klicke dort wo DO NOTHING steht rechts drauf, und mache Insert new action (Anstatt Replace - Du wirst später sehen warum), dann gehe nach Misc und dann Show HUD message [message]. Eine neue Zeile erscheint (sonst expandier's) wo Du die Nachricht spezifizieren kannst. Rechtsklicke drauf und wähle die '`Intro`'-Nachricht.

Gratulation! Du hast nun das eigentliche Erstellen des Skriptes das wir für hier benötigen fertiggestellt.

Das Skript kompilieren und den Level testent

Wenn Dein Dallas ordentlich funzt, sollte das eine Kleinigkeit sein. Klicke einfach den **Save**-Button und laß Dallas sein Ding durchziehen. Wenn es in der Nachrichten-Box unten im Dallasfenster meint 'Done', gehe aus Dallas raus und speichere Dein Level.

Wenn Du die `mn3` erzeugst, ob in Quicktest oder mn3Edit, inkludiere nicht nur die `.d3l` sondern auch die `.dll`, `.msg` und optionalerweise auch die `.cpp`. Du findest diese im Osiris-Verzeichnis (Du weißt doch noch wo das ist, oder?). Dann speichere die `.mn3` und spiel' Dein Level an.

Bonus!

Hast Du erstmal die ursprünglichen zwei Skripten in Descent 3 funktionell, habe ich eine Extra-Aufgabe für Dich. Den Text so am HUD anzeigen zu lassen ist langweilig, oder? Wie wär's, wenn wir das Ganze mit einem Sound-Effekt ein wenig aufpeppen? Gehe zurück und editiere das zweite Skript so, daß ein 2d-Sound mit 100% Volume abgespielt wird. Ich lasse Dir das als Herausforderung!

Danach

Nun, junge Hüpfer, Ihr seid jetzt auf bestem Wege Dallas-Skript-Meister zu werden. Da sind noch immer andere Sachen zu erlernen (Ihr habt die ausgegrauten Funktionen gesehen), aber zu gegebener Zeit werdet Ihr lernen diese zu meistern! Und dann werden Eure Level mehr als nur bloße Shells zum drin herumfliegen. Sie werden zu lebenden, blühenden Reichen werden.

091 - Scripting: Einführung

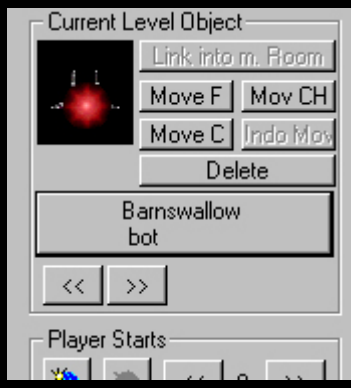
Starkiller

Was ist ein Skript?

Ein Skript ist ein Teil einer von Descent 3 verwendeten Datei, das sagt "*Wenn das passiert, und jenes so ist, mache dieses.*" Der '*Wenn das passiert*'-Teil wird Owner und Event genannt. Der Owner ist das Ding dem was passiert, und der Event ist das, was dem Ding passiert. Die Bedingung ist der '*jenes ist so*'-Teil. Dort setzt Du die genaueren Faktoren des Events. Solche genaueren Faktoren sind was erzeugt den Event und die Definitionen von einigen Level-Ereignissen. Der '*mache dieses*'-Teil ist das Ausführen der nachfolgenden Aktionen.

Mache ein Level

Für diese Übung werden wir einen Default-Raum-Level verwenden. Ich habe den Raum etwas ausgedehnt, indem ich ein paar Verts markiert und bewegt habe. Du kannst natürlich raffiniertere Level in Deinen Experimenten verwenden. Als nächstes füge ein Objekt hinzu, einen Bot. Jeder Bot wird's tun, aber ich schlage vor daß Du nimmst einen relativ einfachen nimmst, du wirst ihn bekämpfen müssen. Setze ihn etwa 30 Einheiten vom Player-Startpunkt entfernt ein. Dann wähle in an und benenne ihn '`bot`' Um ein Objekt zu benennen, wähle es an. Dann, im oberen Teil des Objekt-Paneels, über der Ansicht des Objektes, ist ein Button wo der Typ desselben drinnensteht; wenn Du da draufklickst, geht ein Dialog auf wo Du einen Namen vergeben kannst. Benenne ihn und speichere den Level.



Dallas

Gehe einfach in die World View und wähle File->Dallas Graphical Script Editor. Nun solltest Du genau zwei Fehlermeldungen bekommen, welche besagen "Unable to open levelname.CPP" und "Uable to open levelname.MSG". Das ist normal, wenn der Level noch nicht geskriptet wurde, sie sagen grundsätzlich aus daß die Skript-Dateien nicht existieren... weil Du noch keins gemacht hast! Von dort aus erzeuge ein neues Skript.
Dazu gibt es einen Button bei Scripts, der mit **New** beschriftet ist. Benenne es mit '`turnBotOff`'. Setzte den Owner durch Rechtsklick auf Level. Setze den Event auf Level Start. Setze die Bedingung (If the following condition is met) auf ALWAYS. Abschließend setze die Action mit Replace with a new action -> AI auf Turn [On/Off] AI for object [Object]. Diese Aktion ist recht selbsterklärend; wenn eine Aktion oder Abfrage ('*wenn jenes so ist*') ein wenig schwieriger zu verstehen ist, wirf einen Blick auf den unteren Teil des Dallas-Editors. Dort gibt's eine Box, die Informationen über die gegenwärtig ausgewählte Aktion oder Abfrage enthält. Die Aktion die Du gerade hinzugefügt hast, sollte bereits expandiert sein. Rechtsklicke auf die Variablen die sich darin befinden. Bei dieser sind die Defaults OFF und None. Ändere über rechtsklicken das None zu bot (um etwas zu ändern, immer rechts draufklicken). Dies ist Dein erstes Skript, was es macht ist daß es die AI für den `bot` deaktiviert (Die AI läßt Objekte Aktionen vollführen; alle Objekte die keine Powerups sind haben AI, nicht nur Robots). Nun wird der Bot einfach dasitzen und Geräusche machen, wir müssen einen Weg finden um ihn wieder zu 'beleben'.

Mache noch ein Skript. Benenne es '`turnBotOn`'. Setze den Owner auf bot, den Event auf Collided (with IT). (IT) ist eine skriptweite Variable(es wird im gesamten Skript, aber nicht in der gesamten Datei verwendet). (IT) kann darauf eingestellt werden Aktionen zu empfangen und so eingestellt werden, daß nur bestimmte IT's für das Skript funktionieren. Setze die Bedingung mit Replace with a new Condition -> Comparison Statement with Query -> Objects auf [Object] is a player. Lasse die Default-Einstellungen. Beachte, daß das [Object] zum [IT] wird. Das bedeutet, daß wenn das Objekt welches mit dem Owner (IT) kollidiert jenes Objekt ist auf welches die Bedingung achtet (ein player), der '*wenn jenes so ist*'-Teil auf '*das ist so*' gesetzt wird, und Descent 3 dann in die Aktions-Einstellungen sieht um herauszufinden was es nun machen muß. Du wirst zwei Aktionen in diesem Skript haben.

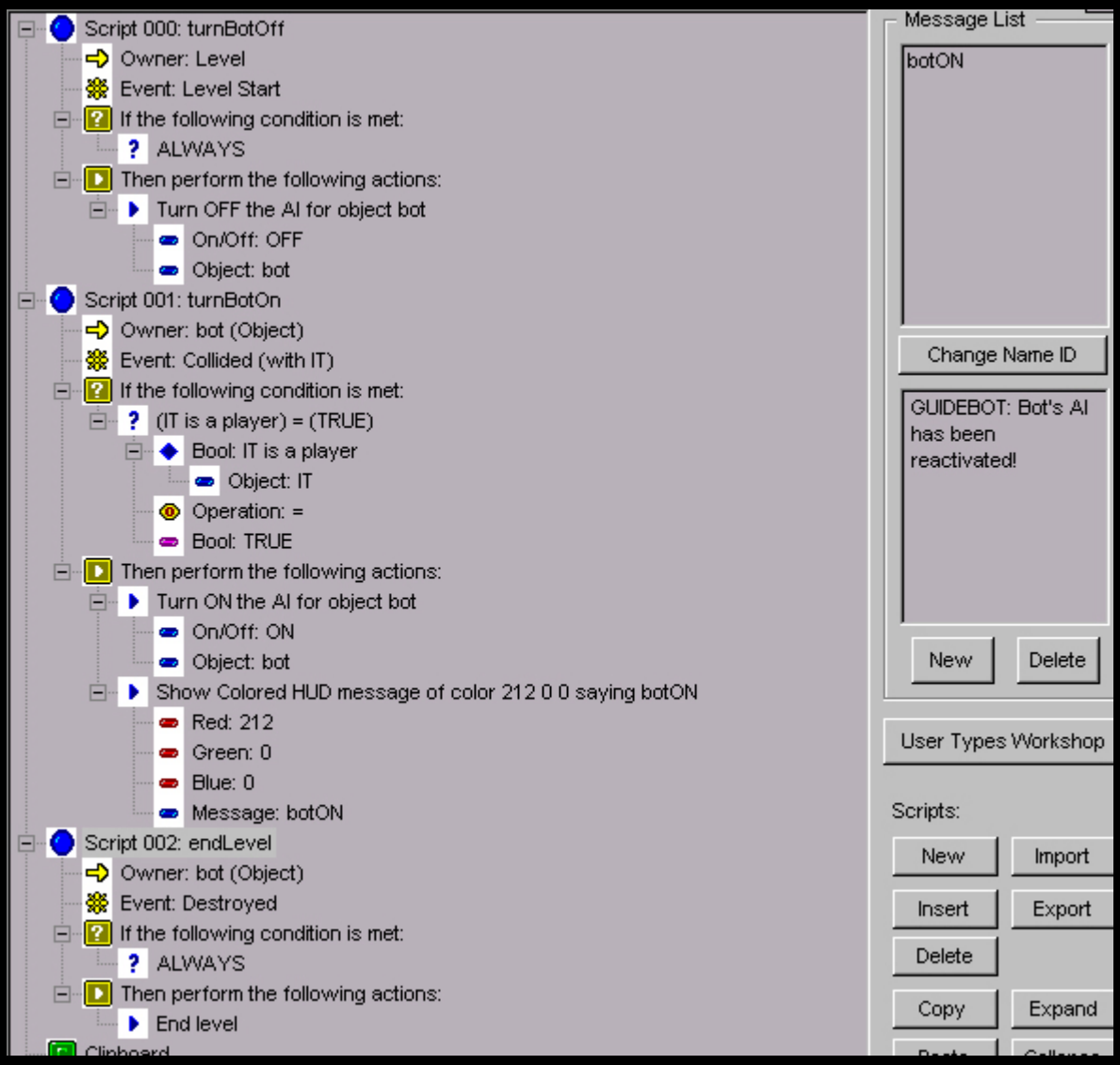
Ersetze das DO NOTHING mit Turn [On/Off] AI for object [Object]. Das ist dieselbe Aktion die Du im anderen Skript verwendet hast, aber diesmal werden wir die Variablen ändern. Setze das Object auf wieder auf bot, aber ändere OFF zu ON. Dann rechtsklicke auf den Header der Aktionen, nicht die Aktion selbst. Gehe nach Add a new action -> Misc -> show Colored HUD message of color [Red] [Green] [Blue] saying [message]. Die Variable für Rot setze auf über 200, die anderen laß auf 0. Dann gehe in den Message List-Bereich, und erzeuge m it **New** eine neue Nachricht; benenne sie mit `botON`. Nun taucht botON im oberen Teil auf, während im unteren noch nichts steht. Dort schreibe rein, auf Groß/Klein-Schreibung achten: `GUIDEBOT: Bot's AI has been reactivated!` Setze die Message:-Variable in der Nachrichten-Aktion auf 'botON'. Dieses Skript schaltet die AI für bot ein, wenn ein Spieler den Robot berührt.

Dein letztes Skript benenne `endLevel`, setze den Owner wieder auf bot. Setze den Event auf Destroyed. Für diese Abfrage müssen wir keine Bedingung hinzufügen. Ersetze die Aktion mit Replace...->Mission->End level. Dieses Skript wird den Level beenden, wenn Du den Robot 'bot' zerstört hast. Speichere Dein Skript, es wird automatisch für Dich kompiliert.

Was mach' ich jetzt?

Mache eine `.mn3`-File wie üblich, aber diesmal mußt Du ein paar Extra-Files hinzufügen. Unter dem **Add file(s)**-Bereich füge `<levelname>.dll` und `<levelname>.msg` hinzu. Wenn es ein Final-Release Deines Levels ist, füge auch noch die `<levelname>.o` hinzu, für Linux-Server. Spiele Dein Level und berührre den Robot, der nur dasitzen sollte. Er kommt dann wieder ins Leben zurück und wird wahrscheinlich auf Dich schießen. Du wirst ihn killen müssen; nach dem er zerstört wurde, sollte das Level enden und Du kommst wieder in den D3-Hauptschirm.
Du hast gerade drei Skripten in ein Level eingebaut!

Hier nochmal alle, die Skripten aufgeklappt:

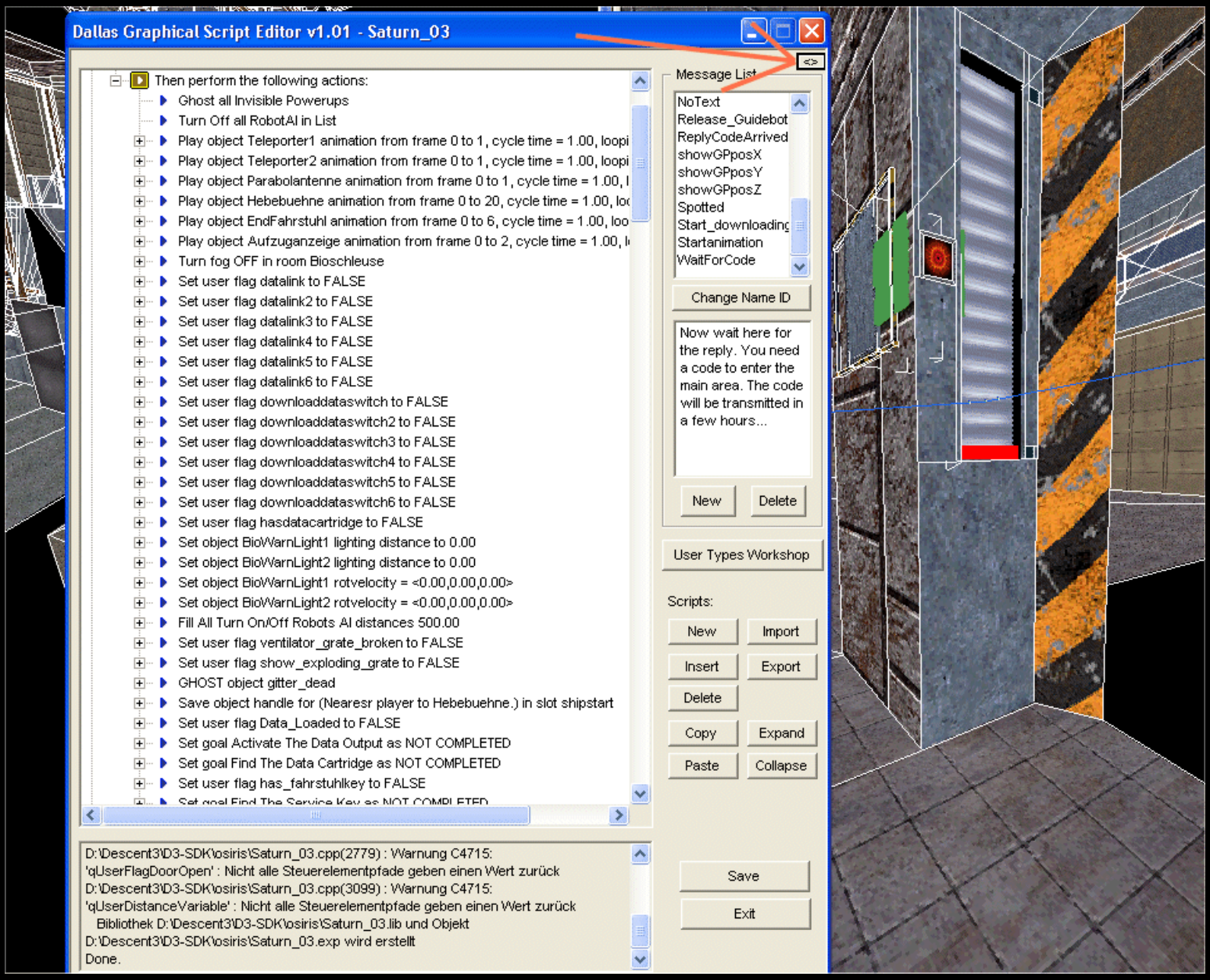


092 - Neue DALLAS Suchfunktionen

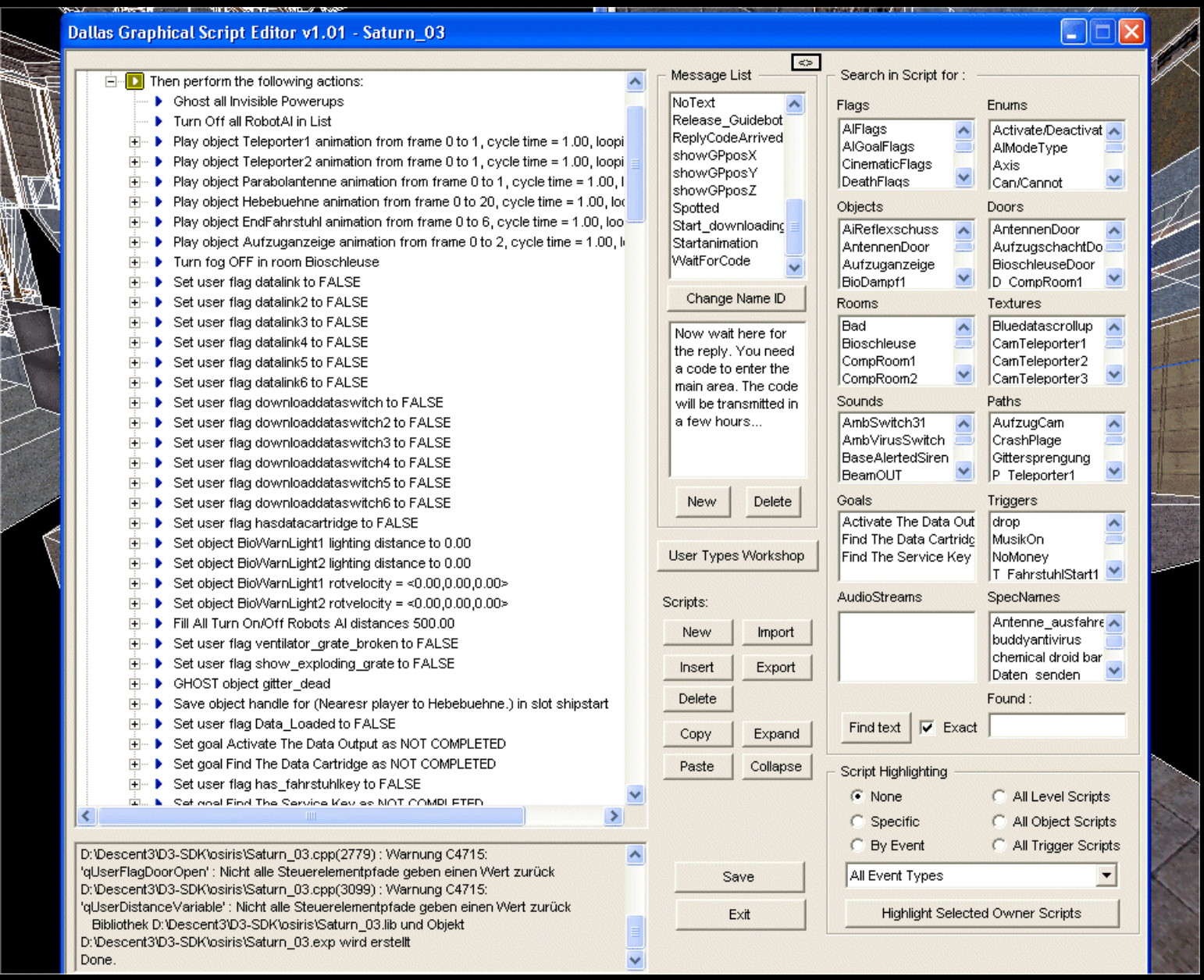
Atan

V 1.0

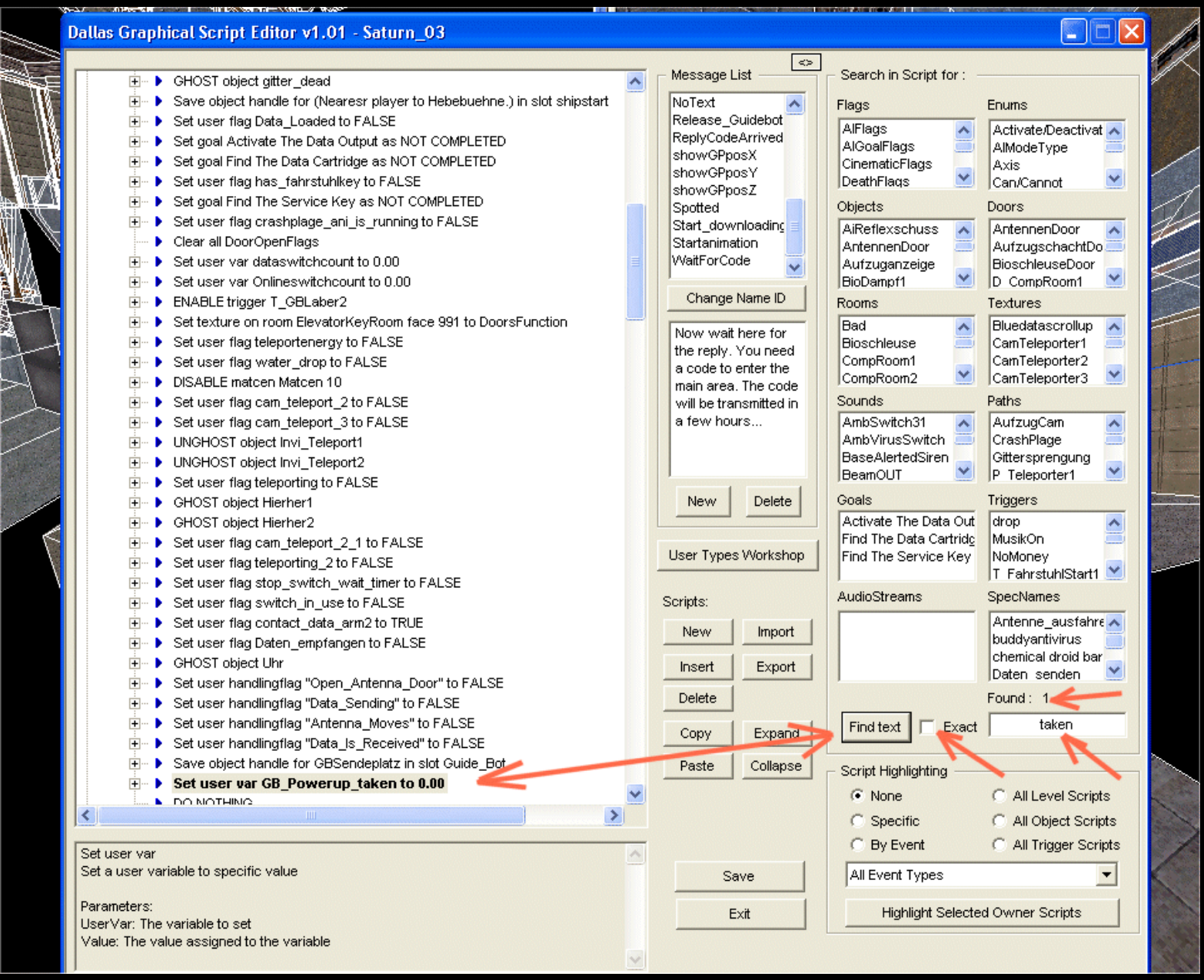
Die Neuerungen können mit einem Schalter Ein- oder Ausgeschaltet werden:



Der Dialog erweitert sich um einen neuen Bereich:

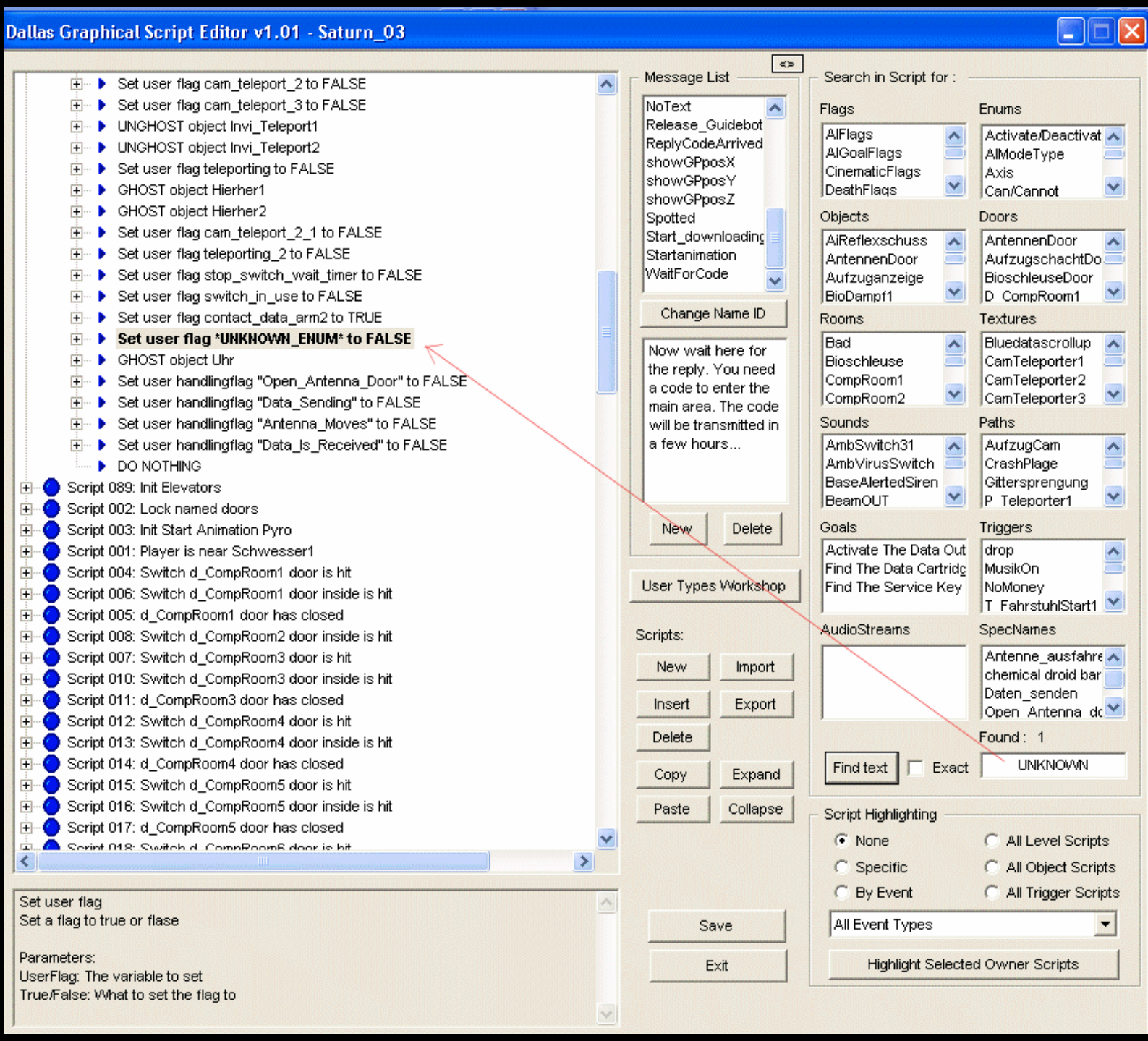


Es ist nun möglich, durch Mausklick oder direkte Eingabe das Script durchsuchen zu lassen und die gefundenen Stellen in den Scriptzeilen hervor zu heben. Hier ein einfaches Beispiel:



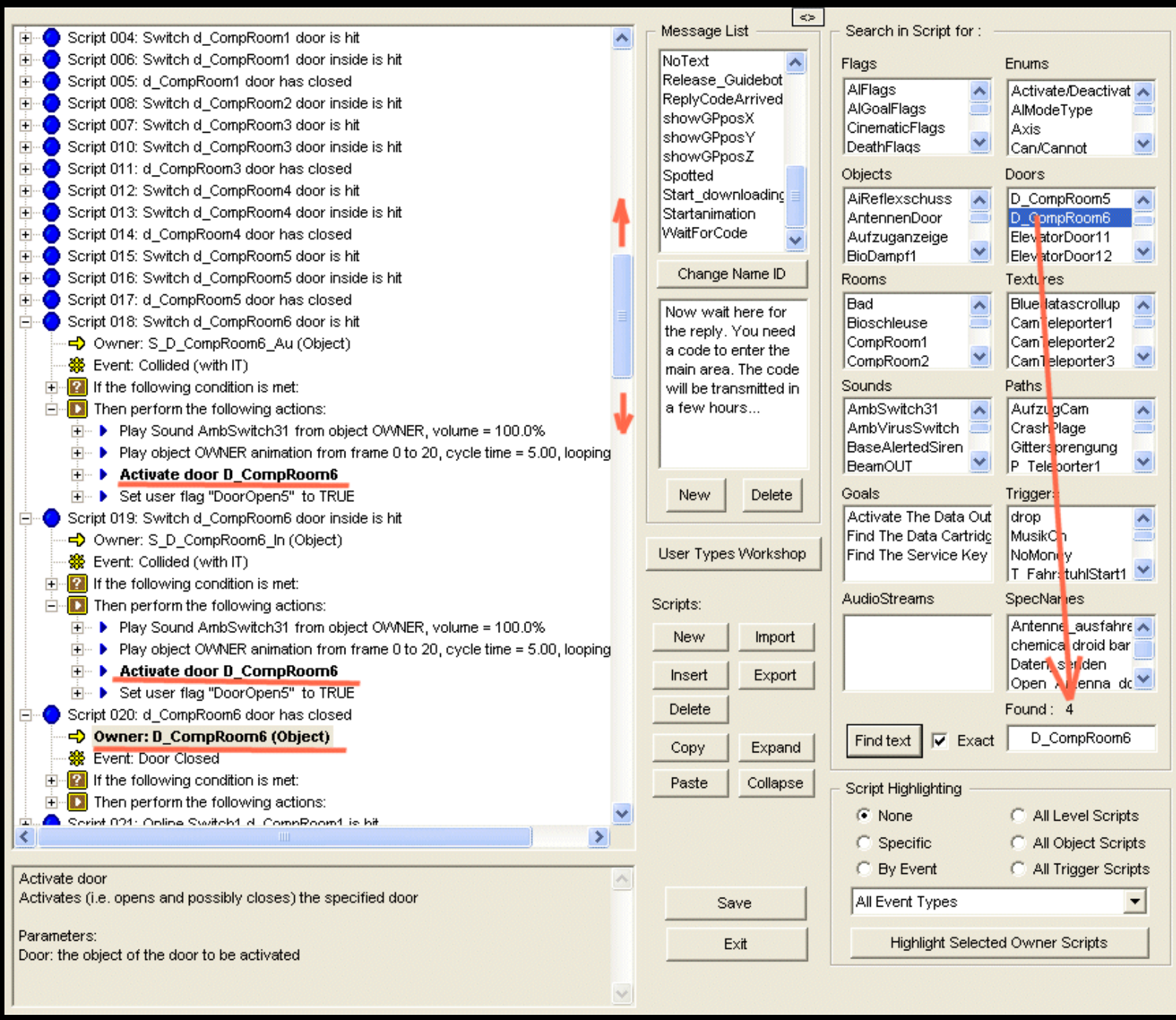
Es wurde das Wort `taken` direkt ins Suchfeld eingegeben, die Checkbox 'Exact' abgewählt und nach Betätigung des '**Find text**'-Buttons sehen wir, dass wir damit genau einen Treffer gelandet haben. Die entsprechende Zeile wird optisch hervor gehoben.

Immer wieder ärgerlich ist es, wenn DALLAS uns mitteilt, das ein oder mehrere Scriptzeilen ein 'unbekanntes' Objekt enthalten das geändert werden muss. Dieses Beispiel zeigt wie die entsprechende Stelle(n) schnell gefunden werden kann:

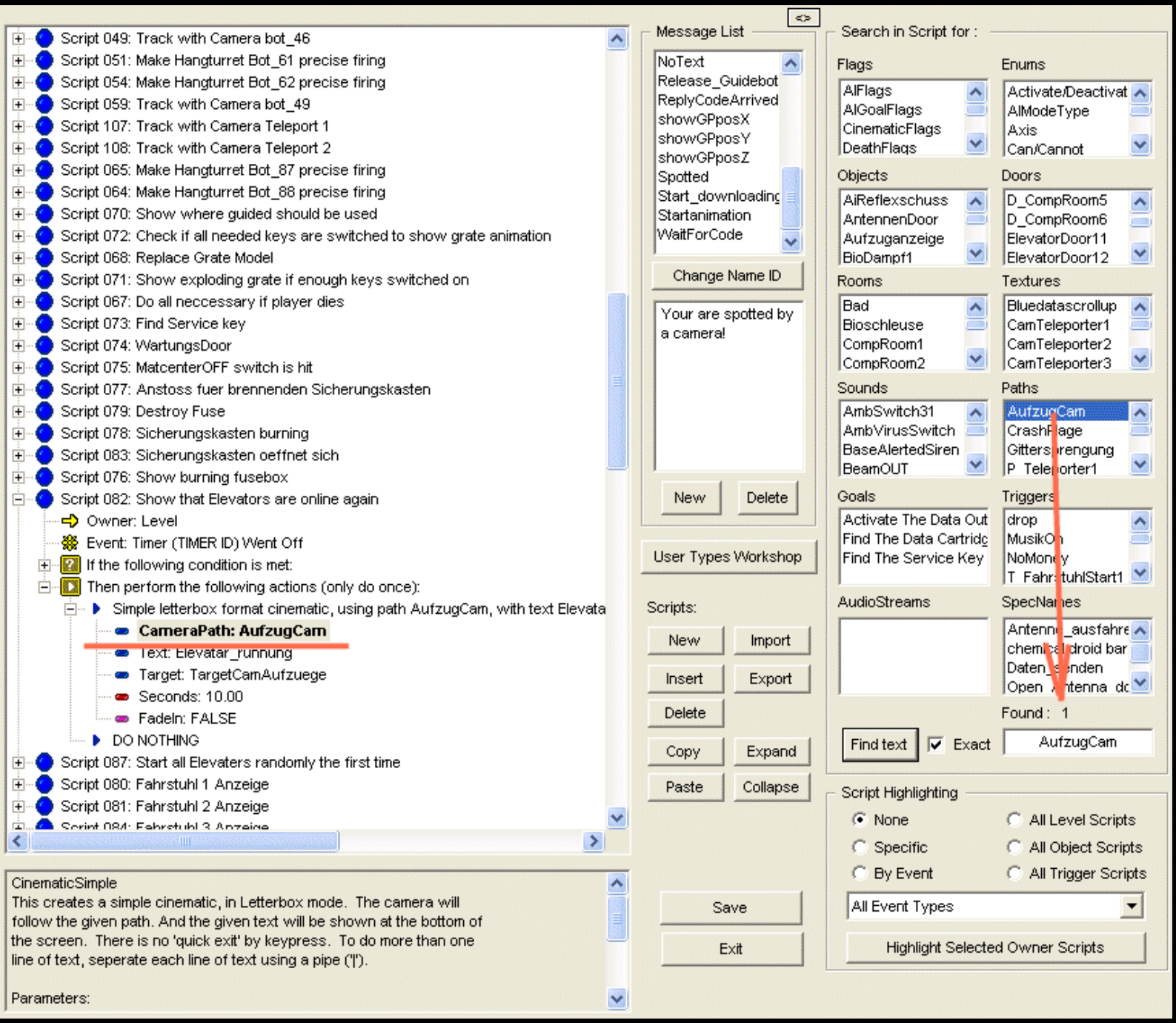


Das Script wird automatisch geöffnet und die entsprechende Zeile angezeigt.
In neuen Versionen sucht DALLAS bereits beim Speichern nach solchen Fehlern und hebt sie Zeile für Zeile optisch hervor.

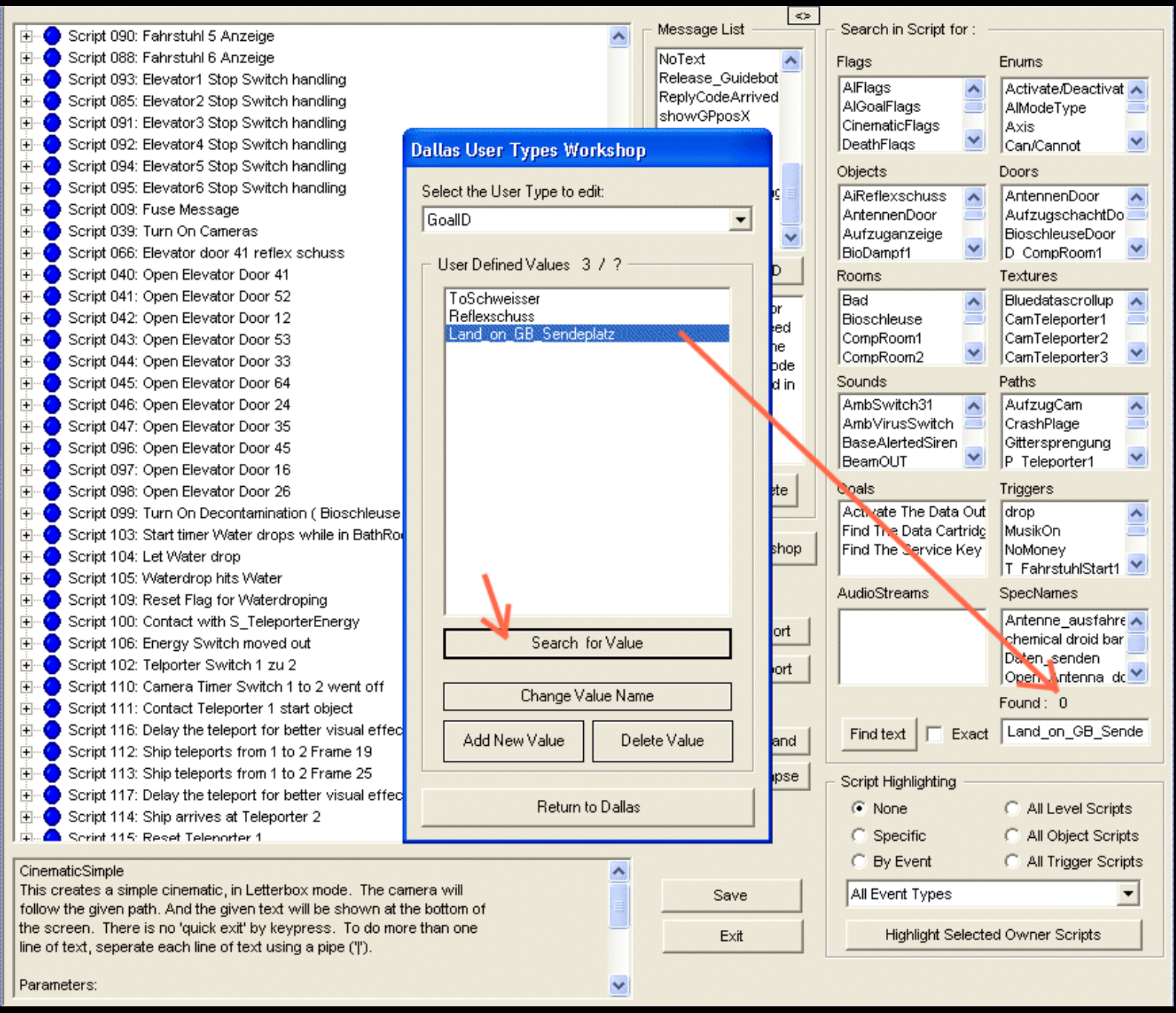
DALLAS füllt beim Starten nun entsprechende Listenfelder. Ein Linksklick auf den gesuchten Eintrag und die Suche wird automatisch gestartet. Der gesuchte Text wird ins Eingabefeld übernommen, die gefundene Anzahl der Treffer angezeigt, alle entsprechende Scripte öffnen sich und der gesuchte Text wird optisch hervorgehoben. Mit dem Scrollbalken kann man weitere Scripte in den Sichtbereich bringen.



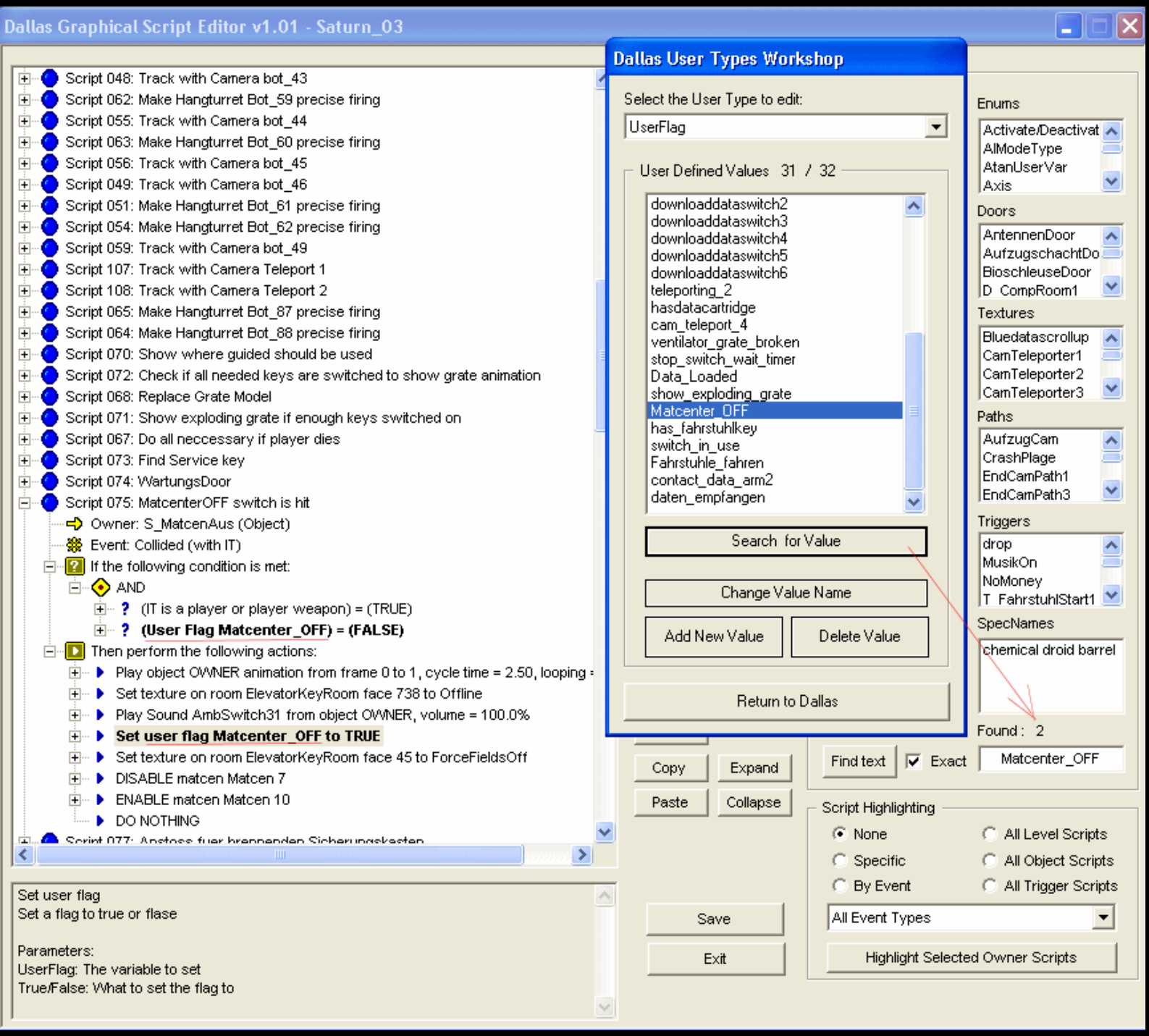
Das Finden von Pfaden und Textausgaben (Messages) wird so zum Kinderspiel.



Um die Einträge des User Types Workshop zu suchen öffnet man diesen, wählt den Eintrag an und klickt auf '**Search for Value**'.
Dieser Eintrag wird in den Scripts nicht gefunden, wird also nicht genutzt und kann u. U. aus der Liste entfernt werden.
Beim Löschen von Einträgen sollte man immer sehr vorsichtig sein, es kann sich dabei immer leicht ein Fehler einschleichen!



Ein weiteres Beispiel:

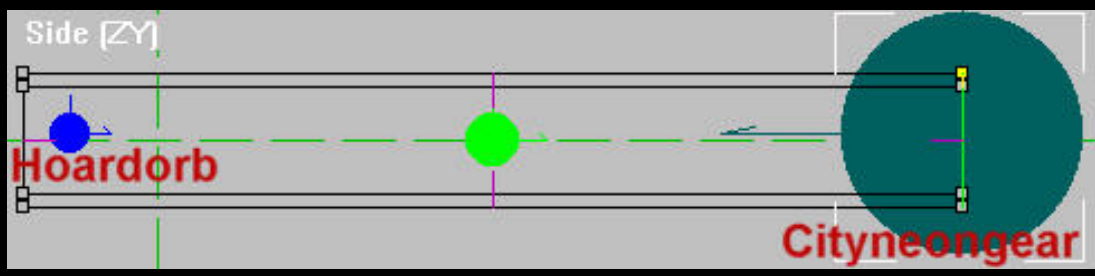


Dieses neue Suchfeature ist sehr hilfreich beim Scripten. Nachdem man sich daran gewöhnt hat, möchte man nicht mehr ohne es Scripten. Sinnvoll genutzt kann man damit sehr viel Zeit sparen, probiert es aus!

093 - Teleporter

Fischlein, inspiriert von Starkiller

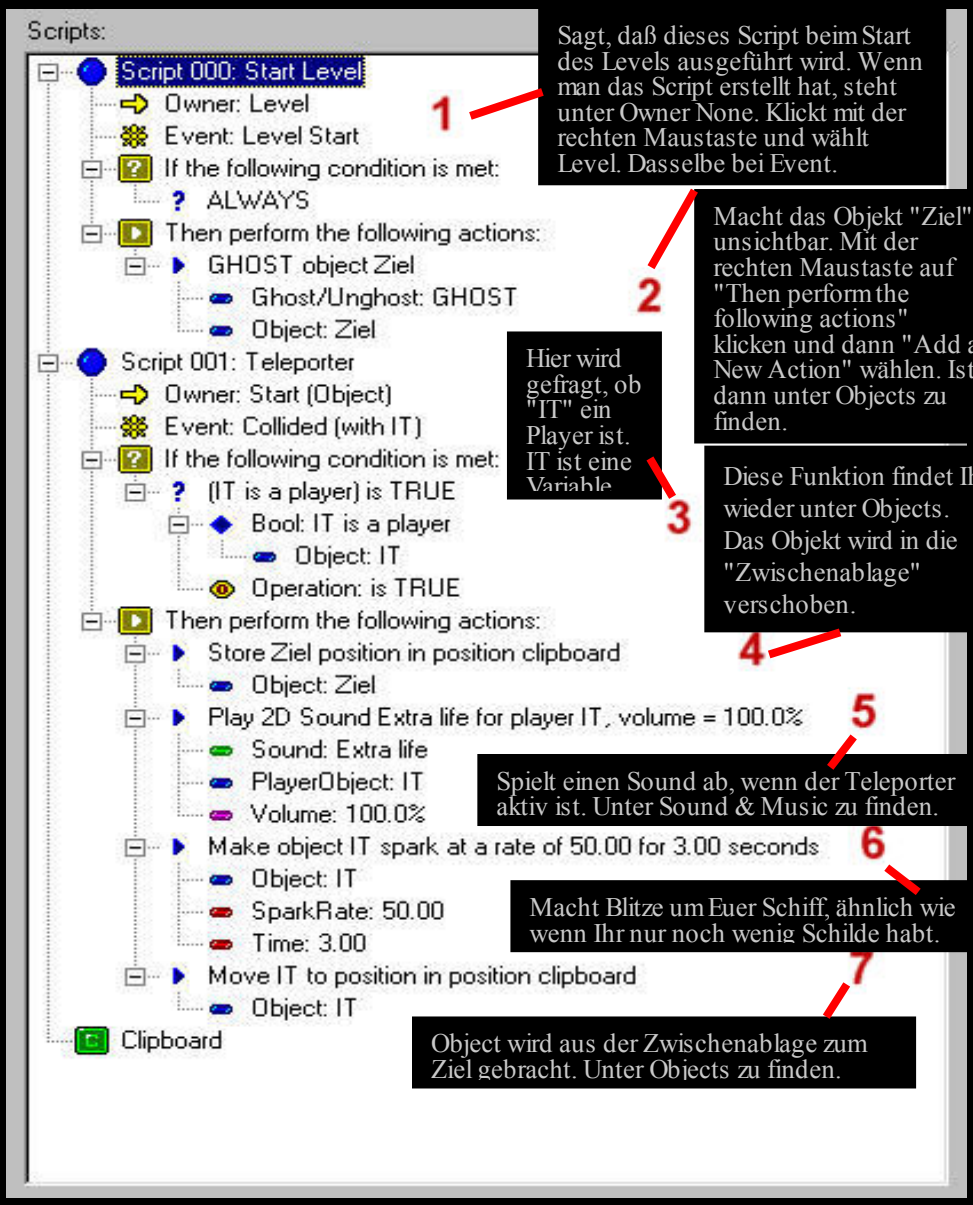
Die Dateien, die sich aus diesem Tutorial ergeben sind in der File.
Wir brauchen für dieses Thema D3Edit und den Dallas Script Editor. Erstellt Euch zunächst ein Level. Ich habe "New Level - Default Room" gewählt und ihn 100 Einheiten in die Länge gezogen.
Den Player Startpunkt habe ich in der Mitte des Raumes positioniert. An dieser Wand habe ich dann im Objekt-Modus das Clutter-Objekt "Cityneongear" positioniert, Bild rechts (man kann auch irgendein anderes Objekt nehmen).



Dieses Objekt habe ich nun "`Start`" benannt, das macht man, in dem man auf die rot umrandete Schaltfläche im Bild links klickt. So, nun brauchen wir noch ein "Ziel-Powerup". Ich habe auf der gegenüberliegenden Wand noch ein Powerup positioniert, ein Hoardorb (sollte ein Powerup sein, da man die Richtung bestimmen kann, in der man austritt). Starkiller weist darauf hin, daß die beiden mindestens 50 Units voneinander entfernt sein sollen. Diesen Powerup habe ich "`Ziel`" genannt. Speichert nun die `.d3l`.

Wir brauchen nun noch ein Script, um den Teleporter zum "Leben" zu erwecken. Wechselt in die World View Ansicht und wählt im Menü File/Dallas Graphical Script Editor, wie dieser eingerichtet wird, erfahrt Ihr weiter vorne. Die zwei Hinweisdialogfelder wie gehabt wegklicken. Wenn Ihr Euer Script das erste mal gespeichert habt, werden diese nicht mehr angezeigt. Im Bild rechts seht Ihr, wie das Script aussehen muss. Die Erklärung dazu steht in den Kästchen.

So das war es schon. Man kann als Teleportereintritt auch ein Portal wählen (Trigger).

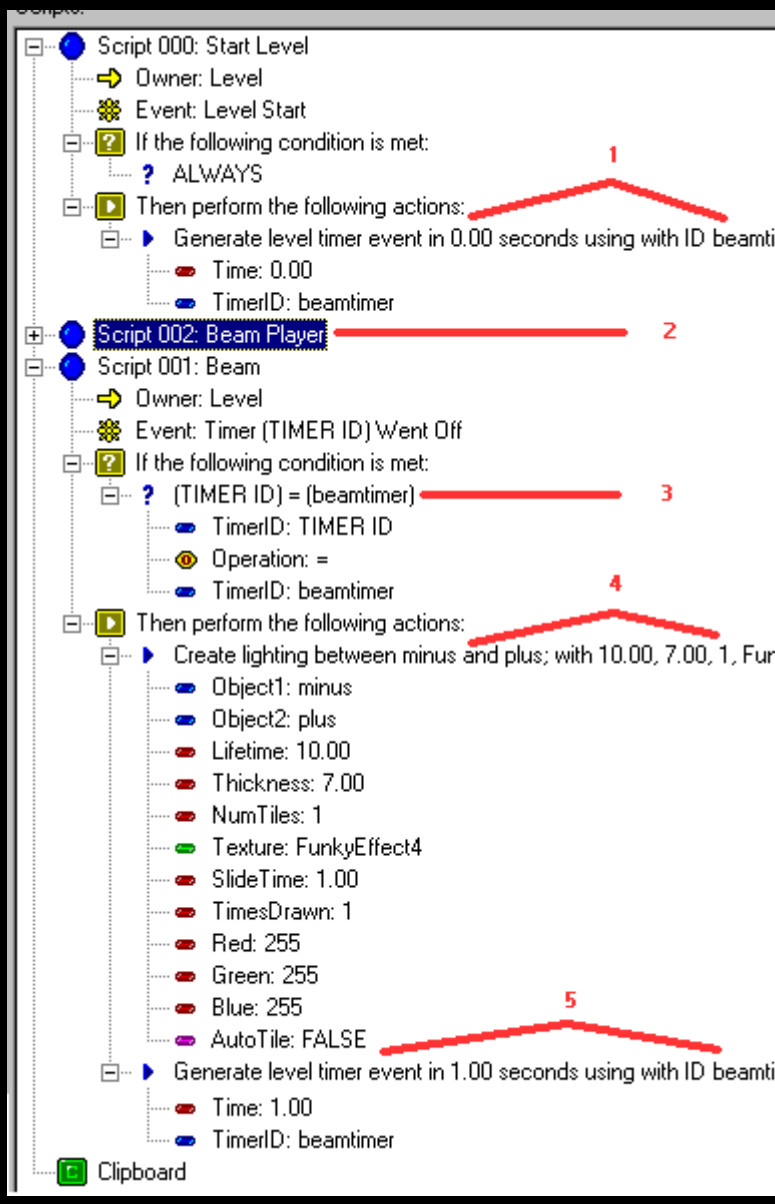


094 - Create Lighting

Fischlein

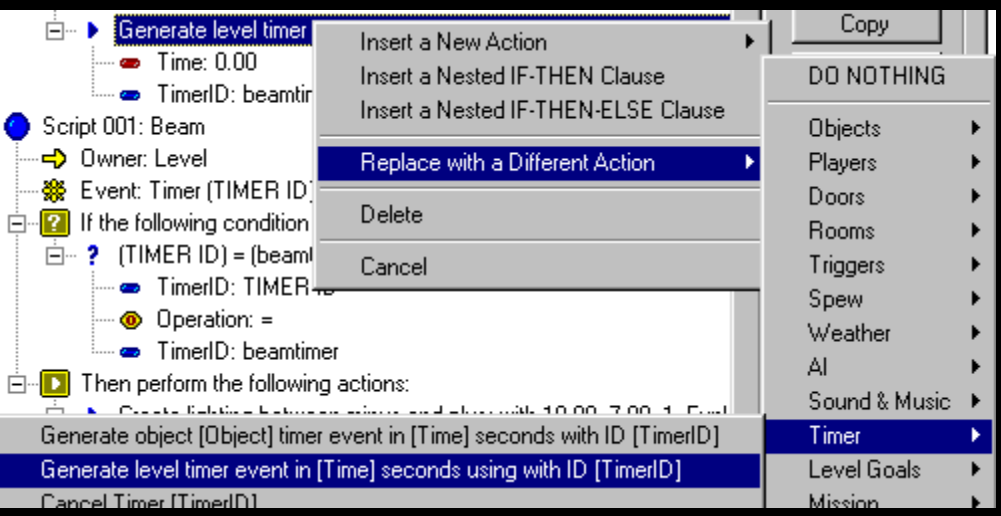
Die Dateien dazu sind wie immer in der `zip`-File.

Zu diesen Thema setze ich voraus, das Ihr Euch mit Scripting schon einmal befaßt habt. Sollte das nicht der Fall sein, dann könnt Ihr weiter vorne nachspicken. Ok, rechts zuerst einmal ein Bild, das Euch alles über das Script zeigt. Noch eine Info vorweg: zu diesem Script braucht Ihr zwei Objekte (Powerups), ich habe den **Invisiblepowerup** genommen, und habe dann das eine Objekt "`plus`" und das andere "`minus`" genannt.



Zu Punkt1:

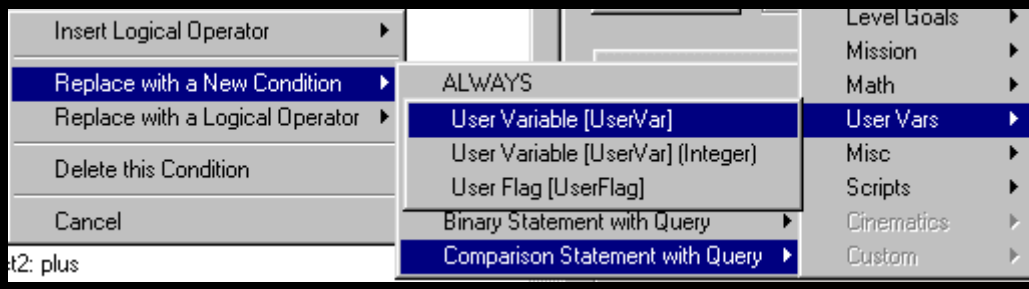
Hier wird ein Level-Timer initialisiert. Jetzt könnet Ihr Euch fragen 'wozu das denn', habe ich anfangs auch gedacht, doch wenn man den Timer nicht benutzt, bekommt man es nicht hin daß der Beam ständig läuft, weil ja mit "Lifetime" bestimmt wird, wie lange der Beam zu sehen ist und ein weiterer Effekt ist, daß man den Beam im Multiplayer nicht sehen kann (Clients). Daraufhin habe ich den Autor von "**Botstemple**" (**Bijwi**) angeschrieben der mir erklärte, daß man den Timer zu Beginn des Levels initialisieren muß, damit der Beam auch im Multiplayer zu sehen ist. Er wird einfach auf Null gesetzt, nur daß die Variable dafür da ist (Bild unten).
Um die TIMER ID anzugeben, müssen wir nun noch eine Variable erstellen, ich habe sie "`beamtimer`" genannt. Klickt auf "**Open the Users Types Workshop**", im darauffolgenden Dialogfeld wählt Ihr unter "Select the User type to edit" aus der Liste "TimerID" und klickt nun auf "**Add New Value**". Tragt dort "`beamtimer`" ein.

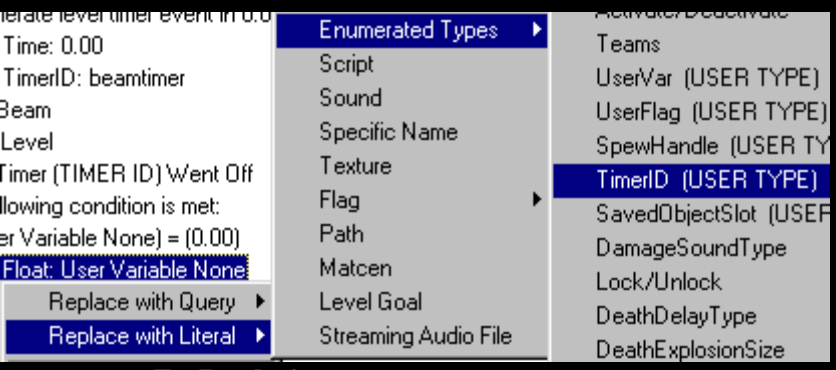


Zu Punkt2:

Ich habe hier noch einmal eine andere Möglichkeit aufgezeigt, die man damit noch machen kann. Nehmt Euch die `.zip` her und schaut es Euch an. Setzt eine Kamera, damit Ihr sehen könnt, was ich meine. In der `.zip` sind auch alle Dateien, die Ihr benötigt um Euch das Script anzusehen.

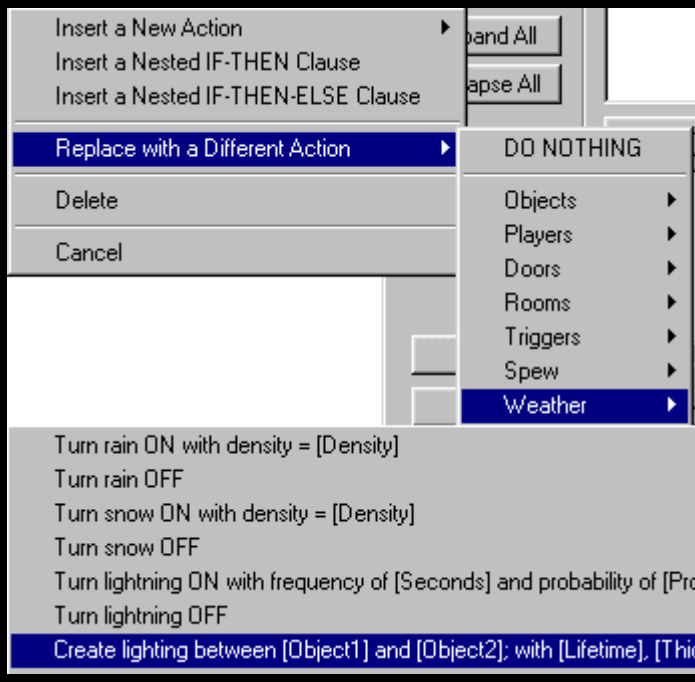
Zu Punkt3:
Es war für mich etwas schwer zu finden, deswegen erkläre ich es auch noch mal. Zuerst die im Bild rechts gezeigte Funktion einsetzen; als nächstes Rechtsklick drauf. nun könnt Ihr "TimerID" und "beamtimer" eintragen. (Bild links).





Zu Punkt4:
Jetzt kommt die eigentliche Funktion um die es hier ja geht. Wo Ihr die Funktion findet seht Ihr im Bild rechts.

In der Tabelle versuche ich einmal die Parameter zu erklären. (Bei der Übersetzung hat mir **Skorpion** sehr geholfen! **Hier noch einmal vielen Dank an Skorpion!**)



Object1	Das erste Objekt, in meinen Fall "**minus**"
Object2	Das zweite Objekt, bei mir "**plus**"
Lifetime	Hier wird die Zeit angegeben, wie lange der Beam angezeigt wird. Ich habe den Timer mit reingebaut, damit er immer läuft.
Thickness	Wie breit/dick der Beam/Textur dargestellt wird.
NumTiles	Wie viele Texturen zwischen den Objekten angezeigt werden, (NumTiles =1 ->|Textur| , NumTiles =2 -> |Textur||Textur| usw.)
Texture	Die Textur, die dargestellt werden soll, in meinen Beispiel "**FunkyEffect4**"
SlideTime	Hier bin ich mir nicht ganz sicher, Übersetzt heißt es so viel wie "*Wenn Du möchtest, daß sich die Texture entlang der Blitzoberfläche? (oder Lichtfläche / Oberfläche) bewegt, kontrolliert dies wie schnell es geht.*" Wenn ich den Wert aber auf 100 stelle, hat es fast den Eindruck der Beam würde still stehen.
TimesDrawn	Dieser Wert sollte immer 1 sein, da sonst die Framerate auf den Nullpunkt geht. Dieser Wert ist für die Sättigung zuständig. Es wird die Texture so oft wiederholen. Für einen mehr verschwommenen Effekt oder um es ein wenig heller zu machen, benutze eine Nummer größer als 1. (Bei 100 bricht die Frame total zusammen)
Red, Green, Blue	Sind die normalen RGB Farbwerte, mit denen man die Lichtfarbe der Beleuchtung der Textur angibt.
AutoTile	Bei "True" werden so viele Texturen ausgegeben, die zwischen Objekt1 und Objekt2 passen. (überschreibt das NumTiles ?)

Ihr könnt ja einmal mit den Parametern etwas rumprobieren.

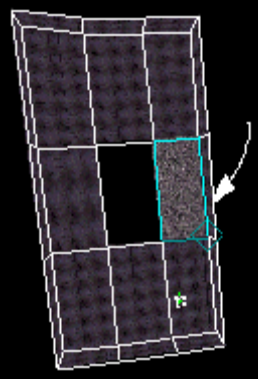
Zu Punkt5:
Hier wird der Level Timer gesetzt, den wir am Anfang initialisiert haben. Ich setzte ihn auf eine Sekunde, damit der Beam auch im Multiplayer zu sehen ist.

095 - Wind skripten

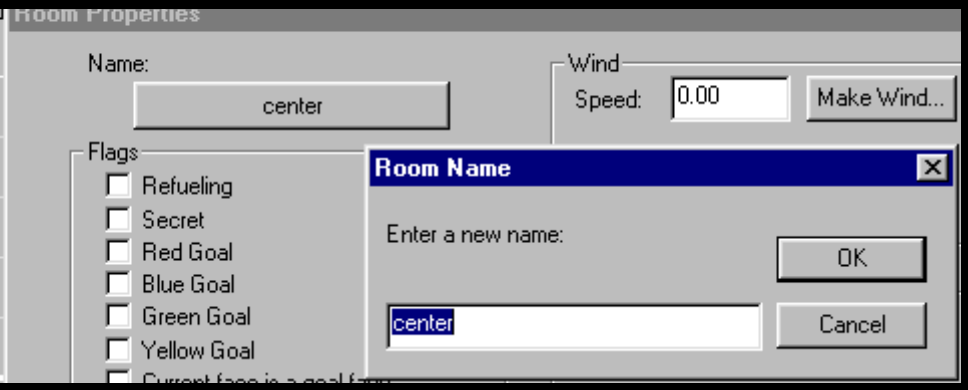
Papacat

Hier werde ich die Verwendung von Wind erklären und wie man ihn in den Level skriptet.
Zuerst einige Regeln zum Wind:

1) Wind ist eine Raumeigenschaft und wird für den gesamten Raum angewendet.
2) Er bläst in eine Richtung pro Raum, er macht keine Wirbel.
3) Du kannst Änderungen der Windrichtung oder -geschwindigkeit skripten, welchee durch ein Ereignis ausgelöst werden.
4) Eine Speed von 10 ist schneller als ein Spielerschiff.
5) Du kannst Wind in den Room Properties setzen (nicht geskriptet), aber der ist immer an und kann nicht geändert werden.

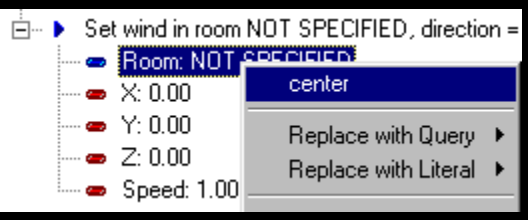


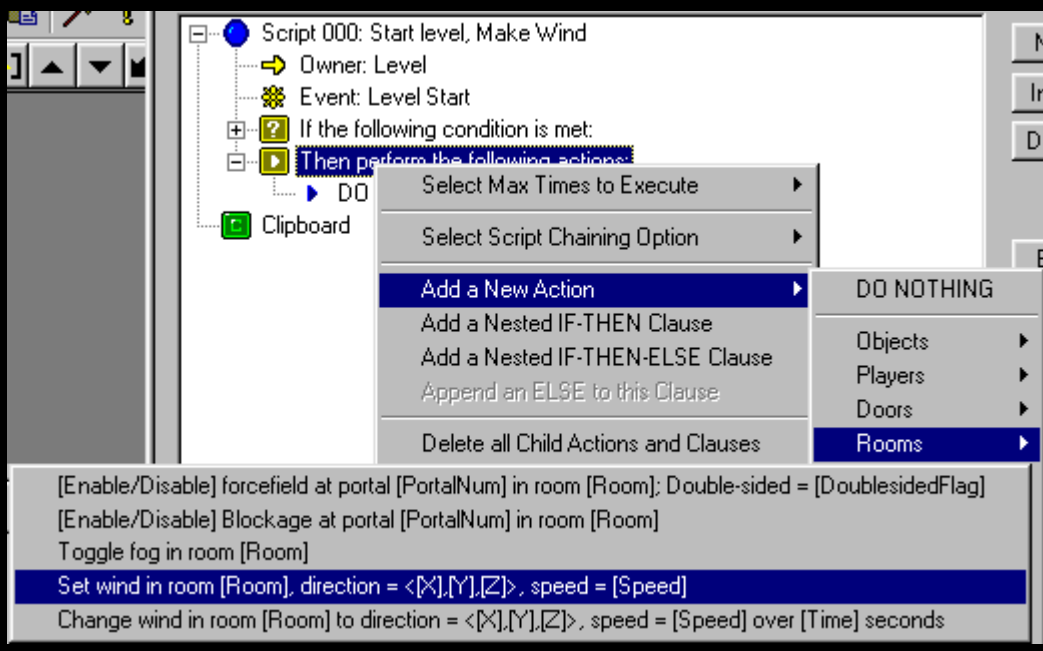
Fangen wir an. Zuerst mache ich einen kleinen Level. Er besteht aus zwei Räumen, einer ist U-förmig, der andere ist ein einfacher Quader. In diesen werden wir den Wind reinsetzen.
Der Raum muß einen Namen haben, sodaß wir auf ihn skripten können. Mache ihn zum current Room. Dann gehe in der Current Room View nach Windows->Room Properties. Im oberen linken Eck siehst Du Name und den Button, der **(none)** sagt. Benenne damit den Raum; der Name taucht dann in der Titelzeile der Current Room View auf.



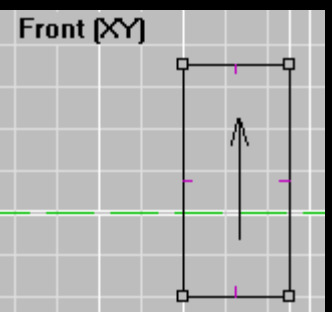
Speichere und starte Dallas. Wir werden den Wind zu Level-Beginn setzen. Gehe als erstes nach Then Perform the following actions: und Add a New Action. Wind ist eine Raumeigenschaft, gehe also nach Rooms. Wähle die Set wind in [Room]...-Action; unten rechts ein Bild wo sich die befindet.

Öffne gegebenfalls den Zweig und setze per Rechtsklick den Raum (unten links).





X, Y und Z sind Deine Richtungen. Wir werden den Wind so wie hier gezeigt setzen, indem wir Y auf 1 stellen.
Setze die Speed auf 6.



Das war's. Speichere das Skript, kompiliere es und gib's in Deine Mission.

096 - Musikregionen

Kyouryuu

Lerne wie man `.osf`-Dateien macht und Ereignis-gesteuerte Musik für Single-Player Level erzeugt. Teile dieses Tuts wurden durch die Arbeit von Wolf on Air und Shoku in diversen DescentBB-Threads inspiriert.

Einführung

Ein Trend in heutigen Computerspielen ist die Verwendung von dynamischer, ereignisgesteuerter Musik. Grundsätzlich bringt dies mit sich, daß sich die Musik entsprechend der Stimmung im Level ändert. Zum Beispiel, wenn wir durch einen leeren Korridor bummeln, könnte die Musik hintergründig und gedämpft sein. Stelle das in Gegensatz zu einem intensiven Dogfight, wo Trashmetal und E-Gitarren aus Deinen Lautsprechern schreien. Der Trick ist, von einem Track zum nächsten zu überblenden, sodaß der Effekt nicht so abrupt wie in einem alten Nintendo-Spiel ist. Descent 3 verwendet diese Methode in allen seinen Single-Player-Levels. Jeder Soundtrack eines Levels ist eigentlich in mehrere Fragmente gesplittet, die mit anderen in einer internen Playlist organisiert sind. Welcher Teil dieser Playlist zu Gehör kommt, wird durch skripten in Dallas bestimmt.

Für Dich, den Leveldesigner, ist der Effekt derselbe, aber Du mußt die Sachen etwas anders machen. Während alle D3-Playlisten hardcoded sind, mußt Du eine `.omf`-Datei erzeugen um Deine eigenen zu verwenden. Du mußt auch Deine Tracks von `.wav` nach `.osf` konvertieren, und dann das Dallas-Skripting so einrichten daß sie verwendet werden. Der Prozeß ist einfach wenn Du weißt wie, außerordentlich unklar wenn nicht. Fangen wir an.

Schnell-Referenz

Musik-Dateien für Descent 3 müssen im `.osf`-Format vorliegen. MusicTester - im D3Edit-Verzeichnis - konvertiert `.wav`'s nach `.osf`. Die .wav-Datei(en) sollten folgende Eigenschaften haben: 22050 Hz, Mono, 8-bit, PCM. Es wurde von pATChes11 angegeben, daß sie weder Mono noch 8-bit sein müssen (daß Stereo und 16-bit auch funktionieren). Allerdings unterliegt die 16-Bit-Einstellung Sample-Distortion. Sie können mit etwas so einfachem wie dem MS-Sound Recorder konvertiert werden. Es sollte noch angemerkt werden, daß schwere Bässe im `.osf`-Format fürchterlich klingen. Große Aufmerksamkeit ist aufzuwenden um sicherzustellen daß die Musik nicht 'poppt' oder überhaupt unerwünschte Effekte in billigen Speaker-Sets erzeugt.

Regeln und Regulationen

Wie ich zuvor erwähnte, bestimmt Descent 3 welche Fragmente eines Soundtracks gespielt werden sollen basierend auf einer Playliste, die als eine `.omf`-Datei gespeichert ist. In dieser definierst Du die Musik-Regionen. So eine Region besteht aus zwei Teilen - einem 'idle'- und einem 'death'-Thema. Das Idle-Thema mag aus mehreren Musik-Fragmenten zusammengesetzt sein, jedes mit seiner eigenen Anzahl an Wiederholungen. Das optionale Death-Thema kommt zu Gehör, wenn der Spieler ins Gras beißt. Nehmen wir an, ich habe zwei Musik-Regionen, eine werden wir 'Upper' nennen und eine 'Lower'. Ein Ereignis, sagen wir der Level-Start, setzt die Musik-Region auf 'Upper'. Also beginnen die Tracks in der Upper-Region zu spielen, wenn der Level beginnt. Nun nehmen wir an, daß ein Spieler später durch seinen Eintritt in einen Raum die Musikregion auf 'Lower' setzt. Die in der .omf definierten Tracks für 'Lower' beginnen nicht zu spielen, bis der gegenwärtige Track aus der 'Upper'-Region beendet ist. Die Lehre daraus ist es die Tracks kurz zu machen, um die 30 Sekunden für jeden. Die andere Konsequenz daraus ist, daß Du die Musik nicht plötzlich ändern lassen kannst. Wenn Du also zum Beispiel 'Boss music' haben willst, mußt Du einen Weg finden zum sicherzustellen daß das Hauptthema des Levels beendet ist, bevor das Boss-Thema beginnt. Andernfalls wird es sich in den Kampf mit dem Boss hineinziehen. Meines Wissens nach kannst Du die Musik nicht unmittelbar abstoppen. Descent 3 überblendet auch keine Tracks miteinander, also liegt es an Dir sicherzustellen, daß sie sanft von einem zum nächsten übergehen.

Die .omf-Datei erzeugen

Die **O**utrage **M**usic **F**ile ist eine kleine Plaintext-Datei, die eine Liste aller verfügbarer Tracks enthält und die Musik-Regionen mit Namen definiert. Du mußt verstehen daß die .omf nur zur Verwendung für das Spiel existiert. D3Edit wird sie nicht einlesen, dazu später mehr. Du kannst mit dem MusicTester .omf's erzeugen (im D3Edit-Verzeichnis). Er macht .wav->.osf-Konvertierungen und definiert auch die Musik-Regionen. Bevor Du die Konvertierung beginnst, stelle sicher daß Deine Musik-Dateien im .wav-Format gespeichert sind. Achte auch auf deren Eigenschaften (s.o.). Dann gehe nach Actions -> Convert Files, verwende den Button **Add File(s)...** um Deine Dateien hinzuzufügen. Setze den Destination Folder auf den gewünschten Ort, dann klicke **Convert Files**. Wenn die Konvertierung einer Datei schiefgeht, wurde sie höchstwahrscheinlich im falschen Format gespeichert. Die Dateikonvertierung dauert etwa die selbe Zeit, wie die Tracks lang sind. Klicke **Exit** wenn Du fertig bist.

Nun haben wir ein paar .osf-Dateien um damit zu arbeiten. Das erste was wir in unserer .osf-Datei machen müssen bevor wir Regionen hinzufügen, ist zu definieren was unsere Stream-Samples sind. Unter Actions -> Add a Group of Stream Samples kannst Du all Deine .osf-Dateien auswählen die Du gerade konvertiert hast. Wenn Du fertig bist, wirst Du sehen daß der Stream Samples-Zweig im Hauptfensterbereich nun eine Liste der .osf-Dateien hat.

An diesem Punkt wreden wir nun die Musik-Regionen definieren. Verwende die + und - neben den Namen um die Zweige aufzuklappen. Expandiere die 'Default'-Region vollständig. Du könntest auf "'Default'-Region" doppelklicken, um sie umzubenennen. Als nächstes solltest Du im Bereich 'default_idle' Subtheme (Looped) 'Samples to Play' sehen. Rechtsklicke drauf und mache Add New Subtheme Sample. Du wirst ein Pulldown-Menü der Dateien sehen, die zu früher hinzugefügt hast, neben einer Box um zu spezifizieren wie oft sie wiederholt werden. Bitte beachte daß diese gesamte Region kontinuierlich geloopt wird - Die Wiederholungen betreffen in diesem Fall nur die jeweilige .osf in der jeweiligen Region. Dadurch kann ich 'rgregion1' zweimal und 'rgregion2' einmal loopen lassen. All dies wird geloopt - bedeutet, daß ich in zwei Durchläufen 'rgregion1' viermal und 'rgregion2' zweimal hören werde. Wiederhole das sooft wie Du .osf-Dateien in dieser Region spielen lassen willst. Dasselbe kannst Du für das 'default_death'-Subtheme machen. Ich schlage vor, weder die Namen der Sub-Themes zu ändern, noch neue hinzuzufügen, da dies die Dinge unnötigerweise verkompliziert. Ignoriere auch die Rules. Wir werden unsere Regeln in Dallas definieren.

Um eine neue Region hinzuzufügen, rechtsklicke auf eine existierende und wähle Insert New Region. Beachte daß die neue Region direkt über der erscheint, auf die Du rechtsgeklickt hast. Dies mag nicht wichtig erscheinen, ist es aber. Wegen eines Bugs im MusicTester ist es entscheidend, daß die erste Region die Du definierst die erste ist die Du abspielst. Wenn zum Beispiel diese Region 'Intro' heißt, für meine Intro-Sequenz, würde ich sicherstellen müssen daß 'Intro' die erste in der Liste ist.
Wenn Du die .omf-Datei speicherst, mußt Du sie mit komplettem Dateinamen speichern. Bedeutet, daß wenn Du sie gamma.omf nennen willst Du genau **gamma.omf** in den Speichern-Dialog schreiben mußt. MusicTester hängt die Dateiendung nicht an den Dateinamen, ein anderer Bug.

Dallas dazu bringen, die Music-Regionen zu sehen

Wie zuvor erwähnt, Dallas ist blind wenn es Musik-Regionen angeht. Es liest keinerlei Informationen aus der .omf-Datei. Wenn es also Zeit wird das Skript hinzuzufügen, verwendest Du Set music Region to XXXX for all Players - aber Moment, was ist XXXX? Irgenwie müssen wir Dallas mitteilen, was die Namen der Musik-Regionen sind. Dazu mußt Du Dich manuell durch die .cpp-Datei deines Levels graben, was bei weitem eine der am meisten obskuren Facetten des D3-Leveldesigns ist.
Normalerweise wird die .cpp-Datei die von Dallas generiert wird im /osiris-Subverzeichnis des Installationsortes des D3SDK-Paketes abgelegt. Öffne sie mit Notepad oder einem anderen Text-Editor. Indem Du das Dokument manuell oder per Hand durchsuchst, halte Ausschau nach:

```
// ==================================================================
// Start of Custom Script Block - DO NOT EDIT ANYTHING BEFORE THIS
// ==================================================================
/**{CUSTOM_SCRIPT_BLOCK_START}** DO NOT EDIT! **/
```

In diesem Bereich der Datei kannst Du manuell die Namen Deiner Regionen angeben, etwa so:

```
// ==================================================================
// Start of Custom Script Block - DO NOT EDIT ANYTHING BEFORE THIS
// ==================================================================
/**{CUSTOM_SCRIPT_BLOCK_START}** DO NOT EDIT! **/

// Enter your custom script code here
/*
$$ENUM Region
0:Intro
1:Upper
2:Lower
3:Reactor
4:Escape
5:Ending
$$END
*/
```

Ich glaube, Du kannst Dir das Muster von hier an vorstellen. Wenn Du Regionen hinzufügen willst, inkrementiere einfach die Numer und füge einen Namen hinzu. Wenn Du Regionen entfernen willst mache das und stelle sicher daß die Nummern fortlaufend bleiben, mit Null beginnend. Allerdings mußt Du diese in der selben Abfolge haben, wie sie in der .omf-Datei sind. Andernfalls bekommst Du die falschen Regionen zu falschen Zeiten abgespielt und - ayayay - einen Sauhaufen.

Musik-Regionen in Dallas skripten

Das ist der einfache Teil. Erstelle einen Standard-Dallas-Trigger. Im Then perform the following actions-Bereich verwende dann Sound & Music -> Set music region to [Region] for all players. Spezifiziere die Region. Fertig. Hier ist ein Beispiel, allerdings etwas zum nachdenken. Wenn Du die Musikregion-Änderung durch Eintreten in einen Raum auslöst, magst Du wollen daß sie sich wieder zurückändert, wenn man den vorigen Raum betritt. In anderen Worten, angenommen ich habe zwei Gebäude in meinem Level. Ich sollte eine Musikregion für Gebäude 1 definiert haben, welche sich ändert wenn ich Gebäude 2 betrete. Wenn ich von Gebäude 2 nach 1 gehe, wäre es nett wenn sich die Musik entsprechend zurück ändert. Nur ein Vorschlag.

Fazit

Jedenfalls, dieses Musik-Zeug - Du machst es einmal und vergißt nie wieder wie man es macht. Es ist nicht so schwierig wie es undurchsichtig ist. Ich hoffe, mein Tut hat ein wenig Licht auf dieses Thema geworfen.

Zusätzliche Anmerkungen

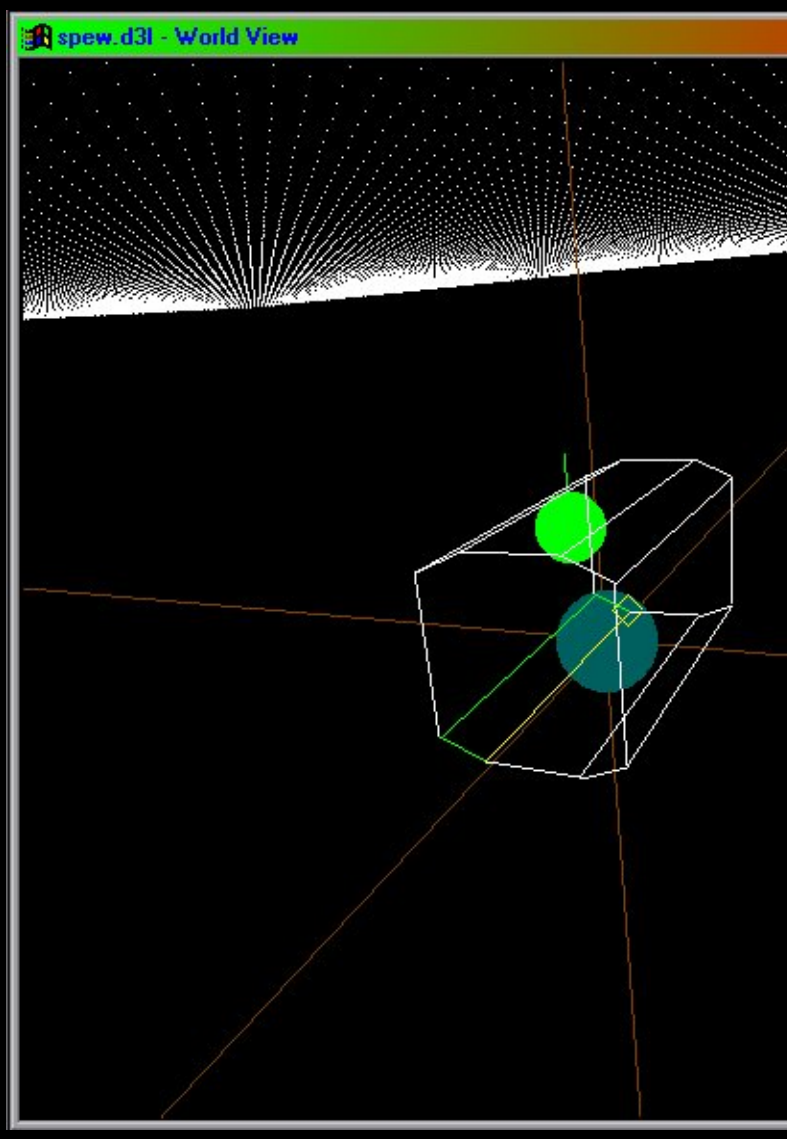
pATChes11 fügt hinzu: 'Da ist auch ein kleines Problem bei Verwendung von 8-bit Stere Audio, wo der OSF-Konverter möglicherweise abstürzt, aber es gibt einen Fix: *wenn* er crasht, öffne Deine `.wav` in einem ordentlichen Audio-Editor und lösche ein einzelnes Audio-Sample, und versuche die Konvertierung nochmals (dies muß wegen der Art und Weise gemacht werden, wie der OSF-Kompressor arbeitet)'.

097 - Spew!

Starkiller

Du wirst lernen wie man Spew in ein Level einsetzt und seine vielen Felder für Level-Effekte verwendet.

Ein neuer Level mit dem Default-Raum wird ausreichen. Setze ein Objekt des Typs forcefieldnode ins Level und benenne es `spewer`.

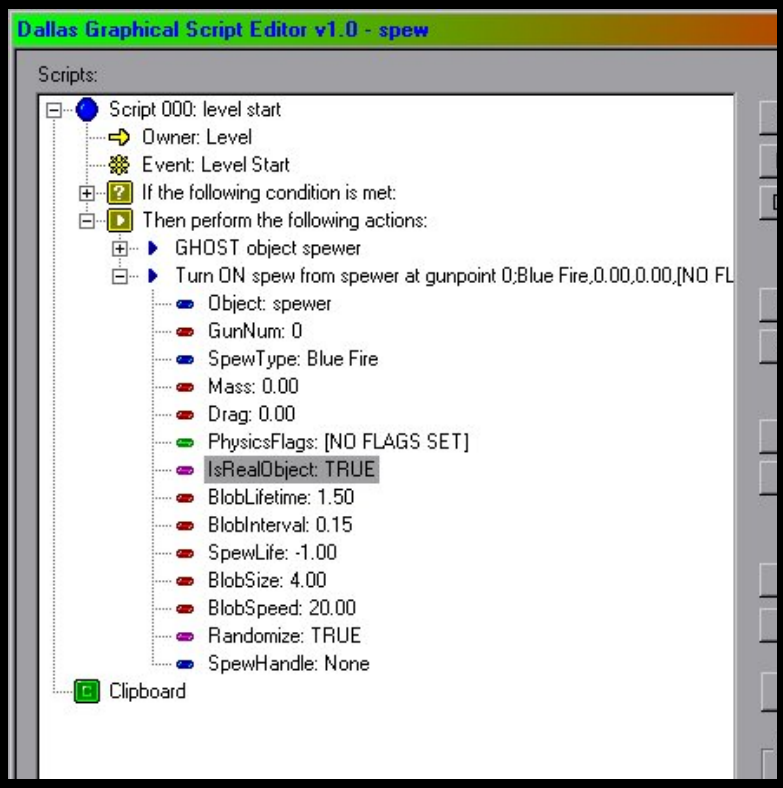


Was ist Spew überhaupt?

Ein Spew ist eine Ansammlung von kleinen 'Blobs', die... ausgespuckt werden. Jeder Blob ist grundsätzlich eine Animation, die mit einigen konfigurierbaren Variablen in die Luft geworfen wird.

Dallas

Mache ein Skript, setze den Owner auf Level, den Event auf Level Start; die Condition setze auf ALWAYS und als erstes Event 'ghoste' das Objekt spewer (Objects -> [Ghost/Unghost] object [Object]).
Füge eine neue Action aus der Spew-Sektion hinzu, Turn ON spew.... Setze das Objekt von dem gespewt werden soll auf spewer, die GunNum auf 0, den SpewType auf Blue Fire, die SpewLife auf -1 und hake alle PhysicsFlags ab. Setze IsRealObject auf TRUE. Laß die anderen Eigenschaften bei ihren Standardeinstellungen. Nun hast Du einen schmerzzufügenden Flammen-Spewer, der immer an ist.



Für experimentelle Zwecke kannst Du einige der Eigenschaften einstellen, um es anschaulicher zu machen wie Spew funktioniert. Dafür nimm das bisherige Spew-Setup und ändere das BlobInterval auf 3. Dies macht jeden Blob einfach erkennbar.

Gut, was habe ich gerade gemacht? Nun zu dem langen, langweiligen und komlexen Teil zu versuchen zu erklären, was alle diese Felder bedeuten und wie man sie verwendet...

Object	Einfach das Objekt, welches die Spew-Blobs emittiert.
GunNum	Dies ist der Gunpoint, der die Spew-Blobs ausstößt. Beachte, daß nicht alle Objekte Gunpoints haben, weswegen ich immer das forcefieldnode als Spewer verwende. Du kannst sehen ob ein Objekt Gunpoints hat, indem Du es in einem Level einfügst und in einer texturierten Ansicht betrachtest. Wenn Du kleine orange Bälle auf aus dem Objekt herausstehenden Linien siehst, hat es welche. Die Gunpoint-Bällchen werden nummeriert, sodaß Du sehen kannst wo die Gunpoints am Objekt sind.
SpewType	Welche Animation für die Blobs verwendet wird. Beachte, der Eintrag 'Napalm' der hierunter aufgelistet wird funktioniert nicht; Du mußt stattdessen Blue Fire verwenden.
Mass	Das 'Gewicht', welches den Blobs zugewiesen wird. Üblicherweise bei 0 belassen; die meisten Spews haben nicht viel Physik. Beachte: Wenn Du die Flags 'Gravity' oder 'Reverse Gravity' verwendest, beeinflußt die Mass wie stark sich die Blobs entsprechend der Gravitation bewegen.
Drag	Der Betrag an 'Bremskraft', den die Blobs haben. Höherer Drag bringt die Blobs dazu langsamer zu werden und sogar mitten in der Luft stehen zu bleiben. Meist auf 0 belassen.
PhysicsFlags	Das sind an-/abhakbare Einstellungen, die definieren helfen wie die Blobs im Spiel arbeiten und funktionieren. FixedVelocity: Heißt, daß die Blobs sich immer mit der gleichen Geschwindigkeit während ihrer Lebensdauer bewegen. Gravity: Bestimmt, ob die Blobs von Gravitation beeinflußt werden. Beachte, wenn Du keine Gravitation zum Level und keine Mass zu den Blobs zuweist, hat das Setzen dieses Flags keinen Effekt. No Collide: Bestimmt, daß die Blobs nichts treffen. Setzen dieses Flags bedeutet, daß ein Blue Fire-Spew ein Schiff nicht treffen und beschädigen kann. Meist verwendet für Dampf oder schwarzen Rauch, die nur visuellen Zwecken dienen. Reverse Gravity: Wie das Gravity-Flag, doch läßt es die Blobs entgegengesetzt zur Gravitation agieren (aufsteigender Dampf.)
IsRealObject	Das definiert, ob die Blobs echte Objekte sind oder nur Dinge zum Anschauen sind. Sehr ähnlich dem No Collide-Flag. Setze es auf TRUE, wenn Du willst daß Spieler und andere Objekte davon getroffen werden können.
BlobLifetime	Wie lange jeder Blob 'lebt'. BlobLifetime x BlobSpeed = Wie weit sich der Blob bewegt.
BlobInterval	Wie oft ein Spewer einen Blob in die Luft spuckt. Höhere Werte bedeuten weniger Blobs per Sekunde, niedrigere Werte mehr. Die Maßeinheit für BlobInterval ist 'Abstand zwischen den Blobs in Sekunden'.
SpewLife	Die Lebensdauer des Spews als Ganzes. Einfach wie lange der Spewer die Blobs von seinem 'Startpunkt' aus spuckt, oder wenn das Skript den Spew anschaltet. Setzt man die SpewLife auf -1, geht der Spewer niemals aus.
BlobSize	Die Größe der Blobs, oder wie groß die Animation ist die in die Luft geworfen wird.
BlobSpeed	Die Geschwindigkeit, mit der sich die Blobs bewegen.
Randomize	Gibt dem Spew einen leichten Offset auf das BlobInterval, was den Spew realistischer macht Immer auf TRUE lassen ☺
SpewHandle	Das ist der Name, den Du dem Spew zuweisen kannst. Dies würde von einem Skript verwendet werden, wo ein Flammen-Spew vorhanden bleibt bis ein Schalter umgelegt wird um ihn abzuschalten. Für Spews die nicht abgeschaltet werden oder Timer-kontrolliert sind, kann das leergelassen werden.

File: 098-glasstut.zip

098 - Glasbruch mit Scherben

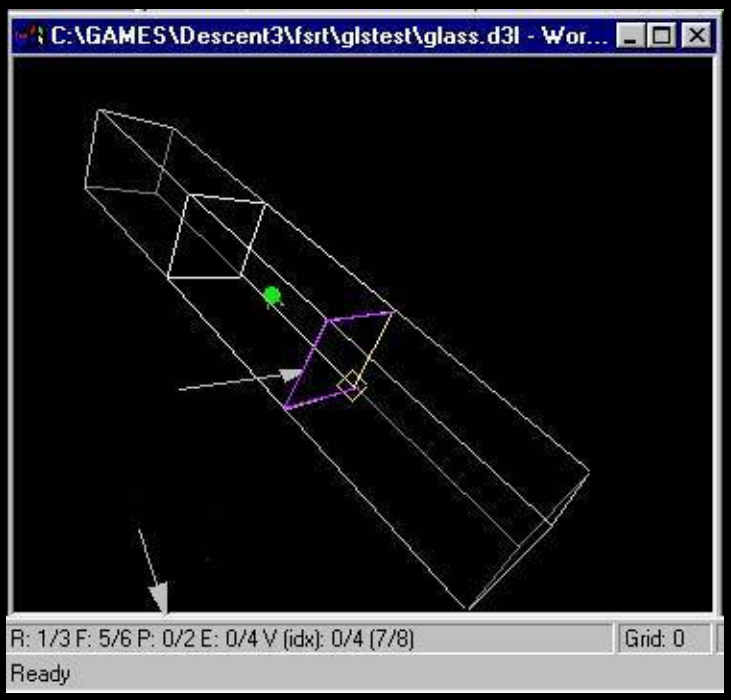
Papacat

Als Voraussetzung brauchst Du eine Textur die das Breakable-Flag hat. Wie Du sowas erzeugst, schau in der Sektion über Texturen nach.

Krag hat aufgezeigt, daß man nicht unbedingt skripten muß, um Glasbruch zu machen. Ich habe das getestet; er hat recht, doch mußt Du 'Dirty Glass' oder eine Custom-Textur mit dem Flag Breakable haben. Manche Leute hatten auch Probleme ohne Skripting (weiß den Grund nicht).

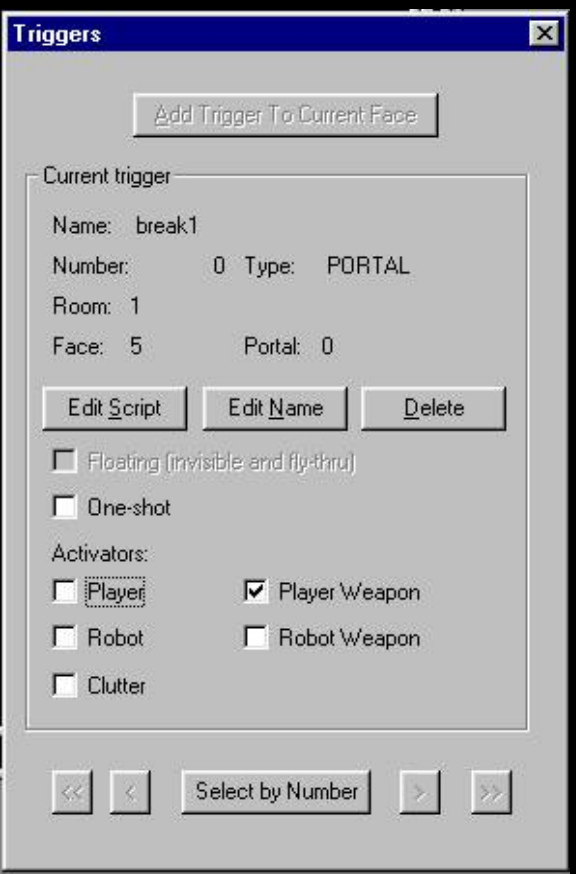
Zuallererst, (falls Du es noch nicht wußtest) Glas ist ein Kraftfeld auf einem Portal. Die Faces beider Seiten des Portals sollten als Glas texturiert sein. Fangen wir an.

Einen Trigger auf ein Portal legen



Links ist das Beispiellevel. Es besteht aus drei Räumen.
Im Bild unten links habe ich die Portal-Nummer aufgezeigt; die zeigt uns daß das current Portal #0 ist (fliederfarben) und daß der current Room insgesamt zwei Portale hat. Das mußt Du wissen, wenn es an der Zeit ist zu skripten. Drücke **P** auf deinem Keyboard und cycle damit durch die Portale des Current Rooms.
Stelle sicher daß die Texturen auf beiden Seiten des Portales Dirty Glass sind (oder eine andere mit dem Flag Breakable). Ebenso ist es wichtig, daß Du den Raum in den Room Properties benennst, meinen habe ich `center` genannt.

Um das Glas zu zerbrechen brauchen wir einen Trigger. Gehe nach Window -> Triggers, Du bekommst den unten rechts gezeigten Dialog.

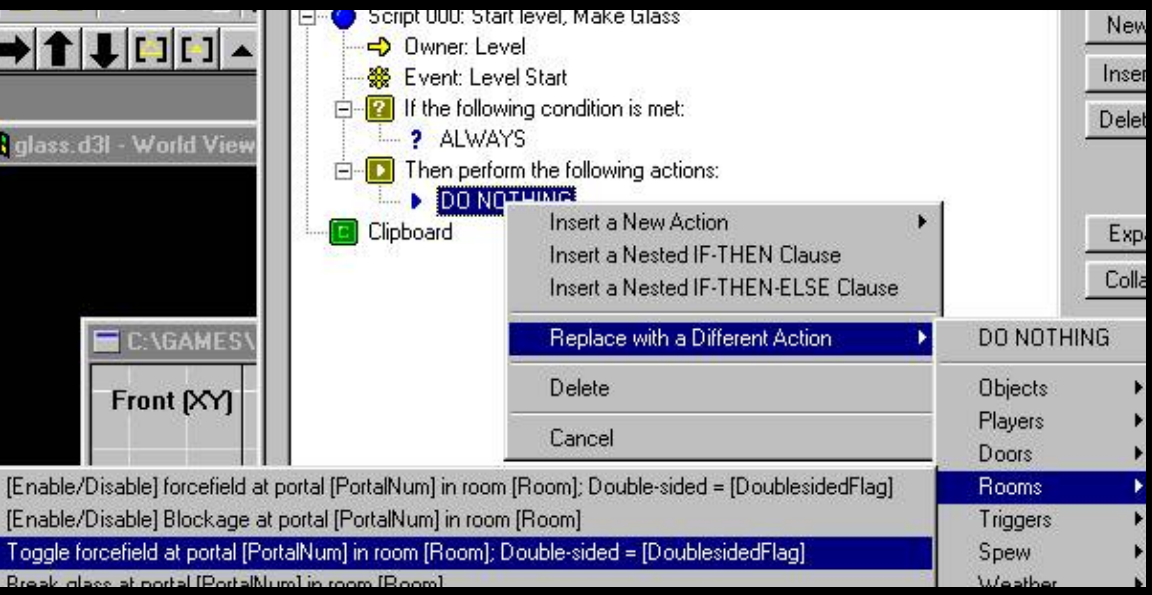


Während das Face des Portals hervorgehoben ist, klicke Add Trigger To Current Face. Im darauffolgenden Dialog benenne den Trigger, darauf wirst Du beim Skripten für den Bruch referenzieren:

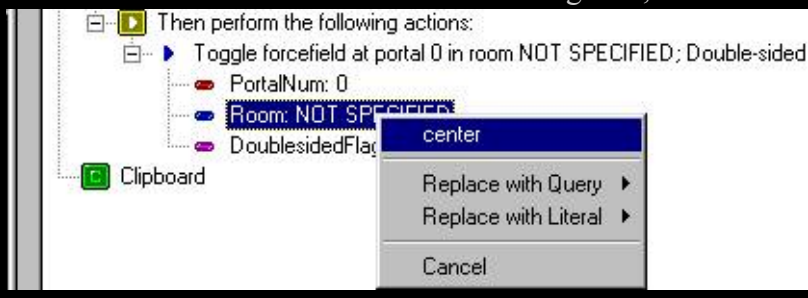
Stelle sicher daß bei Activators der Haken bei Player Weapon gesetzt ist. Dein Trigger-Dialog sollte etwa so wie links aussehen.

Das Kraftfeld im Skript aktivieren = das Glas machen

Hier sagen wir dem Skripting daß es Glas ins Portal stellen soll. Das kann ein unabhängiges Skript sein, aber für gewöhnlich wirst Du es als eine der Aktionen zum Start des Levels setzen wollen. Als erstes gehe nach Then Perform the following Actions und entweder Insert a New Action oder Replace with a Different Action wie hier gezeigt. Ich habe beides probiert, [Enable/Disable] forcefield... und Toggle forcefield... und keinen Unterschied gefunden. Wichtig ist es DoublesidedFlag zu verwenden und es auf TRUE zu setzen. Dadurch mußt Du das ganze nur für eine Seite des Portales machen.

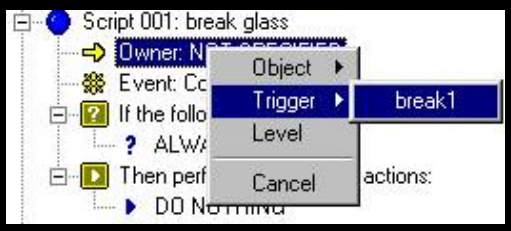


Um zu den Spezifikationen der Action zu gelangen expandiere den Zweig. Rechtsklicke auf PortalNum, wähle Enter Integer Value und setze 0 ein. Als nächstes rechtsklicke Room: um anzugeben, für welchen Raum wir uns mit der PortalNum beziehen. Im erscheinenden Menü sind alle Räume mit zugewiesenen Namen aufgelistet. Falls Du vergessen hast, den Raum zu benennen, mache das noch vorher. Das DoublesidedFlag setze noch auf TRUE. Damit ist dein Glas eingebaut.



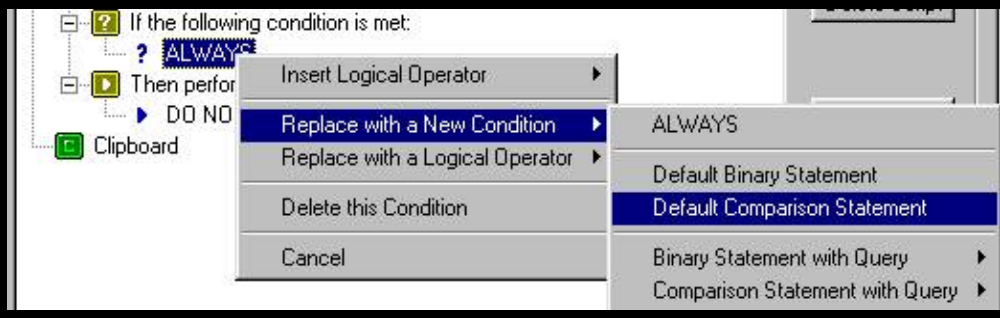
Den Bruch skripten

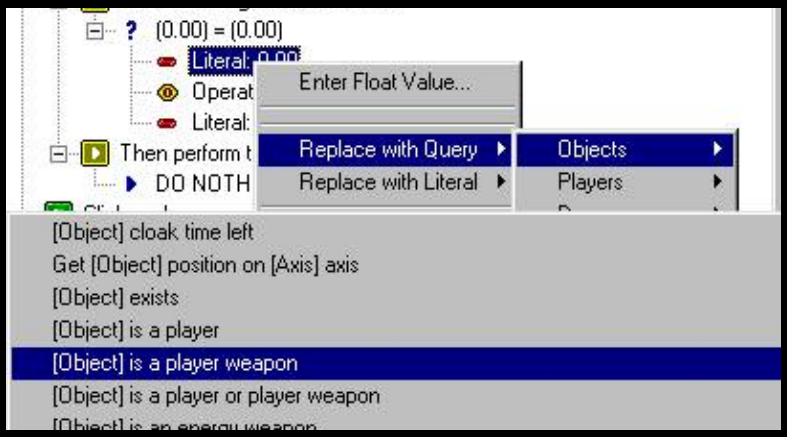
Mache ein neues Skript und benenne es.



Als erstes setzen wir den Owner des Skriptes auf den Trigger. Dazu rechtsklicke ihn, und Du bekommst eine Liste aller Trigger in Deinem Level. Wenn Du den Owner auf einen Trigger setzt, ändert sich der Event automatisch zu Activated by (IT).
Wir wollen, daß unser Glas nur unter bestimmten Umständen bricht. In diesem Fall ist das: 1.: es ist eine Player-Waffe, und 2.: es ist KEINE Energiewaffe.

Als erstes ändern wir ALWAYS zu einem Comparison Statement. Wir vergleichen ob IT eine Spieler-Waffe ist oder nicht.

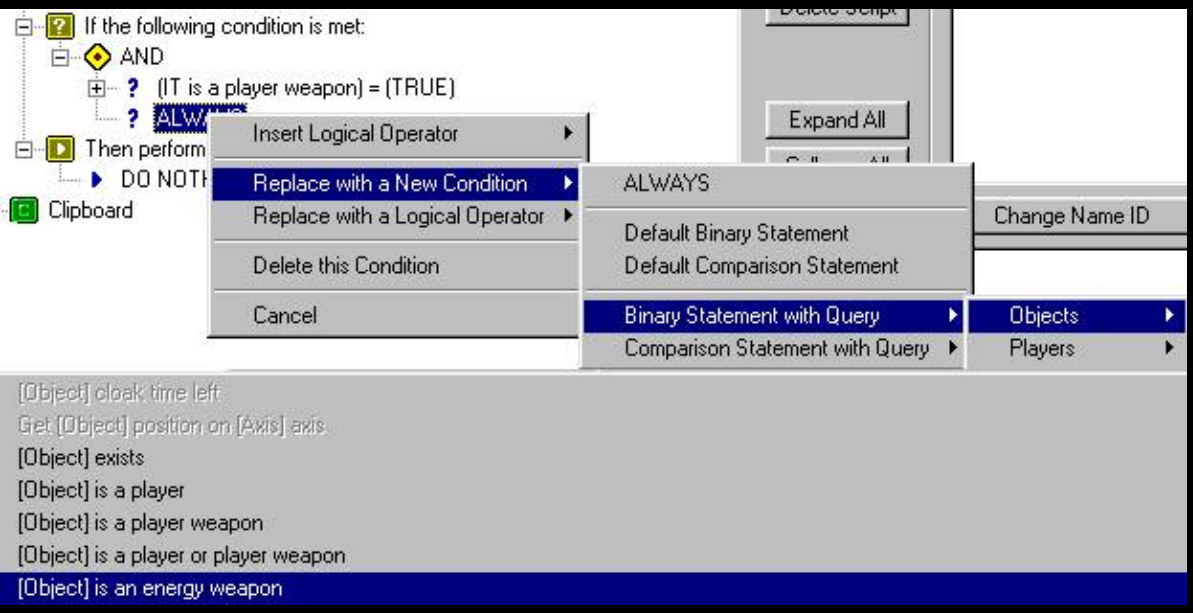




Rechtsklicke auf Literal und setze es entsprechend dem Bild links, um zu prüfen ob IT eine Spielerwaffe ist; es ändert sich daraufhin in Bool. Ändere gegebenenfalls das Bool nach Operation zu TRUE. Das Resultat ist (IT is a player weapon) = (TRUE).

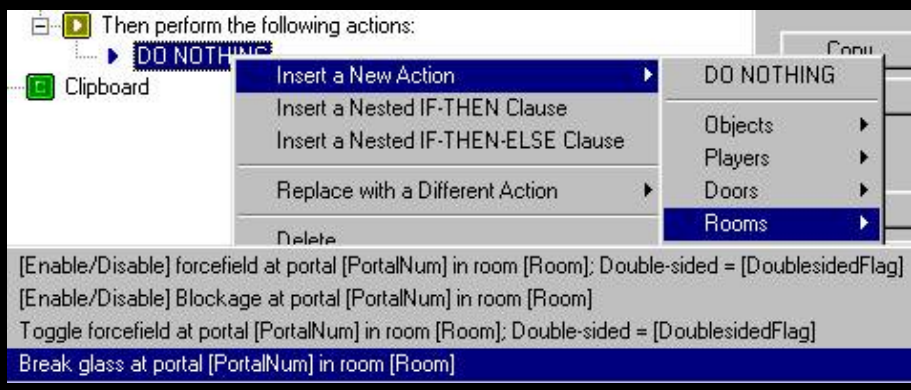
Zur Zeit würde das Glas brechen, wenn es von irgeneinder Spieler-Waffe getroffen würde. Wir wollen es nur Non-Energy-Waffen (Vauss, Concussions, etc...) brechen lassen. Rechtsklicke auf (IT is a player... und setze einen AND-Operator ein.

Unten siehst Du wie sich das AND eingliedert und die zwei Bedingungen,die existieren müssen um das AND zu erfüllen. Man kann das AND einfügen, bevor man die Bedingungen angibt; es wären einfach nur zwei ALWAYS-Statements zum Ändern verfügbar. Wenn man eine dritte Prüf-Bedingung einsetzen will, fügt man einfach weitere AND oder OR anstatt einer Bedingung hinzu, und es verzweigt sich weiter.



Ändere das verbleibende ALWAYS nach Binary Statement with Query -> (Object) is an Energy Weapon wie hier links. Setze die Operation nach FALSE.

Alles was noch bleibt, ist das Glas zu zerbrechen. Setze gemäß dem Bild rechts die Action und setze die [PortalNum] und [Room] auf die selbe Weise, wie Du es bei Toggle forcefield... weiter oben gemacht hast.



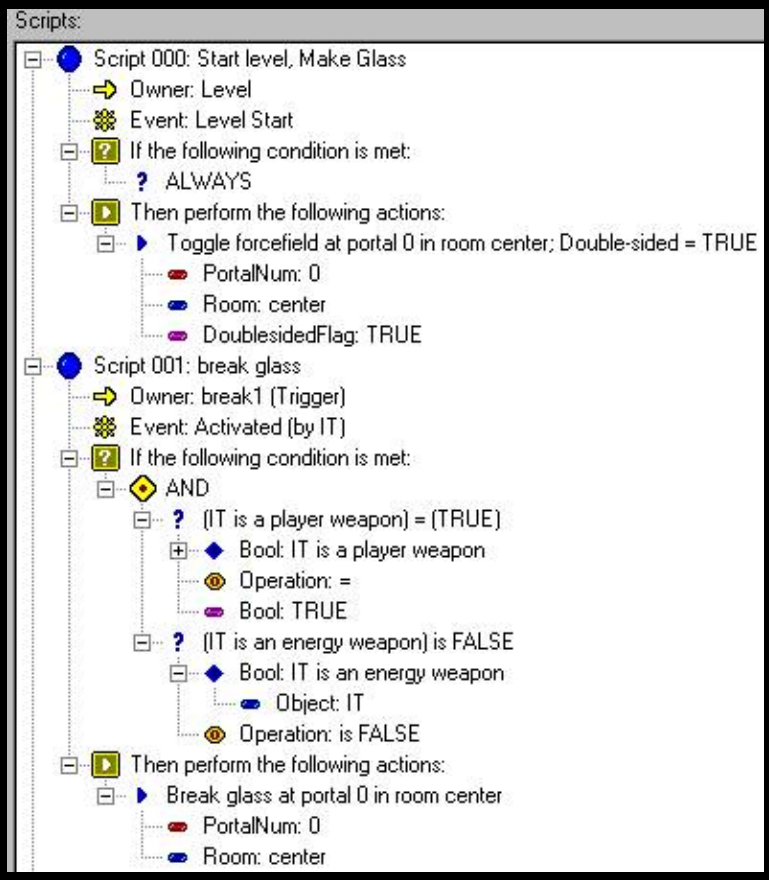
Deine zwei Skripten sollten nun so wie hier rechts aussehen.

Speichere/Kompiliere sie.

Obwohl man dem geskripteten Bruchglas die Eigenschaft Spiegel zuordnen kann, ist dies nicht zu empfehlen. Es entsteht ein Effekt ähnlich dem, wenn man zwei Faces übereinander hat (=Duplicate Faces).

Das Glas bricht, aber der Spiegel bleibt stehen.

Congrats! Nun hast Du Glas in Deinem Level, das beim Brechen Scherben macht.



099 - Einen Schalter skripten I

WillyP

Credits

Atan schrieb das meiste dieses Tuts als Antwort auf eine Frage von Nightmare im Descentforum.de. Dank an beide.

Fangen wir an

Mache ein einfaches Level, zwei kleine Räume getrennt durch eine Tür. Baue irgendwo im ersten Raum einen Schalter ein, dort wo der Player-Star ist. Einer des Typs Handleswitch wäre am besten. Benenne ihn `MySwitch`. Verifiziere Dein Level und speichere es unter `switchtest`. Testfliege Dein Level und halte nach Deinem Schalter Ausschau. Darauf schießen bewirkt nichts, es ist nur um sicherzustellen daß Dein Level Ok ist. Teste auch die Tür. Gehe in die World View und starte Dallas.

1. Schritt - Erstes Skript

->New Skript
Script000: benenne es `PlaySound`
Owner: MySwitch (Name des Schalters, Rechtsklick -> Objects -> Clutter -> MySwitch)
Event: Collided with (IT)
If The following Condition is met:
ALWAYS
Then perform the following actions:
Play 2d Sound Bell for Player object IT Volume 100% (Oder einen beliebigen anderen Sound, außer ENV-Sounds)
DO NOTHING

Dallas arbeitet bis '`Done`' und produziert - wenn alles gut ging - die Dateien `switchtest.dll`, `-.cpp`, `-.o` und `-.msg`, wenn Du Dein Level `switchtest` genannt hast (im Verzeichnis `<SDK>\osiris`). Packe die Skriptdateien mit in Deine .mn3 und testfliege Dein Level. Wenn Du den Schalter anschießt oder berührst, solltest Du den Sound hören. Das muß hinhauen bevor wir weitermachen! Als nächstes werden wir den Schalter animieren. Ich empfehle wärmstens, nach jedem Schritt einen Testflug zu unternehmen. Funktiniert? Gut! Weiter!

2. Schritt - Den Schalter animieren

Das machen wir mit collide; füge dem vorigen Skript diese Action hinzu:
Play object MySwitch animation from frame 0 to 20, cycle time = 1.00, looping=FALSE

Nicht vergessen zu speichern/kompilieren. Verwende Quicktest zum testfliegen, dann mußt Du nicht jedesmal die Skript-Files manuell einfügen.
Ein Treffer erzeugt nun Sound und Bewegung. Beachte, die Animationsgeschwindigkeit in dieser Action wird mit cycle time bestimmt; erhöhe sie beispielsweise auf 5.0 und schau wie es aussieht.

3. Schritt - Über eine Bedingung die Schalterreaktion auf Spielerwaffen-Treffer limitieren

Ihr kennt doch den lärmenden Effekt, wenn man auf einen Schalter drauffliegt?
Ersetze das ALWAYS in der ...following Condition... durch:
(IT is a player weapon) = (TRUE)

4. Schritt - Eine Nachricht anzeigen lassen

Erzeuge in Dallas eine neue Message und benenne sie `ShowHitText`. Schreibe etwas wie "Switch is Activated!" in das Fenster unterhalb von ChangeNameID.
Füge folgende Action hinzu: Show HUD message ShowHitText
Zusätzlich zum Sound und der Ani wird jetzt auch am HUD was angezeigt.
Anmerkung für Leute die Sonderzeichen wie Umlaute wollen/brauchen: ASCII-Code verwenden (etwa Alt+246 (÷) für ein 'ö')!

5. Schritt - Den Schalter verwenden, um eine Tür zu aktivieren

Füge folgende Action hinzu:
Activate Door MyDoor
Nun erfüllt unser Schalter einen Zweck.

6. Schritt - Türe öffnen wenn der Schalter getroffen wird

Falls das Auto Close-Flag der Tür gesetzt ist, entferne dieses jetzt.
Ändere die zuvor hinzugefügte Action in Set door MyDoor position to 100.0%

Du siehst den Unterschied? Zuvor öffnete/schloß sich die Tür nun öffnet sie sich nur.

7. Schritt - Die Tür per Skript versperren

Wie können wir verhindern daß sich die Tür bei direkten Treffern öffnet, sodaß nur der Schalter sie öffnen kann? Irgendwelche Ideen? Tip: Wir müssen etwas beim Event Level Start machen. Activate betätigt die Tür, welche sich automatisch wieder schließt wenn das Auto Close Flag im Türen-Paneel gesetzt ist. (Anmerkung: verwende keine 'blastable'-Tür hier.) Set position to 100% bedeutet, daß sich die Tür nur öffnet. Das Auto Close muß dafür abgecheckt sein; sonst schließt sie sich ohne ein neues Event wieder. Stelle sicher daß alles wie oben angegeben funktioniert und mache gegebenenfalls Einstellungen. Bei Level Start können wir das Verhalten der Tür einstellen. Füge ein neues Skript hinzu:

Script001: Start
Owner: Level
Event: Level Start
If the following Condition is met:
ALWAYS
Then perform the following actions:
LOCK door MyDoor

Dies sollte die Tür für Dich per Skript versperren. Versuche die Tür direkt zu öffnen - sie sollte versperrt sein. Schieß auf den Schalter. Zusätzlich zu Sound, Animation und der HUD-Nachricht wird sich die Tür öffnen und so bleiben. Wenn nicht, fixe das Problem bevor Du weitermachst.

8. Schritt - Einen Timer zum Schließen verwenden

Nun werden wir die Tür über ein Timer-Ereignis zumachen. Öffne den **User Type Workshop** -> TimerID -> **Add a new Value**, nenne ihn `CloseDoor`.
Füge nochein Skript hinzu:

Script002: Close Door
Owner: Level
Event Timer (TIMER ID went off)
If the following Condition is met
(CloseDoor) = (TIMER ID) (Erreichbar über Default comparison statement, Literal: Replace with literal ->enumerated types -> Timer ID; dann über Rechtsklick CloseDoor und TIMER ID setzen)
Then perform the following actions:
Set door MyDoor to 0.0%

Ok, dieses Skript sieht nach einem Level-Timer-Event namens CloseDoor und setzt dann die Tür-Animation auf 0.0 (=schließt die Tür). Aber bevor das funktioniert, was müssen wir machen? Genau, wir müssen zuerst einen Level-Timer starten,

9. Schritt - Einen Level-Timer hinzufügen

Dazu fügen wir unserem wohlbekannten Skript000 folgende Action hinzu:
Generate level timer event in 10 seconds using with ID Close Door.

Die Tür sollte sich nun öffnen und nach zehn Sekunden wieder schließen. Funktioniert's bei Dir?

Nun, was ist mit multiplen Treffern am Schalter etwa mit Vauss oder Laser? Könnte das unsere Idee gefährden? Hmm, welche Idee? Ok, das Ziel ist daß bei bei Beschießen des Schalters sich die Tür nur einmal öffnet und dann wieder schließt. Die nächsten Treffer sollten die Tür nicht öffnen, Du hattest Deine Chance ein Goodie zu erwischen (oder vielleicht im zweiten Raum eingeschlossen zu werden ☺). Irgendeine Idee wie man das erreicht? Wir werden das diesmal über zwei verschiedene Methoden versuchen. Funktioniert der Timer bei Dir wie beschrieben? Wenn ja, können wir weitergehen, eine Action zu limitieren.

Das einfache zuerst. Rechtsklicke auf Then perform the following actions, Wähle Select max times to execute -> Once.
Dies tut was wir wollen. Wo sollen wir das einstellen? Denke daran, das Ziel ist es daß sich die Tür nur einmal öffnet.

Zweite Methode, mit ein wenig mehr I/O diesmal, genau hinschauen!
Wichtig: Ändere das Once wieder zurück auf Run Infinitely!

Erzeuge eine neue Nachricht mit dem Text `%d Chances left`. Benenne sie `TellMe`. Verwende den **User Type Workshop** um eine neue UserVar zu erzeugen, benenne die `Count`. Dann füge eine Action in Skript001 hinzu:

Set user var Count to 5.0

mache folgende Änderungen in Skript000: es kommen zwei Actions hinzu,

Decrement user var Count
Show HUD message TellMe (User Variable Count (Integer))

Ok, das ist ein wenig kompliziert, also versuche ich Dir mal zu erklären was passieren sollte: Zu Level-Beginn hatten wir eine Variable mit 5 voreingestellt. Wenn wir den Schalter anschießen, sieht das Skript nach ob das eine Spielerwaffe war und die Variable größer als 0 ist. Wenn diese Bedingungen erfüllt sind, startet das Skript die uns bekannten Aktionen und verringert die Variable (nun 4), zeigt dann den Text '4 Chances left'. (das `%d` ist ein Platzhalter; `%f` würde einen Gleitkommawert anzeigen. Siehe dazu Formatierungs-Angaben für C.)
Das ist ein großartiger Weg Zähl-Aktionen in D3 anzuzeigen! Wenn unsere Variable Null erreicht, zeigt das Skript '0 Chances left' aber jeder folgende Schalter-Treffer wird vom Skript nun ignoriert werden (da die Bedingung nicht mehr zutrifft...)
Probiere es aus bis es bei Dir wie beschrieben funktioniert. Paß auf, daß Du den Schalter nur einmal triffst um ihn auszulösen, das heißt verwende Flares. Dann versuche was passiert wenn Du mit dem Laser auf den Switch feuerst. Welche Probleme treten in diesem Fall auf? Wie können wir das lösen?

100 - Einen Schalter skripten II

WillyP

Wir verblieben mit einer Frage... und einer Herausforderung!

Laß uns das ganze aufpolieren!

Nun gut, das war das Ende des Threads. Doch wir hatten noch ein Problem. Wenn wir testfliegen und mit Flares - einer nach der anderen - schießen, funktioniert unser Skript perfekt. Aber wenn wir mit Laser feuern, verbrauchen wir unsere ganzen Chancen mit einem schnellen Burst von Laserfeuer. Die HUD-Nachricht besagt 0 Chancen und nach zehn Sekunden ist die Tür zu. Warum tut sie das? Sehen wir unser Skript an um die Antwort zu finden.

Im Skript000 machen wir eine Reiche an Überprüfungen, dann führen wir eine Reihe von Aufgaben aus. Das Skript startet und läuft jedesmal, wenn eine Kollision mit einer Spieler-Waffe auftritt, bis `Count` 0 erreicht. Wenn wir also schnell schießen, läuft das Skript fünf mal in schneller Abfolge. Daher ist unser Vorrat an Chancen schnell aufgebraucht. Ist das der gewünschte Effekt? Nicht wirklich! Wir müssen noch etwas Arbeit leisten. Präzisieren wir unser Ziel: Die Tür bleibt pro Versuch für zehn Sekunden offen. In Skript000 können wir prüfen, daß die Tür geschlossen ist bevor wir dem Skript erlauben ausgeführt zu werden. Rechtsklicke unter If the following condition is met auf das AND und füge eine neue Bedingung hinzu, um zu prüfen daß die Türposition 0.0% ist (=geschlossen): (Door MyDoor position) = (0.0%)

Nun gehe testfliegen. Wir sehen, daß wenn wir den Schalter anschießen sich die Tür öffnet. Wenn diese offen ist, passiert nichts wenn wir den Schalter beschießen. Aber wenn Du den Schalter mit beiden Lasern zur selben Zeit triffst, läuft das Skript zweimal bevor sich die Tür öffnet. Also verlieren wir einen unserer Versuche. Das Skript läuft in zwei Instanzen, zeitmäßig so knapp beieinander, daß bevor die erste Instanz die Tür zu öffnen beginnt die zweite sich bereits nach dem Punkt befindet wo sie prüft ob die Tür sich zu öffnen begonnen hat. Also müssen wir in der ersten Instanz eine Bedingung setzen, welche die Ausführung der zweiten blockiert. Laß uns eine UserFlag hinzufügen. Gehe in den **User Types Workshop** und erzeuge eine UserFlag namens `flagit`. Im Script000 füge die Bedingung hinzu (User Flag flagit) = (FALSE). Nun wird Skript 000 nicht ausgeführt, außer `flagit` ist FALSE. Diese neue Zeile sollte die letzte unter dem AND sein. Nun füge eine neue Aktion unter Then perform... hinzu, ebenfalls in Script000. Errätst Du was das für eine Action ist? Wir müssen `flagit` hier auf TRUE setzen: Set user flag flagit to TRUE. Ziehe es an den Anfang der Liste. Als nächstes gehe zu Skript 002 und füge unter Then perform... eine neue Action hinzu: Set user flag flagit to FALSE. Außerdem habe ich die Zeilen Decrement user var Count und Show HUD message TellMe nach Script002 bewegt. Und ich habe die Action Show Timer CloseDoor on HUD in Script000 hinzugefügt. Dieses sieht nun so aus:

```
Script000: PlaySound
        Owner: MySwitch
        Event: Collided (With IT)
        If the following condition is met:
        AND
                (IT is a player weapon)= (TRUE)
                (User Variable Count) > (0.0)
                (Door MyDoor position) = (0.0%)
                (user Flag flagit) = (FALSE)
        Then perform following actions:
                Set user flag flagit to TRUE
                Play 2D Sound Bell for player object IT Volume 100%
                Play object MySwitch animation from frame 0 to 20, cycle time = 5.00, looping=FALSE
                Set door MyDoor position to 100%
                Generate level timer event in 10 seconds using with ID CloseDoor
                Show Timer CloseDoor on HUD
```

Script001 hat sich nicht geändert, und Script002:

```
Script002: Close Door
        Owner: Level
        Event: Timer(TIMER ID) Went Off
        If the following condition is met:
                (CloseDoor) = (TIMER ID)
        Then perform the following actions:
                Set door MyDoor position to 0.0%
                Decrement user var Count
                Show HUD message TellMe (User Variable Count (integer))
                Set user flag flagit to FALSE
```

Prüfe Deine Skripten sorgfältig!
Testfliege. Unser Skript arbeitet diesmal perfekt. Show Timer zeigt uns die Zeit bis die Tür sich schließt... wir würden nicht im zweiten Raum eingeschlossen werden wollen. Ich überlasse es Dir, dafür einen Grund anzubieten!

Troubleshooting

Wenn Dallas funktioniert Du aber noch immer Probleme mit Deinen Skripten hast, gehe bitte die vorigen Schritte nocheinmal durch. Jedwede Variation kann unerwartete Resultate zeitigen. Nachdem Du sie zum funktionieren gebracht hast, kannst Du ins Dallas gehen und damit herumspielen.

Wenn alle Stricke reißen...

Ich schlage vor, daß dies die letzte Lösung ist, da Du nichts lernen wirst indem Du ein Skript kopierst. Aber wenn Du mit allem Anderem das Skript nicht zur Funktion bringst, ist hier ein Screenie des Skriptes.

- Script 000: Play Sound
  - Owner: MySwitch (Object)
  - Event: Collided (with IT)
  - If the following condition is met:
    - AND
      - (IT is a player or player weapon) is TRUE
        - Bool: IT is a player or player weapon
          - Object: IT
        - Operation: is TRUE
      - (User Variable Count) > (0.00)
        - Float: User Variable Count
          - UserVar: Count
        - Operation: >
        - Literal: 0.00
      - (Door MyDoor position) = (0.0%)
        - Percentage: Door MyDoor position
          - Door: MyDoor
        - Operation: =
        - Percentage: 0.0%
      - (User Flag flagit) = (FALSE)
        - Bool: User Flag flagit
          - UserFlag: flagit
        - Operation: =
        - Bool: FALSE
  - Then perform the following actions:
    - Set user flag flagit to TRUE
      - UserFlag: flagit
      - True/False: TRUE
    - Play 2D Sound Bell for player IT, volume = 100.0%
      - Sound: Bell
      - PlayerObject: IT
      - Volume: 100.0%
    - Play object MySwitch animation from frame 0 to 20, cycle time = 5.00, looping = FALSE
      - Object: MySwitch
      - StartFrame: 0
      - EndFrame: 20
      - CycleTime: 5.00
      - Looping: FALSE
    - Set door MyDoor position to 100.0%
      - Door: MyDoor
      - Position: 100.0%
    - Generate level timer event in 10.00 seconds using with ID CloseDoor
      - Time: 10.00
      - TimerID: CloseDoor
    - Show Timer CloseDoor on HUD
      - TimerID: CloseDoor
    - DO NOTHING
- Script 001: Start
  - Owner: Level
  - Event: Level Start
  - If the following condition is met:
    - ALWAYS
  - Then perform the following actions:
    - LOCK door MyDoor
      - Lock/Unlock: LOCK
      - DoorName: MyDoor
    - Set user var Count to 5.00
      - UserVar: Count
      - value: 5.00
    - DO NOTHING
- Script 002: Close Door
  - Owner: Level
  - Event: Timer (TIMER ID) Went Off
  - If the following condition is met:
    - (CloseDoor) = (TIMER ID)
      - TimerID: CloseDoor
      - Operation: =
      - TimerID: TIMER ID
  - Then perform the following actions:
    - Set door MyDoor position to 0.0%
      - Door: MyDoor
      - Position: 0.0%
    - Decrement user var Count
      - UserVar: Count
    - Show HUD message TellMe (User Variable Count (Integer))
      - Message: TellMe
      - Integer: User Variable Count (Integer)
        - UserVar: Count
    - Set user flag flagit to FALSE
      - UserFlag: flagit
      - True/False: FALSE
    - DO NOTHING
- Clipboard

File: 101-ScriptedCloak.zip

101 - Scripten einer Inventory-Cloak

Starkiller

Ich hab' Euch den gesamten Level beigepackt. Wenn ihr nachspicken oder Euch das Selbstbauen der Objekte ersparen wollt, extrahiert sie aus dem Beispiellevel.

Dieses Skript war nicht für diese Seite (http://gameedit.net/custom/GameEdit_OG/index.html), sondern für Skyalmian, für sein Projekt "Lunar Outpost MN0012" als er ein Cloaking Device für das Schiff brauchte, wie in dem Roman "Descent" (obwohl die Dallas-Screenies aus seinem Skript für Sein Level "Space Station Kreathlan" sind, wo er die Verwendung des Cloaking Devices zum Einsatz brachte als es dieses nicht in den "Lunar Outpost MN0012" schaffte). Dieses Skript ist sehr fortgeschritten und nicht im geringsten für Anfänger, da nicht alles erklärt wird (es wird vorausgesetzt daß Du komplett mit Dallas umgehen kannst). Skyalmian hatte eine schwere Zeit beim Versuch dies in Dallas zu duplizieren, als es ihm übergeben wurde.

Dieses sogenannte Tutorial wird Dir zeigen, was gemacht werden sollte um eine Tarnvorrichtung zu erschaffen, die von jedem Schiff verwendet werden kann und wird allgemein erklären, was jedes Skript tut. Insgesamt sollte es zwischen 30 Minuten und einer Stunde dauern, abhängig davon wie gut Du mit Dallas umgehen kannst.

01 - User Types

Das erste was Du machen solltest wenn der Dallas Graphical Script Editor offen ist, öffne den User Types Workshop und erzeuge folgende User Types:

SavedObjectSlot: `Cloaked_Handle`
TimerID: `Cloak_Energy_Drain`
UserFlag: `Cloaked_Flag_1`
UserFlag: `Cloaked_Flag_2`
UserVar: `Useless`

Das Cloaked_Handle ist die Identifikation des Spielerschiffes, welches die Cloak hat, der Cloak_Energy_Drain ist für den Timer, der verwendet wird um dem Schiff Energie für die Tarnvorrichtung abzuziehen, die Cloaked_Flag_# sind unverzichtbar wichtig für die Zustände der Tarnung, und Useless ist nur nötig um ein Leerzeichen für das erste Skript einzufügen.

02 - Nachrichten

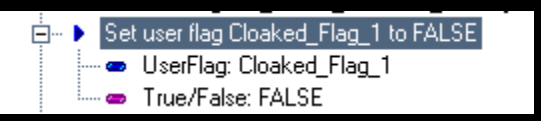
Erzeuge diese Nachrichten in der Message List (Name -> `Textstring`):

Cloaked_No_Energy_Left -> `Not enough energy to use the Cloaking Device.`
Cloaking_Device -> `Cloaking Device`

03 - Das Powerup

Erzeuge ein Powerup im Level mit dem Namen Cloaking_Device und setze es knapp an, wenn nicht in den ersten Spielerstartpunkt. Wähle ein gutes, solides Objekt, da die Spieler das Model sehen werden wenn der Spieler draufgeht, der das Cloaking Device hat. Skyalmian hat für seine Version das Model der Data Cartridge gewählt.

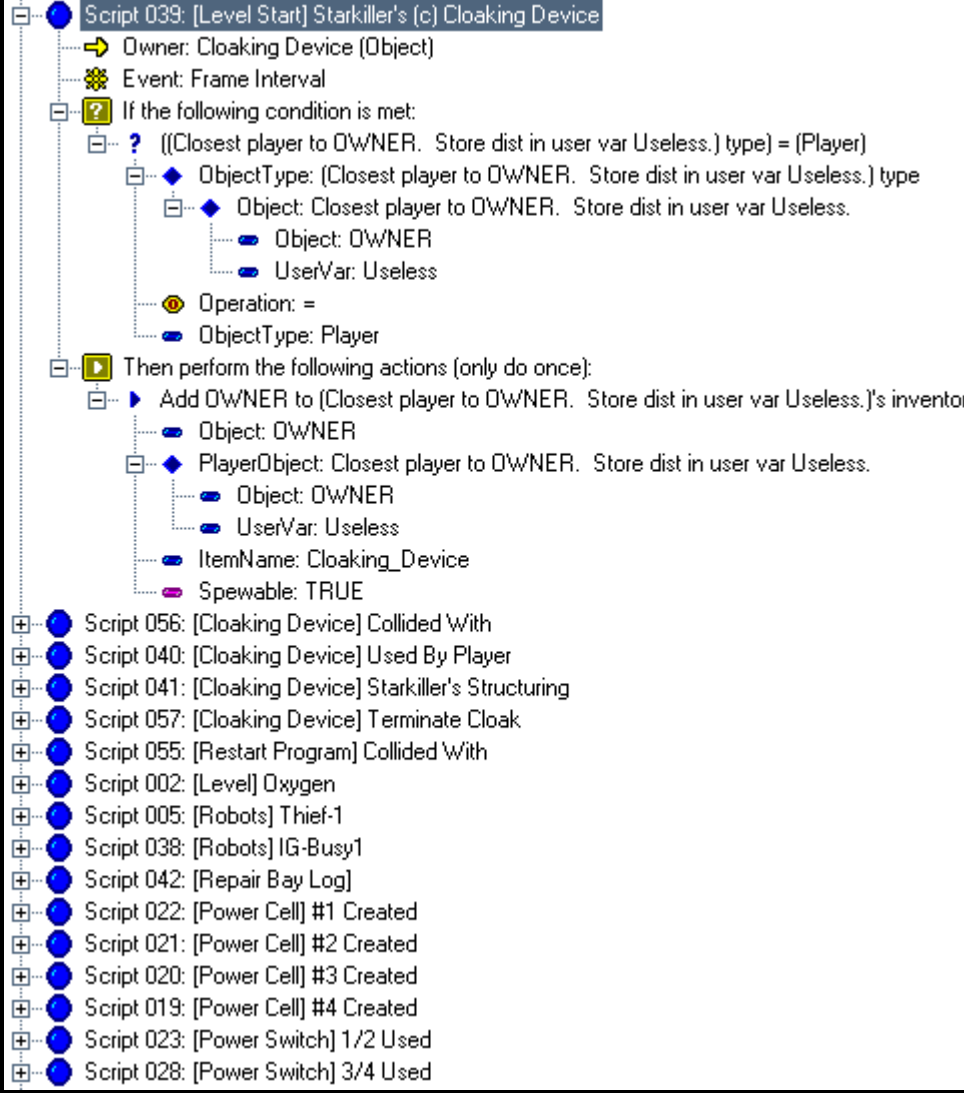
04 - Levelstart-Skript



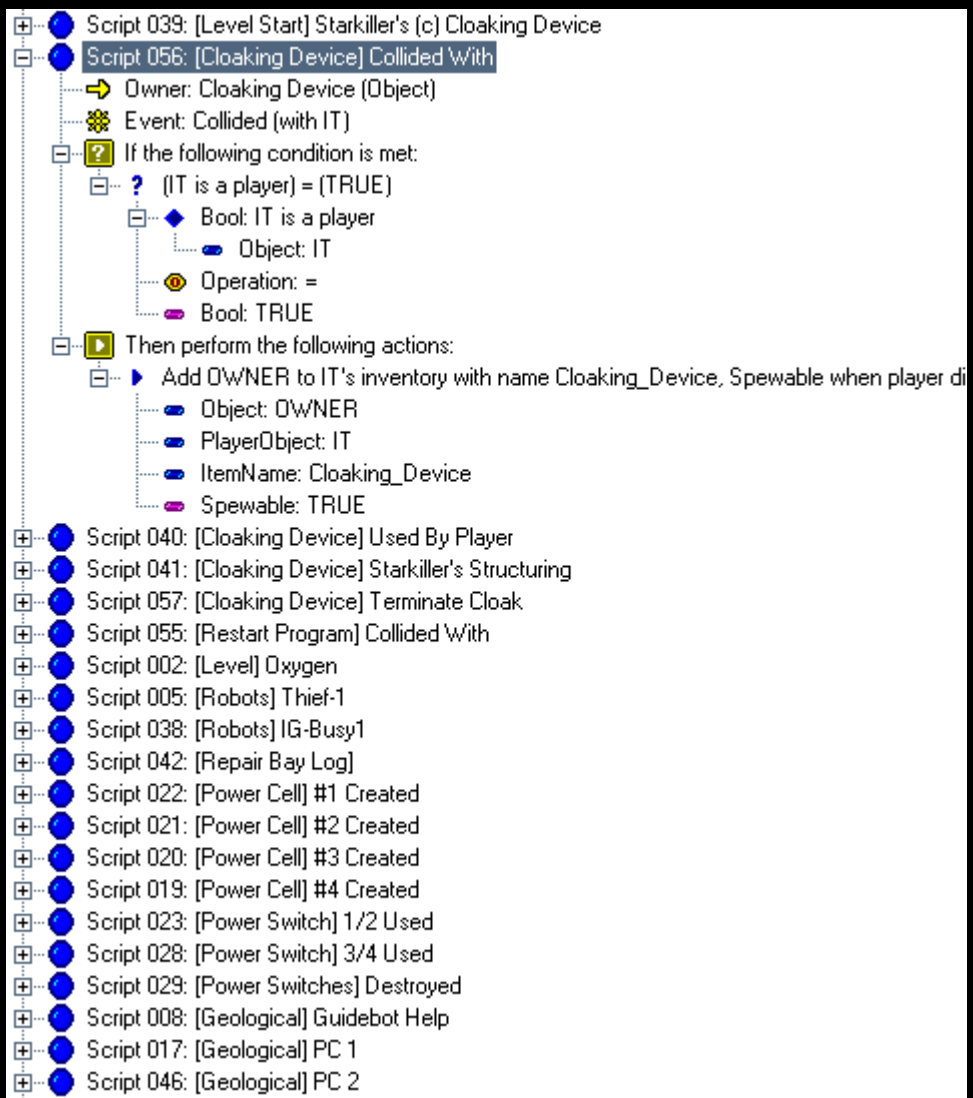
Erzeuge ein Skript das beim Levelstart ausgeführt wird. Darin setze die UserFlag `Cloaked_Flag_1` auf FALSE.

05 - Erstes Skript

Dieses Skript gibt das Objekt automatisch mit dem Namen Cloaking Device jenem Spieler ins Inventar, der ihm am nächsten steht und [-uff- merkwürdiger satz... läßt das Objekt verloren gehen, wenn der Spieler stirbt, während es spewable ist]
Anmerkung: Wenn Du willst, kannst Du diesen Schritt überspringen, sodaß der Spieler das Objekt im Level finden/aufsammeln kann (in diesem Fall lösche die UserVar Useless, da sie nicht gebraucht werden wird).
Verwende dazu dieses Skript (Script039):



06 - Zweites Skript



Dieses Skript fügt das Cloaking Device ins Inventar jedweden Spielerschiffes hinzu, das es berührt, mit dem Attribut beim Spielertod ausgespuckt zu werden. (dupliziere Script056)

07 - Drittes Skript

Verwende dazu Script040. Dieses Skript ist dafür, wenn der Spieler das Cloaking Device aktiviert. Wenn ausgeführt, fügt es das Device wieder dem Inventory hinzu, läßt es verloren sein (?????) gibt dem Spielerschiff die Cloaked_Handle-Identifikation, startet den Energy_Drain-Timer da das Gerät ja nun an ist, und setzt die Cloaked_Flag_#'s die für die Operation des Gerätes notwendig sind.

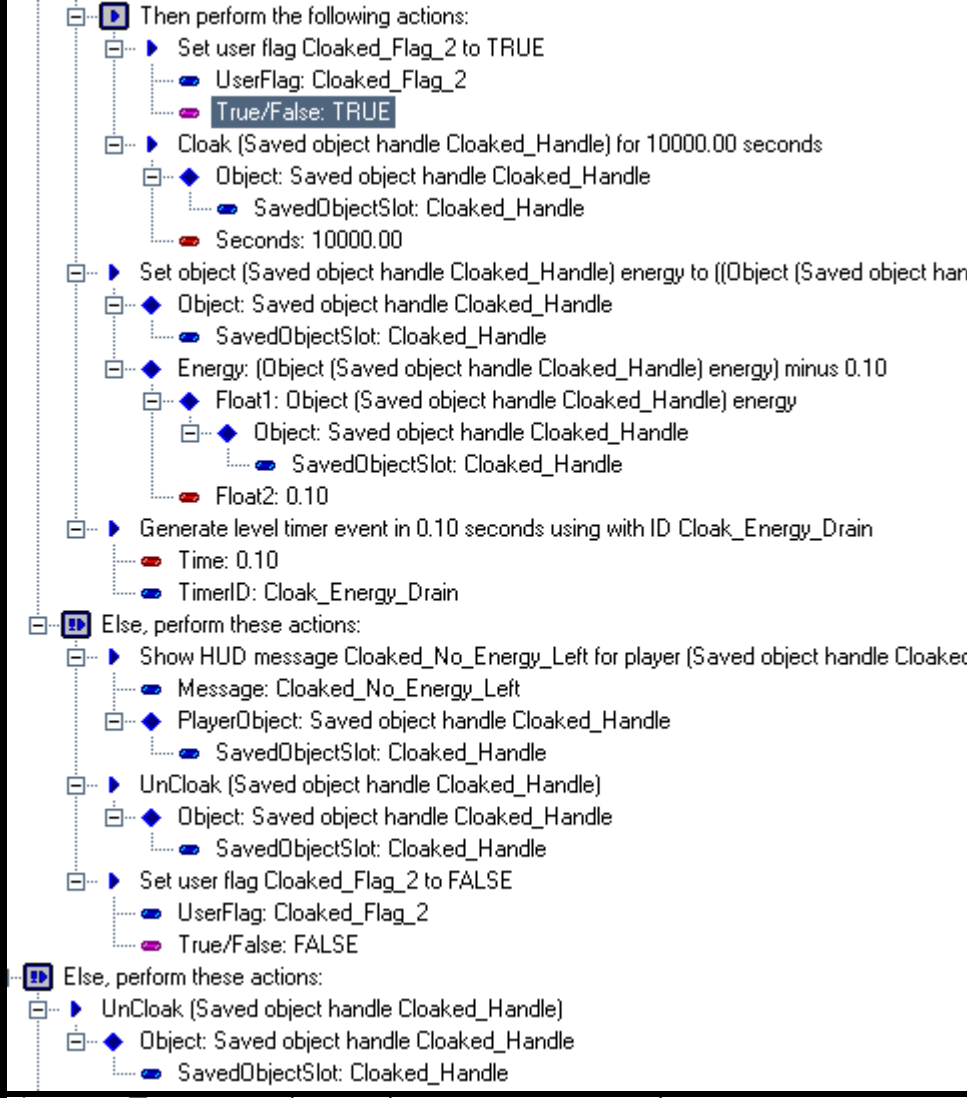
```
Script 040: [Cloaking Device] Used By Player
  Owner: Cloaking Device (Object)
  Event: Used (by IT)
  If the following condition is met:
    ? ALWAYS
  Then perform the following actions:
    Add OWNER to IT's inventory with name Cloaking_Device, Spewable when player di
      Object: OWNER
      PlayerObject: IT
      ItemName: Cloaking_Device
      Spewable: TRUE
    Save object handle for IT in slot Cloaked_Handle
      Object: IT
      SavedObjectSlot: Cloaked_Handle
    Generate level timer event in 0.10 seconds using with ID Cloak_Energy_Drain
      Time: 0.10
      TimerID: Cloak_Energy_Drain
    If the following condition is met:
      ? (User Flag Cloaked_Flag_1) = (TRUE)
        Bool: User Flag Cloaked_Flag_1
          UserFlag: Cloaked_Flag_1
        Operation: =
        Bool: TRUE
    Then perform the following actions:
      Set user flag Cloaked_Flag_1 to FALSE
        UserFlag: Cloaked_Flag_1
        True/False: FALSE
    Else, perform these actions:
      Set user flag Cloaked_Flag_1 to TRUE
        UserFlag: Cloaked_Flag_1
        True/False: TRUE
Script 041: [Cloaking Device] Starkiller's Structuring
```

08 - Viertes Skript

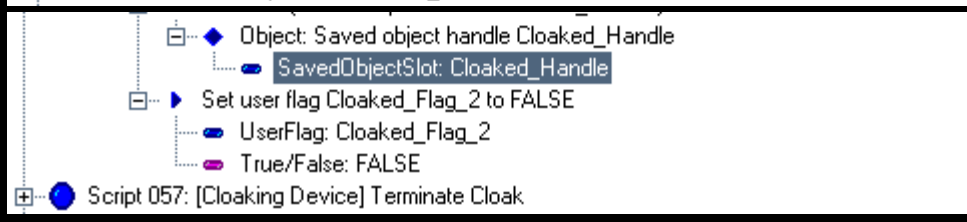
Dieses ist das schwierigste von allen, da es die Tarnvorrichtung arbeiten läßt. Es ist in mehrere Bilder aufgeteilt, da es nicht auf einen einzigen Screenshot paßt. Im ersten Bild ist der Skript-Titel hervorgehoben. In den darauffolgenden Bildern ist jene Zeile hervorgehoben die im Bild zuvor als letzte vorkam, damit Du die Übersicht behältst. Zum Beispiel ist im zweiten Teilbild die Zeile "True/False: TRUE" hervorgehoben, da sie die letzte im ersten Teilbild war. Außerdem ist im zweiten Teilbild das Fenster etwas nach rechts gescrollt, um zu zeigen was eine der längeren Zeilen im Skript macht, jene, die 0.10 Energie vom Spielerschiff abzieht. Obwohl die erste Hauptzeile des Textes abgeschnitten ist (die oberste expandierbare Baumzeile) ist es dennoch möglich zu wissen was sie besagt nachdem man die Skriptzeilen danach analysiert hat.
(Das gesamte Script-Bild ist in der .zip zu finden)

```
Script 041: [Cloaking Device] Starkiller's Structuring
  Owner: Level
  Event: Timer (TIMER ID) Went Off
  If the following condition is met:
    ? (TIMER ID) = (Cloak_Energy_Drain)
      TimerID: TIMER ID
      Operation: =
      TimerID: Cloak_Energy_Drain
  Then perform the following actions:
    If the following condition is met:
      ? (User Flag Cloaked_Flag_1) = (TRUE)
        Bool: User Flag Cloaked_Flag_1
          UserFlag: Cloaked_Flag_1
        Operation: =
        Bool: TRUE
    Then perform the following actions:
      If the following condition is met:
        ? (Object (Saved object handle Cloaked_Handle) energy) >= (1.00)
          Float: Object (Saved object handle Cloaked_Handle) energy
            Object: Saved object handle Cloaked_Handle
              SavedObjectSlot: Cloaked_Handle
          Operation: >=
          Literal: 1.00
      Then perform the following actions:
        If the following condition is met:
          ? (User Flag Cloaked_Flag_2) = (FALSE)
            Bool: User Flag Cloaked_Flag_2
              UserFlag: Cloaked_Flag_2
            Operation: =
            Bool: FALSE
        Then perform the following actions:
          Set user flag Cloaked_Flag_2 to TRUE
            UserFlag: Cloaked_Flag_2
            True/False: TRUE
```

Zweites Teilbild:

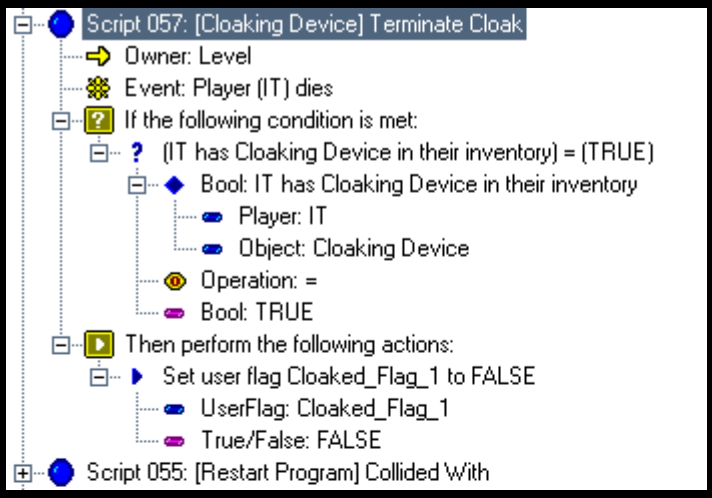


Drittes Teilbild:



09 - Letztes Skript

Während das schwierigste nun hoffentlich erledigt ist, bleibt noch eins über. Repliziere Script057; dieses schaltet die Tarnvorrichtung ab wenn der Spieler stirbt.



10 - Fertig!

Sechs Skripten insgesamt, mitgezählt das Level-Start-Skript, welches die wichtige `Cloaked_Flag_1` auf FALSE setzt. Eine Sache die Du fürs Verfeinern des Skriptes machen kannst, ist den Wert Float: 0.10 im zweiten Teilbild des achten Schrittes zu variieren. Je höher der Wert, desto schneller wird Dein Schiff Energie verlieren. Wenn Du es auf 0 setzt wird keine Energie abgezogen, aber das würde den Timer sinnlos machen. Wenn Du weißt wie, kannst Du den Timer aus diesem Skript ausbauen, sodaß das Tarngerät sich einfach an-/ausschaltet ohne Energieverlust oder unnötigen Script-Code. Obwohl, hey, niemand würde dann jemals das blöde Ding ausschalten...

▶Cloaking Device

Ich habe das Skript so modifiziert, daß auch kein 'Waber'-Effekt mehr auftritt. Versucht es mal derart umzubauen, daß die Tarnvorrichtung nicht anspringt, wenn zuwenig Energie da ist!

102 - Reactor Gamma: Script Companion

Kyouryuu

Der Begleit-Guide zur `.cpp`-Datei, die Reactor Gamma in der `.mn3` beigelegt ist

Extrahiert Euch die Datei(en) aus 'gamma.mn3', daß Ihr diesem Tut folgen könnt. Ich werde sie hier nicht bereitstellen, da ich Kyouryuu's Arbeit respektieren werde; ebensowenig habe ich hier Screenies davon. Holt Euch gegebenenfalls den Level über eine der einschlägigen Level-Sites. Ladet den Level in D3Edit und öffnet die Skriptfile mit Dallas. Vergeßt nicht, zuvor die Tablefile zu setzen!

Intro

Als ich vor einer Weile mein Reactor Gamma Level für Descent 3 geskriptet habe, War ich manchmal bezüglich Tutorials über diese Materie etwas verloren. Obwohl vieles in Dallas logisch abläuft, gibt es ein paar obskure Elemente auf die jeder Leveldesigner eventuell stoßen wird. Aus diesem Grund habe ich die `gamma.cpp`-Datei, welche von Dallas verwendet werden kann, in der `.mn3` belassen. Ich habe dasselbe in einem früheren SP-Level gemacht, Centroid Military Base, und es war von großem Nutzen als ich Dallas ein Jahr später wieder für Reactor Gamma verwenden mußte. Dieser Guide für fortgeschrittene bis sehr fortgeschrittene User von Dallas wird Euch durch das `gamma.cpp`-Skript führen. Stellt Euch `gamma.cpp`, `-.msg`, `-.dll` und `-.o` in Euer Osiris-Subverzeichnis Eurer SDK-Installation.

Intro Cinematic

Script 000: Level Start FF AND REACDOOR TOGGLE!

Vielleicht das größte und variierteste Skript im Set, es wird beim Level-Start ausgeführt. Als erstes schaltet es alle Spewer im Level an und läßt sie grauen Rauch produzieren. Beachte daß InvisiblePowerups, welche für OutdoorSpew1, OutdoorSpew2, OutdoorSpew3, OutdoorSpewTall und viele andere den Gunpoint auf -1 gesetzt haben müssen, um korrekt zu funktionieren. Andere, wie ShaftSpewerX und ReacSpewerX funktionieren mit Gunpoint auf 0 gesetzt. Beachte auch, daß die ReacSpewers das Physics Flag Gravity gesetzt haben. Dies bringt sie dazu, sich bogenförmig nach 'unten' zu bewegen während sie durch die Luft fliegen. Script000 aktiviert auch eine Vielzahl an Kraftfeldern. Beachte, daß nur eins davon tatsächlich ein Kraftfeld ist - jenes zum unteren Korridor, welches das Portal 3 der Central Junction ist. Die anderen 'aktivieren' eigentlich Glas beim Upper Core und den Tower-Portalen. Die Türen in die Generatorräume werden versperrt und viele Variablen werden mit 0 oder FALSE initialisiert. Die Pumpen werden darauf gesetzt von Frames eins bis drei zu animieren, und viele Objekte - zumeist InvisiblePowerups für die Spewer - werden versteckt. Ein Objekt zu verstecken macht es komplett unsicht- und durchfliegbar. Die FireGadgets, welche in der Abschluß-cutszene auftauchen werden ebenso geghostet wie der HiddenLaser (Ein Bot). Und ReactorGamma bekommt seine Schilde auf 10000.00 gesetzt. Der Thiefbot wird auch auf 1000.00 Schild aufgemotzt. Ich schalte die AI für die Spyhunter Spy1 und Spy2 ab, genauso wie für die CorSwatters. Dazu später mehr. Ich setze auch das EntryDoor auf offen für die folgende Cinematic. Regen und Donner werden auch aktiviert. Du siehst auch ein ungewöhnliches Kommando für ENABLE forcefield at portal 0 in room 2 mit DoublesidedFlag=FALSE. Der Grund dafür wird dann auch im Intro cutscene-Skript besprochen.

Script017: Intro Cinematic 1

Auch dieses wird beim Levelstart ausgelöst. Ich setze die Music Region für alle Spieler und sage der CED Asagiri daß sie sich entlang ihrer Pfad-Wegpunkte für die Einführungs-Kinematographie bewegen soll. Ich starte eine einfache Letterbox-formatige Kinematographie mit der CED Asagiri als Fokuspunkt für die Kamera. Man muß die Path-Nodes nicht ausrichten, da der Kamera gesagt wird daß sie immer auf die Asagiri schauen soll. Ich generiere auch einen Level-Timer mit der Introvoice. Nach drei Sekunden wird das nächste Ereignis ausgelöst.

Script 048: intro cinematic voice
Der Auslöser für dieses Skript ist in Intro Cinematic 1. Der Timer Introvoice wird dieses Ereignis drei Sekunden später ausgeführen, nachdem Intro Cinematic 1 ausgeführt wurde. Alles was dieses Ereignis macht, ist den Sound 2d sound Gamma Intro abzuspielen, welcher die Anfangserzählung für Reactor Gamma ist.

Script018: Intro Cinematic 2
Dieses Skript involviert ein bischen Trickserei. Sein Zweck ist es, die zweite Cutszene direkt auf das Ende der ersten folgend auszuführen. Um dies zu erreichen, ist der Owner auf das Ziel der Kamera in Intro Cinematic 1 gesetzt, was die CED Asagiri war. Der Ereignisauslöser ist AI-Movie Ended. Also, wenn der Film der die CED Asagiri betrifft endet, beginnt dieser Event.

Abschnitt J – Designing

Dies sind keine Tutorials im engeren Sinne, eher philosophische Design-Überlegungen zu den verschiedenen Spielmodi, um die jeweilige Spielvariante bestmöglichst zu bedienen.

103	**Level Design Rules**		Dark Wolf	369
104	**Single Player Level Building**		Hydra	372
105	**Single Player**		Robo	374
106	**Anarchy**		Hydra	376
107	**Anarchy**		Robo	378
108	**Team Games**		Hydra	380
109	**When Great Levels Die**		Schplurg	382
110	**Eye Candy**		Robo	389
111	**Waffenbalance**		Robo	392
112	**Warum Missionen bauen?**		Robo	394

Toc

103 - Level Design Rules

Dark Wolf

Ich habe viele Level für Descent, Duke3d gebaut und auch ein paar für Shadow Warrior als auch ein paar Quests für Zelda Classic. Und ich habe ein paar Schlüsselpunkte im Leveldesign erlernt, und ich dachte daß es hilfreich wäre einige dieser Punkte mit angehenden Leveldesignern zu teilen, die sie vielleicht ernstnehmen und verwenden. Das sind jetzt Design(punkte) auf die ich selbst gekommen bin, und die am besten für mich funktioniert haben. Sie sind auf Leveldesign für Descent abgezielt, aber sie können auf jedes Spiel da draußen angewendet werden. Viel Spaß.

Das Hauptthema:

Das ist die fundamentalste Regel im Leveldesign, die allererste Sache bevor Du auch nur anfängst einen Bleistift aufs Papier zu setzen oder zur Maus zu fingern, erdenke Dir ein Thema für Dein Level. Soll es eine Alien-Welt sein? Eine üppige grüne Mine mit vielen Wasserfällen und Gewässern oder ein feuriges Inferno mit viel Lava? Welche Farben und Texturen wirst Du verwenden wollen? Am besten ist es, mit einem zentralen Levelthema anzufangen und alle Deine Gedanken darum herum aufzubauen. Dein Level wird dadurch dann sehr wahrscheinlich viel besser werden.

Framerate:

Obwohl das wegen der weiterhin stärker werdenden 'Super'-Computer am Markt keine große Sache für die meisten Leveldesigner da draußen ist, habe ich entschieden trotzdem was darüber zu schreiben. Einst war das eine wichtige Sache in Descent, daß es am besten ist wenn der Spieler nicht mehr als 40 Cubes gleichzeitig am Schirm sieht. Dasselbe gilt für Descent 3, versuche nicht mehr als einige Räume in direkter Sichtlinie zu haben, oder das Spiel wird sich merkbar verlangsamen, speziell bei Online-Spielen. Versuche Wände oder Hindernisse, oder sogar Türen zu verwenden um die unsichtbare Grenzen zwischen den Cubes oder Räumen auszusperren, welche die Framerate einbrechen lassen. Eine bessere Framerate bedeutet ein besser spielbares Level, bedeutet daß es eher mehr gespielt wird.

Kontinuität:

Das hängt direkt mit dem Hauptthema zusammen und hat beinahe gleichviel Einfluß auf das Level wie das Hauptthema selbst. Diverse Male habe ich Level gesehen die mit einem einzelnen Thema begannen und einen unmittelbar woandershin versetzt haben. Eine seoul-artige Stadt zu haben die mit vulkanischen Minen kontrastiert, wird nicht gerade zu einem zusammenhängendem Level beitragen. Dein Level muß wie ein glaubwürdiger Ort scheinen, und es wird durchaus nicht zerstückelt wenn da ein Vulkan im Zentrum einer Koreanischen Stadt ist. Nun, das wird funktionieren wenn Du auf's Verdrehte und Extreme aus bist, aber für die meisten Leveldesign-Themen wird das nicht hinhauen und den Spieler befremden. Entscheide Dich für ein Set von Texturen und Strukturen die Du verwenden willst und bleibe während des ganzen Levels dabei.

Details und Realismus:

Der Spieler ist notorischerweise darauf aus kleine Details im Level zu erhaschen, obwohl Du in dem was Du in Descent 1 und 2 machen kannst limitiert bist, gibt Dir Descent 3 immense Freiheit mit Details in Deinem Level zu arbeiten. Wenn Dir vorkommt, ein bestimmtes Gebiet ist kahl, füge mehr Details hinzu. Die zwei grundsätzlichen Details sind Schattierung und passende Texturen. Wenn die appliziert sind und der Raum/Level noch immer kahl ist, versuche Objekte und/oder händisch erstellte Strukturen ins Level einzusetzen um den Realismus zu verbessern. In meinem Level Chocobo Word zum Beispiel habe ich Schubkarren, Hacken und Heuhaufen als Details in der Scheuen eingebaut.

Zuckerbrot und Peitsche:
Ein guter Level sollte eine Serie von Herausforderungen und Belohnungen haben. Die Herausforderung kann vor oder nach einer Belohnung sein. Oft habe ich Levels mit Goodies und Baddies gesehen, die ohne Rhytmus und Vernunft verstreut waren. Schau, daß Deine Herausforderungen und Belohnungen miteinander zusammenhängen, und Du wirst höchstwahrscheinlich eine bessere Befriedigung der Spieler erreichen. Zum Beispiel, ein paar Class 1 Heavy Drillers, die ein oder zwei Smart Missiles bewachen. Oder Thresher, die eine der überaus wertvollen Omega Cannons bewachen. Wenn möglich, versuche die Goodies und Objekte die Du in Deinem Level verwendest mit dem Hauptthema in Beziehung zu setzen.

Wähle Dein Publikum und baue für es:
Du mußt entscheiden, ob Du für ein Singleplayer- oder Multiplayerpublikum baust. Dies wird den Fluß wie Du das Level baust diktieren. Ein gutes Singleplayerlevel tendiert dazu groß zu sein und viele Gebiete zu enthalten die zu erkunden sind. Du willst ja, daß Dein Singleplayer-Publikum eine Menge Spaß dabei hat, sich wild wehrende Horden von Bots zu suchen und jagen, oder verschlagene Rätsel zu lösen. Ein guter Multiplayerlevel andererseits muß offener und signifikant kleiner sein, mit einem Design wo ein Spieler gleich schnell zu allen Stellen gelangen kann und den anderen Spieler innerhalb maximal einer oder zwei Minuten finden kann. Manchmal kannst Du beide Publikumsarten mit einem Level bedienen, indem Du dem Singleplayer den Zutritt zu Bereichen mit Schlüsseln oder geskripteten Objekten verwehrst, und Gebieten die mit Multiplayer-Blowouts oder Skripten geöffnet werden können.

Spielerinteraktion:
Das ist ein sehr wichtiger Faktor in vielen Leveln, wo's Descent ein wenig ermangelt, aber Du kannst immernoch Interaktion in Deine Levels einbauen, Du mußt nur innovativ sein. Versuche auch, so viele Dinge wie möglich einzusetzen die der Spieler verändern oder zerstören kann. Sowas wie Lichter die man ausschießen kann, Kisten und Fässer die explodieren können und Schaltpaneele die man bedienen kann. Mache auch soviele Geheimbereiche wie möglich, was den Spieler dazu zwingt seine Umgebung zu betrachten und interaktiver mit seiner Umwelt zu sein.

Ein gut ausbalancierter Level:
Stelle sicher, daß Dein Level bei Waffen und Robots austariert ist. In Singleplayer sollte gerade genug Munition und Schild vorhanden sein, um das auszugleichen was einem die Bots abnehmen. Zu wenig Goodies bedeutet daß der Level nicht beendet werden kann und zuviele daß der Level ein Kinderspiel ist. Du mußt auch über die verschiedenen Schwierigkeitsstufen im Spiel nachdenken, wo vier Fusion-Hulks im Trainee-Mode recht easy scheinen aber in Insane eine tödliche Barrikade darstellen. Vergiß nicht, daß Dein Publikum mit deinem Level nicht vertraut ist und daher nicht weiß was es zu erwarten hat, also packe 'etwas mehr' Goodies hinein als Du selbst brauchen würdest. Eine gute Multiplayerbalance wäre, daß die Goodies, die Du im Level ausgestreut hast den Spieler dazu bringen, deinem 'beabsichtigten' Pfad durch den Level zu folgen.

Gehe an die Grenzen:
Im Kontrast zu Realismus und FPS ist es immer gut manchmal an die Grenzen zu gehen und die Beschränkungen der Engine zu sehen und was sie schafft. Das Spiel ist trotz allem nur eine virtuelle Realität und Du kannst mit verblüffenden Räumen und Ausblicken im Spiel den Spieler ins Staunen versetzen.

Muster:
Spieler, ungeachtet dessen ob sie es wissen oder nicht, lieben Muster. Sie helfen dem Spieler den gesamten Level zu lösen. Wenn da ein Schalter auf einer Seite des Levels ist, wird wahrscheinlich ein korrespondierender Schalter auf der anderen Seite sein. Wenn Du einen Reaktor zerstörst, wird ein anderer in einer ähnlich aussehenden Region zu zerstören sein. Denke auch in Takten wenn Du Gegner in bestimmten Gebieten platzierst - old scratch, pest, pest, old scratch, pest, pest, etc. Sei nicht zu monoton mische ein bischen um dem Spieler etwas Unbehagen zu verschaffen.

Verbindungen:
Das ist doppelt wichtig, wenn Du die 'besten' Geheimnisse innerhalb des Levels einsetzen willst. Wenn Du kannst, gib dem Spieler durch Fenster oder Gitter einen Ausblick auf Gebiete die er möglicherweise kommen kann und gib ihm später Zugang zu diesen Gebieten. Es ist auch befriedigend, ein Powerup hinter einem Gitter zu sehen und letzten Endes den Weg auf die andere Seite zu finden und sich das Goodie zu schnappen. Da ist etwas wundervoll angenehmes daran, etwas früh im Level zu sehen und später Zugang dazu zu erringen.

Rätsel:
Hier sind die meisten Level in der Descent 3-Welt unzureichend. Es gibt Zeit und Ort für ein Guided-Missile-Rätsel, aber das ist NICHTS für ein erforderliches Level-Ziel! Mache Rätsel mit Guideds immer optional und nur dazu, Zugang zu geheimen Gebieten zu erlangen. Es ist OK, ein paar Schaltpaneel-Rätsel im Level zu haben, aber halte sie für den überwiegenden Bereich in den Neben-Zielen. In Descent geht es darum herumzufliegen und Bots zu zerstören, Rätsel sollten nur zweitrangig sein. Wenn Du unbedingt ein Rätsel mit Guideds haben mußt, gib ein paar Guided-Pakete genau vor das Rätsel, um den Spieler in Richtung der Lösung zu dirigieren. Ebenso ist es in Descent 3 das Beste, wenn das ein wiederkehrendes Raketenpaket ist.

'The Fiddler's Roof':
Zwinge den Spieler nicht dauernd, ein lineares Herangehen ans Level zu nehmen. Dein Level sollte nicht nur 'den einen Weg' haben, um gelöst zu werden. Obwohl in Descent 3 schwerer zu machen als in D1 oder D2, ist es das Beste dem Spieler eine Auswahl mehrerer Routen zu geben, um ein bestimmtes Ziel zu erreichen. Laß manchmal die Spieler dorthin kommen, wo sie nicht hinsollen und begrüße sie mit Feinden und Bedrohungen, für die sie noch nicht gerüstet sind.

'Practice What Your Preach':
Teste Dein Level immer im Spiel! Stelle sicher daß Du alles mögliche bei Deinem Level geprüft hast, Du solltest absolut krank und erledigt von Deinem Level sein, wenn Du mit dem Testen fertig bist und bereit das nächste Unternehmen anzugehen. Ungeachtet ob's für Single- oder Multiplayer ist, teste es unter exakt diesen Bedingungen. Folge diesen Design-Richtlinien und nur wenig kann schiefgehen.

104 - Single Player Level Building

Hydra

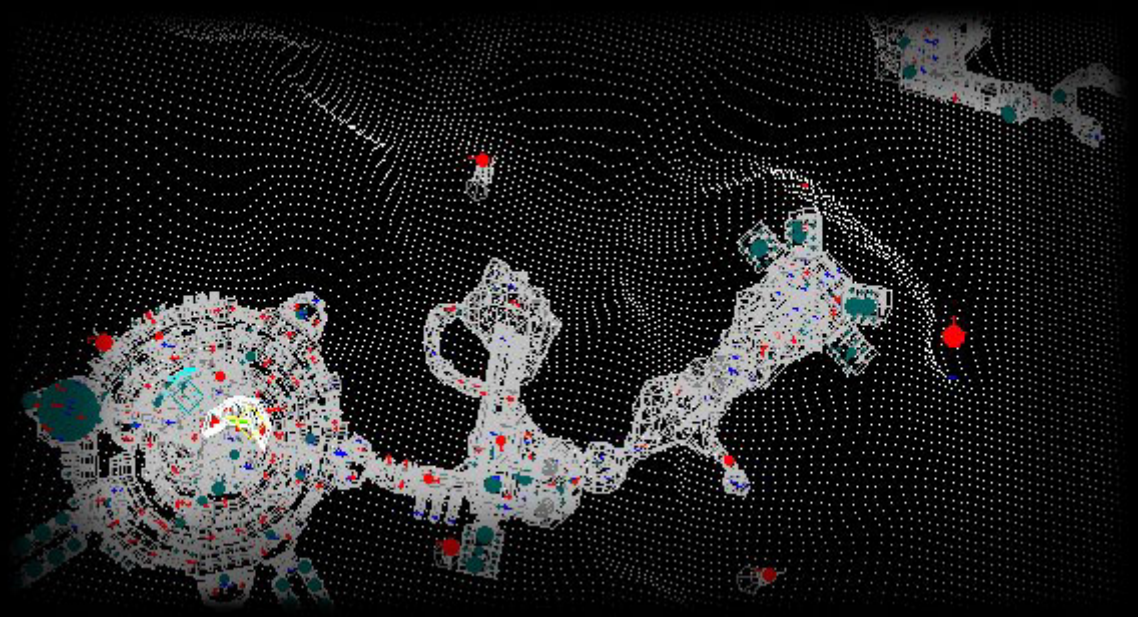


Image taken from level 2 of the level set "Descent 3: Retribution".

Single Player-Levels sind die am schwersten zu bauenden. Du mußt einen Haufen an Gedankenarbeit in die Level-Struktur stecken, viel mehr als bei irgendeinem Multiplayer-Level. SP-Levels sind auch durchschnittlich viel viel größer, da die meisten Multiplayer-Levels klein genug sind, um in etwa 2 Minuten durchflogen zu werden. Auch liegt viel mehr Schwerpunkt auf Skripting. Während Skripting in MP nicht viel Effekt hat, ist es in SP-Levels die 'Farbe' der Level darselbst, was die Levels interessanter macht.

Vor langer, langer Zeit habe ich einen Phrase über das Bauen von Single-Player-Levels für jedes Spiel gehört: 'Mache das Level wie du die Spieler haßt'. Obwohl ich damit nicht vollständig übereinstimme, Denke ich daß es einen guten Punkt aufzeigt. Wenn du ein actiongeladenes Level willst, kannst du das Spiel nicht lasch machen, indem Du es ermöglichst es mit ein oder zwei Leben relativ leicht zu bestehen. Dieses Gleichgewicht ist schwer zu erreichen, es braucht einen Haufen Spieltests um sicherzustellen daß es relativ einfach als Trainee ist, aber nicht zu unmöglich als Insane. Hier kommen jetzt andere Leute (ß-Tester) ins Spiel, da sie ziemlich wahrscheinlich andere Descent-Skills als Du haben. Außerdem, wenn Du einen Level spielst den du gebaut hast, weißt Du wo alle die Geheimnisse und Fallen sind, was das Level etwas einfacher macht.

Architektur ist viel wichtiger in SP-Levels, als in jedem anderen Spielmodus. In Multiplayer bist Du immer damit zu beschäftigt, vor jemandem wegzurennen oder jemandem nachzujagen, als die wundervollen 'Wand-Effekte' zu bemerken, welche im Spiel wirklich nicht viel anderen Nutzen haben als zur Atmosphäre beizutragen. Andererseits, in Singleplayer, könntest Du denselben Raum mehrmals durchlaufen, abhängig vom Level-Layout, und ich (für mich) mag es immer stehenzubleiben und die Szenerie zu bewundern, nachdem ich einen Raum 'gesäubert' habe. Ein letzter wichtiger Tip zur Raum-Kreation für Single-Player ist die Verwendung des Innenraums. Einen großen Raum mit nackten Wänden zu haben kann billig und zusammengewürfelt aussehen, außer er paßt in die Atmosphäre des Levels. Wenn Du einen Raum findest, der zu nackt aussieht, verwende Extrude oder Lathe um ein paar Dinge reinzubauen, wo man sich verstecken kann. Denke nur dran ihn nicht zu überhäufen, die Bots rennen nicht sooft in die Wände wie die Spieler.

Das Level-Layout ist eine andere essentielle Sache beim Bauen. Obwohl es einfach ist, und manchmal notwendig, ein 'Einbahn'-Level zu erzeugen wo Du nur in einer Richtung weiterkommst ist das nicht immer so spaßig oder spannend.Die Einbahn-Levels scheinen mehr ein Spießrutenlauf von Räumen zu sein, die mit Bots gefüllt sind, bis Du das Ende erreichst, wo Du einen Boss/Reaktor/etc. zerstörst. Wenn Levels auf andere Art machst, bringst Du die Spieler dazu durch das ganze Level zu fliegen um einen Schlüssel zu suchen, mit Fallen und Matcens die losgehen und mehr Bots erzeugen. Eine andere große Sache ist Symmetrie. Anders als Andere, denke ich das Symmetrie in Levels gut ist solange sie nicht exakt ist. Wenn Du es symmetrisch baust, solltest Du bestimmte Dinge hinzufügen, die jede Seite auf ihre Weise einzigartig machen, etwa einen zerschmetterten Pfeiler auf einer Seite, während die anderen stabil sind. Wenn Du ein Level machst, versuche ihm nicht diesen 'Copy-Paste-Look' zu geben, das kann langweilig werden.

Roboter- und Waffenplatzierung ist ein Muß für Fallen. Platziere einen schwachen Bot oder eine starke Waffe in der Mitte eines Raumes und ein paar dummherzige Spieler werden reinrauschen und ihn/sie zerstören/aufnehmen. Dann rauschen alle im Schatten versteckten Bots heran und attackieren den Spieler, oder du läßt die Matcens im Raum anspringen. Die Menge an Waffen im Level sollte direkt mit der Menge und Schwierigkeit der Bots korrelieren. Es ist schwer einen Boss zu schlagen, wenn alles was das Level bereitstellt eine Vulcan und ein paar Ammoclips sind. Andererseits ist es zu einfach, mit einer Blackshark und Omegaeine Horde schwacher Bots zu zerstören. Ganz wie der Schwierigkeitsgrad des Levels, ist es sehr schwer das richtige Gleichgewicht zwischen Waffen und Bots zu erreichen. Hier kommen wieder die Betatester ins Spiel, jeder hat andere Skills in Descent.

Nun, ich denke ich habe über das Kreirens eines guten SP-Levels alle wichtigen Dinge abgedeckt. Alles was es braucht ist Kreativität, Zeit und eine eine ganze Menge Geduld. Wenn ich denke daß ich hier etwas vergessen habe, bin ich sicher daß ich noch einen oder zwei Absätze hinzufügen werde.

105 - Single Player

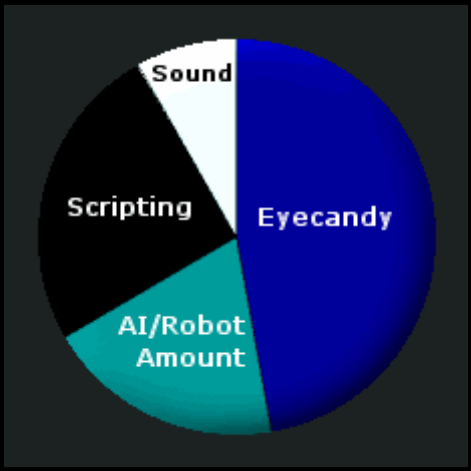
Robo

Singleplayer ist die Basis aller Spiele. Grundsätzlich bist Du auf Dich gestellt. In Descent 3 bedeutet das Robots zerstören, Objectives erfüllen und aus der Mine entkommen. Single-Player-Levels sind die bei weitem am schwersten zu bauenden. Dieses Tut könnte es *leichter* für Dich machen....

BlueFlames hat mir mit seiner Soupe Du Jour über Atmosphäre wirklich die Worte aus dem Mund (oder vom Gamepad könnte man sagen) genommen. Ich zitiere:

In **Singleplayer** muß eine gehörige Menge an Planung hineingesteckt werden wie ein Level aussehen wird, **wenn gegeben ist daß es eine Hintergrundgeschichte zur Existenz des Levels gibt. Minen brauchen tiefe Schächte und felsige Oberflächen, Kommandozentren müssen effizient erscheinen und um die Betreiber des Ortes herum designt sein, und so weiter und so fort. Der Level muß so aussehen, daß er in einer bestimmten Hinsicht funktionell (oder nicht mehr, abhängig von der Hintergrundgeschichte) scheint, oder der Spieler verliert die Atmosphäre.** Diese ist offensichtlich nicht die oberste Priorität in einem Multiplayer-Level. Dateigröße, Spielbalance und gesamte Effizienz sind ganz klar die notwendigsten Komponenten eines Multiplayer-Levels, aber es so aussehen zu lassen als hätte es einen 'Zweck' sollte nicht ganz vergessen werden. Es ist echt etwas ätzend, wenn Du in einem ruhigen Moment in einem Level herumfliegen mußt, das aussieht wie... naja... Korridore mit zufällig applizierten Texturen an jeder Wand.

Und wie er da recht hat. Anders als die meisten Multiplayer-Levels werden die Meinungen, das Gameplay und das Feeling des Levels selbst direkt durch die Atmosphäre und die Struktur des Levels beeinflußt. Sogar jetzt, nach 17 (!) Wiederholungen der in D3 inkludierten Singleplayer-Missionen schaue ich noch immer die Levels an und sage "Wow...". Dies ist jedes Leveldesigners Traum und Ziel, oder zumindest sollte es das sein. Ohne Atmosphäre ist Dein Level, wie soll ich sagen... tot.



Also was genau ist Atmosphäre? Nun, man könnte sagen es ist die Menge an Features und den Eigenschaften Deines Levels. Guck Dir diese Tortengrafik an. Ich würde schätzen, daß 45% der Atmosphäre vom Eyecandy und der Geometrie in Deinem Level abhängt. 20% von der Menge an Robots, 25% vom Skripting und 10% vom Sound.

Ich habe dem Eyecande wohlüberlegt so einen hohen Anteil gegeben, da es der größte entscheidende Faktor **ist**, in jedem Level. Wie ich in meinem Tutorial über Eyecandy ausführe, ohne ist Dein Level aus blanken Wänden und nicht interessant. Du mußt Eyecandy haben! ☺

Anders als in Multiplayer, fliegst Du nicht in totaler Raserei herum; Du wirst Dir wahrscheinlich Zeit nehmen, alle Rätsel zu lösen und die Bots zu rösten. Während Du herumfliegst, wirst Du auch die Zeit haben Dich umzusehen, nachdem jeder wissen muß wo er hinmuß - und jeder braucht 'Landmarken' um die Position festzustellen. Es gibt wahrscheinlich so viel was ich sagen könnte, ich könnte so vier Seiten weitermachen - aber ich machs nicht. Ich hinterlasse Dich einfach mit den wichtigen Punkten:

- Stelle sicher, daß du sehr attraktiven Eycandy baust, [dieses Tut] könnte helfen.
- Versuche die Atmosphäre Deines Levels auszubalancieren, nicht zu viel Sound oder Scripting, aber nicht zuwenig.
- Dein Level, wenn richtig gemacht, sollte alle Typen von D3-Piloten ansprechen - ziele auf das ab.
- Viele Leute nehmen sich Zeit, wenn sie durch Singleplaye-Levels grehen, Du solltest dasselbe machen wenn Du sie baust.
- Stelle sicher daß Du den Level in ein paar guten Archiven veröffentlichtst, etwa wie PlanetDescent, GameEdit.com oder Levels4You.com (mittlerweile nur mehr mit Anmeldung)!
- Wenn diese Infos Dir geholfen haben, in irgendeiner Weise - schicke mir eine Nachricht und ich werde Dich rühmlich avisieren ☺

Scheint ein wenig kurz, ich habe vielleicht ein paar Sachen vergessen, ich werde sie hinzufügen wenn ich mich erinnere ☺ Viel Glück mit den Leveln!

Seems a little short, I've probably forgotten some things, I'll add them when I remember ;) Good luck with those levels!

106 - Anarchy

Hydra

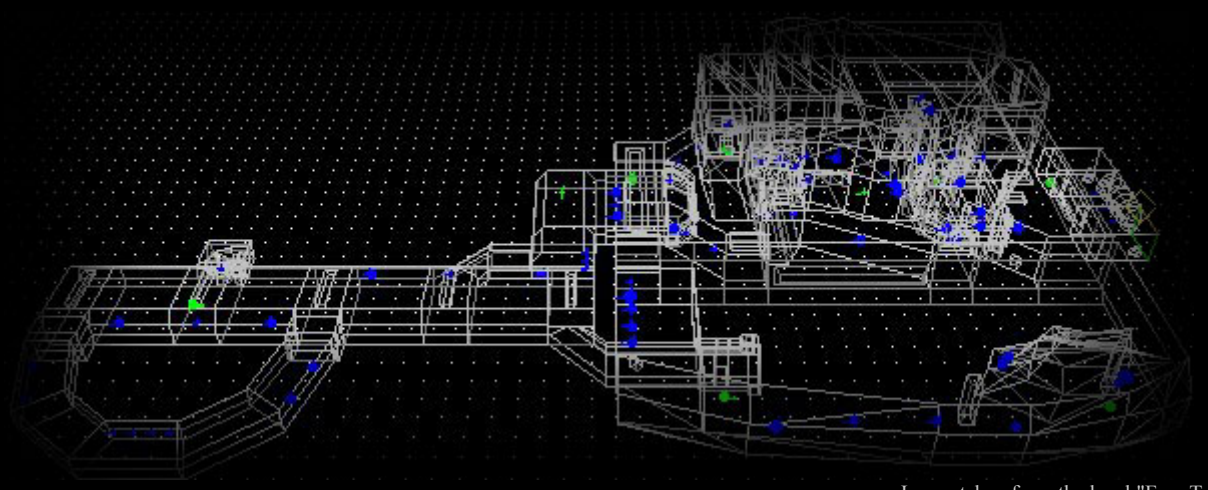


Image taken from the level "Frag Tag".

Anarchy ist möglicherweise der meistgespielte Mode in Descent3. Anarchy kann auch der facettenreichste Leveltyp sein den es gibt. Manche große Anarchylevel können aus einem oder zwei großen Räumen bestehen, während andere gut mit Waben oder Tunnels sind. Jedes hat seine starken und schwachen Punkte fürs Gameplay. Skripting ist keine große Sache in Anarchy-Levels, da die meisten Spieler zu beschäftigt sind entweder davonzurennen oder jemanden zu verfolgen, um nettes Skripting oder aufwendiges Strukturdesign zu bemerken. In Anarchy liegt der Schwerpunkt auf die Spielbarkeit des Levels und nicht so sehr wie komplex das Leveldesign ist.

Ambiente ist eine große Sache in Anarchy Levels. Nicht nur die Sound-Atmosphäre, sondern die zusätzlichen Effekte wie Nebel. Diese geben dem Spieler mehr von dem Gefühl wirklich im Pyro zu sitzen, ohne die Framerate zu vermindern oder zu laggen. Ich empfehle, Ambiente wie benötigt zu verwenden, ich würde nicht jeden Raum mit einem Sound oder Nebelfarben aufmascherln, es würde das Gameplay vermindern. Einen Atmosphäre-Sound für das gesamte Level ist eine nette Note, speziell wenn da ein spezielles Thema oder Motiv in Deinem Level ist. Nebel anzuwenden ist immer eine gute Sache, wenn moderat verwendet. Vergiß nicht, daß Du Deinen Opponent sehen mußt, bevor du auf ihn schießt. Blindlings zu schießen bringt manchmal einen Kill, aber zieht Dir die Energie viel schneller ab.

Wenn Du in einem Feuergefecht bist, ist das Letzte was Du machen willst in Auswüchse der Wand zu rammen. Wenn Du Dein Level machst, halte Ausschau nach Dingen in Deinem Level die Den 'Fluß' des Levels unterbrechen. Alle freistehenden Hemmnisse im Level sollten eine gute Distanz von zwei oder drei Schiffslängen haben (etwa 50-70 Units), um die Navigation zwischen ihnen einfac zu machen. Natürlich, manchmal einen engen Korridor oder Durchgang zu haben der wirklich Deine Skills in einer Jagd fordert, kann echt pulstreibend sein.

Platzierung ist ein anderer Schlüsselpunkt beim Kreiren eines Anarchy-Levels. Hauptsächlich die Platzierung von Energiestationen. Die Menge an Stationen sollte von einer Sache abhängen, die Menge an Spielern die dieses Level durchschnittlich beherbergen wird. Wenn Du Energiestationen platzierst, achte auf die Menge der Energiewaffen, der Menge an Spielern die es enthalten wird und die Größe Deines Levels. Meiner Erfahrung nach, die meisten Spieler werden ein Energiezentrum nicht teilen, wenn sie sich ihrer gewahr sind. Wenn das level für 1vs1 designt ist, sollte ein Energiezentrum voll ausreichen. Ansonsten solltest Du zumindest zwei haben.

Waffen-Balance sollte sehr sorgfältig in deinem Level geplant sein, abhängig von Deiner Levelstruktur. Bei offenen Levels sind Waffen wie EMD, Homers, MD und Omega sehr effektiv und sollten spärlich platziert werden, außer Du willst einen hohen Kill-Level. Wenn Du enge Nahbereichs-Level kreirst, dann richten Waffen wie Fusion, Frags, Vauss und Napalm verheerenden Schaden an, wenn korrekt eingesetzt. Die Menge an Waffen kann ein Level ausmachen oder es vernichten. Zu wenige wird dazu führen, daß ein oder zwei Spieler die einen Haufen Waffen aufgesammelt haben, einfach alle anderen, waffenlosen, Spieler abschlachten. Während zu viele schwere Waffen einfach alle die ganze Zeit killen.

107 - Anarchy

Robo

Anarchy ist der Spielmodus, wo Du auf Dich gestellt bist. Du sammelst Waffen auf, killst andere Leute und vermeidest es selbst gekillt zu werden. Das führt zu vielen Taktiken wie Doubling Back, Out-turning und zahlreiche Waffentaktiken. Aber der Trick Leute in Deinem Level Anarchy zu spielen zu bekommen, ist den Level zuerst auszuplanen.

Taktik ist alles in einem Anarchy-Game. Ohne Taktik bleibst Du im Newbie-Mode, und Dir wird 'Newbie' für viele Spiele die noch kommen angeklebt, wenn Du keine Kills schaffst :)

Wenn Du nicht lethal, schnell und effizient agierst, ist es das Schicksal daß der arme Kerl da unten auf Dich entfällt. Bezweifeln auf eigene Gefahr!

Aber Du kannst Deine Taktiken nicht voll zur Anwendung bringen, in einem Level das Dir unterkommt - manchmal auf mehrere Arten. Jemals die Erfahrung gehabt, daß Du Dich in einem Level recht gut gehalten hast und nachdem Du in einen Level gekommen bist den Du das erstemal gespielt hast, mit verwirrenden Gedanken gestrandet bist?

Zum Beispiel:

- Wo bin ich?
- Wo ist dieses Energiezentrum?
- Was, WIEDER diese Waffe?
- Blödes Rohr, geh mir aus dem Weg! Argh!
- Doh! Wo fliege ich jetzt hin?
- Was zur Hölle war das!?!?

Gedanke #1: Wo bin ich?

Das Problem - verwirrende Tunnels. Ich habe Level gespielt, wo es beinahe unmöglich zu sagen ist wo Du bist, und vertraue mir, dieser Gedanke hallt mir durch den Kopf. Diese Level sind wie Labyrinthe. Es gibt viele Lösungen für dieses Problem, und Du kannst versuchen Deine Räume zu verändern, oder bunte Lichter hinzufügen. Was die Leute in einem komplexen Level brauchen, ist ein Referenzpunkt um zu sagen wo sie sind.

Gedanke #2: Wo ist dieses Energiezentrum?

Wie ich in meinem Energy-Center-Tutorial sagte, sind diese ein essentieller Teil eines Levels. Und das bedeutet, daß die Leute es ohne viel Schwierigkeiten finden können müssen (nicht immer der Fall, hehe). Es sollte leicht zu lokalisieren sein und taktisch gut platziert sein.

Gedanke #3: Was, WIEDER diese Waffe?

Waffenbalance ist auch ein essentieller Teil des Levels. Das Letzte was Du willst ist daß Dein Level für Fusion-Schrubber oder easy Kills mit Megas bekannt wird. Anarchy kann einen Mix an Waffen haben, gut oder schlecht, es hängt nur von der Art des Levels ab die Du baust. Wenn Du enge Tunnels hast, wirst Du nicht (viele) Homings oder Smarties haben wollen, aber vielleicht Plasma oder Superlaser (und nur ein paar Frags um ein wenig Chaos zu erzeugen).

Gedanke #4: Blödes Rohr, geh mir aus dem Weg! Argh!

Eyecandy ist gut, überexzessive Mengen davon können allerdings für Deine Spieler tödlich sein. Ich kann garnicht sagen, wie oft ich herumgeschossen bin um eine Napalm in irgendjemandes Windshield zu werfen und eine nahegelegene Röhre oder Balken getroffen habe. Dammit - ich verbrenne! 😬

Sicher, füge ein paar Balken oder Lichtaufhängungen hinzu, aber blockiere nicht die Sichtlinie oder den Flugpfad der Spieler.

Gedanke #5: Doh! Wo fliege ich jetzt hin?

Sackgassen: One-Way-Tickets auf den Pfad des Verderbens, mein Freund. Großer Fehler! Habe zumindest zwei Ausgänge, oder Du wirst etwas fiesen Druck bekommen!

Gedanke #6: Was zur Hölle war das!?!?

Ängstige Deinen Spielern nicht das Hirn raus, das gibt eine fürchterliche Unordnung! Fallen, blinkende Lichter, Explosionen (Kool! Was istn das?... Boom!) und Nebel kann ängstigen, irritieren, verärgern oder sogar das Gameplay aufwerten... man kann nie wissen...

Ich hoffe das war hilfreich für Dich, jetzt baue diese Anarchy Level!

108 - Team Games

Hydra

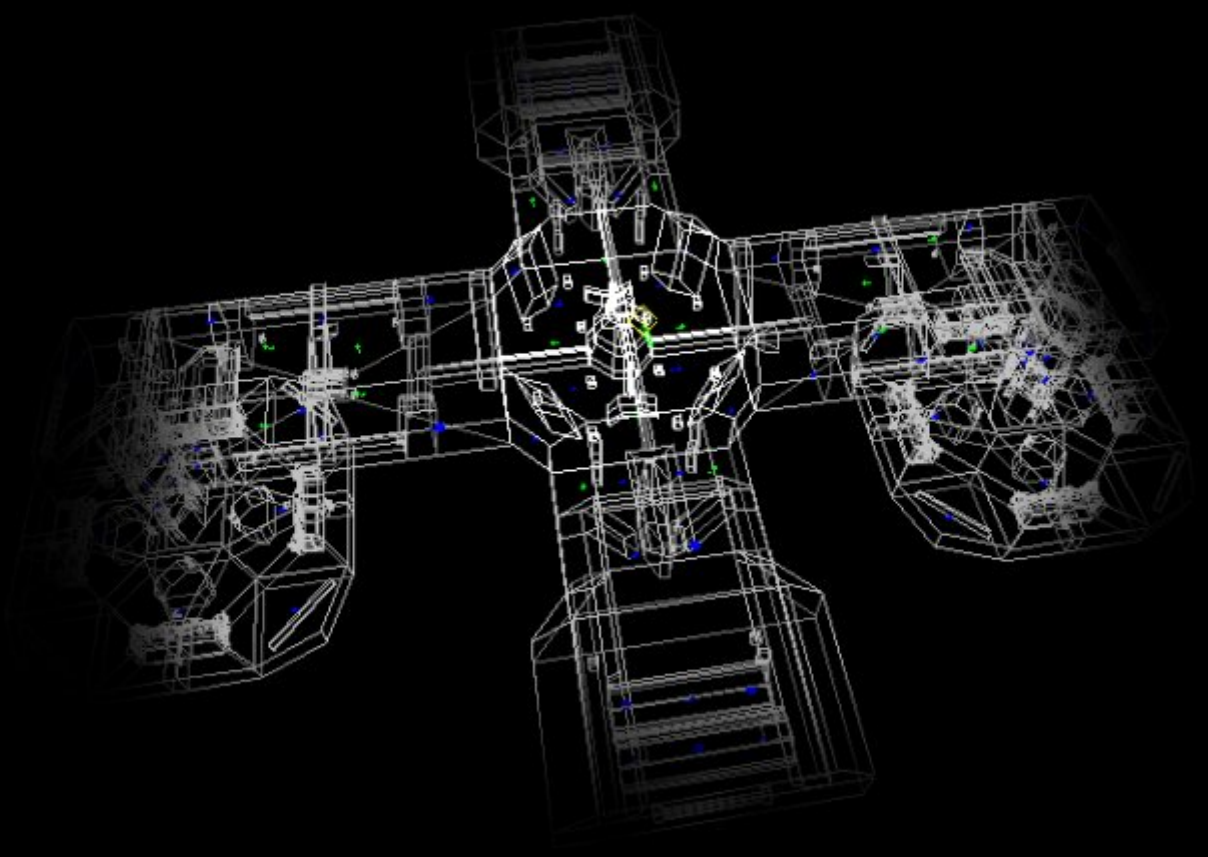


Image taken from the level "No Mans' Land".

Symmetrie ist der Schlüssel für Team-Games. Warte, ich formuliere's besser, BALANCE ist der Schlüssel für Teamspiele. Symmetrie zu machen ist nicht immer eine gute Sache, weil wenn Du Deine Basis in-und-auswendig kennst, kennst Du die gegnerische auch. Die Team-Gebiete balanciert zu halten ist die wichtigste Sache, die Du im Hinterkopf behalten mußt wenn Du ein Teamspiel-Level machst. Ein Team könnte Strukturen in seiner Basis haben um nette Offensiv-Attacken auszuführen, während andererseits die andere Team-Basis gut sein könnte, ihre Position zu verteidigen. Wie Du die Basen aufstellst, bestimmt den Level.

Zugang zu den Basen ist auch eine wichtige Sache. Eher öfters wirst Du nicht nur einen Eingang zu einer Teambasis machen. Das würde es Campern erlauben bei der Öffnung zu sitzen und jeden niederzunageln, der zur Basis rein- oder rauskommt. Üblicherweise sollten zwei oder drei Einfluglöcher sein, womit Du kreativ sein kannst (das Haupttor und eine geheime Passage, etc...)

Es gibt aus meiner Perspektive zwei verschiedene 'Stile' von Teamspielen. Einer, wo Du eine 'Basis' hast die nur mit ein paar Eingängen befestigt ist. Der andere ist mehr ein Labyrinth von Tunnels und diese Tunnels verbinden zu einem größeren Raum mit den Flag/Entropy-Goals. Jeder dieser Stile hat seine eigenen Stärken und Schwächen. Der 'Basis'-Stil ist nett weil generell leichter zu verteidigen, aber normalerweise ist das Gebiet zwischen den zwei Basen eine Art großes 'Niemandsland' (siehst Du wo ich den Namen meines Levels herhab'? :) was bedeutet daß die meisten Gefechte hier geschlagen werden. Es ist schwer zu einer Basis zu schleichen, außer es gibt einen Tunnel der beinahe direkt hineinführt. Der 'Labyrinth'-Stil hat seine eigenen Stärken, zum einen ist es leichter zur gegnerischen Basis zu schleichen, da dort viel mehr Öffnungen für sie zu verteidigen sind und Chancen bestehen, daß sie nicht alle verteidigen. Obwohl eine der größten Schwächen ist, daß wenn man keinen Sinn für die Richtung hast, wie ich, ist es einfach sich in den Tunnels zu verlieren und in Kreisen zu fliegen.

Waffenplatzierung sollte ausgewogen für beide Teams sein. Es sollte die gleiche Menge an Waffenspawnpunkten pro Team sein und viele mehr im Niemandsland, oder Labyrinth oder woauchimmer. Abhängig von der Levelgröße, sollten von jeder Waffe zwei vorhanden sein, sodaß wenn ein Team mit einer Waffe anfängt, irgendwo in der anderen Teambase, die anderen mit der selben Waffe anfangen. Natürlich, wenn das Level mittlerer Größe ist wären zwei Blacksharks nicht die beste Idee, also verwende Deinen Hausverstand auch beim Platzieren von Waffen.

109 - When Great Levels Die

Schplurg

Part I

Warum spielt niemand meinen Level?

Es ist den meisten von uns passiert - du wendest Monate mit D3Edit auf, um das zu bauen was Du für das "Ultimative Level" hältst. Custom Texturen, Custom Schiffe, ein paar ungeheuerliche neue Waffen, großartige Architektur so kool, daß Du glaubst ein realer Platz sollte danach modelliert werden! Nachdem Du Tage damit verbracht hast jedes Detail zu prüfen und nochmals zu prüfen, packst Du es zusammen und schickst es für die Welt an die Level-Seiten, um sich daran zu erfreuen.

die Spannung steigt, nachdem du so lang wie möglich gewartet hast bevor Du online nachcheckst wie viele Server Dein neulich veröffentlichtes Meisterstück laufen haben. Du siehst daß Dein Level nicht gespielt wird... "Oh, gut, es braucht vielleicht noch einen Tag oder zwei...". Falsch. Nach Tagen der Suche, ist Dein Level nirgends zu finden. Vielleicht sahst du einen Server der es einen Moment laufen hatte, zumindest denkst Du das. Aber andererseits, Dein Level scheint von den Massen ignoriert zu werden. Dein Level wurde in das Schwarze Loch der Level gesaugt, um niemals von jemandem gesehen oder gespielt zu werden. All die ganze harte Arbeit für nix... nun, DAS ist eine traurige Geschichte, mein Freund.

Also warum bleibt dieses großartige Level ungespielt, während andere Levels aussehen als wären sie an einem Abend zusammengeschustert worden auf dutzenden verschiedenen Servern gelistet sind? Was fehlt dem neuen Level exakt? Was ist mit dem anderen, schäbig aussehenden Level was es so populär auf PXO, HEAT oder sonstwo macht? Warum spielt niemand mein Level ?!?!?!

Ich weiß es nicht genau, leider. Aber ich habe viele Ideen und Vorschläge, die Deinem Level eine echte Chance auf Unsterblichkeit in der Gamingcommunity verschaffen.

Layout und Balance

Das meiste der behandelten Layout-Infos ist speziell für Anarchy-Levels relevant. Das Design-Layout ist der wichtigste Faktor beim Levelbau. Wenn Deine Räume und Tunnels zu klein oder zu groß sind, oder wenn das Layout der Räume nicht so gemacht ist daß es einen gleichförmigen 'Fluß' durch den Level begünstigt, kann keine Menge an koolen Waffen, Beleuchtung oder fescher Architektur den Level retten. Gleichzeitig wird ein perfekt angelegter Level durch schlechte Beleuchtung, schlechter Waffenbalance oder häßlicher Texturierung leiden. Aber Waffen hinzuzufügen oder wegzunehmen oder ein paar Lichter oder Texturen zu ändern ist viel leichter als alle Räume in einem Level zu vergrößern. Du MUSST ein gut durchtachtes Layout für das Level haben um eine Chance zu haben daß es spielbar ist, Spaß macht und daher erfolgreich wird.

Also was macht ein gutes Level-Layout aus? Ich habe befürchtet, daß Du das fragen wirst, es ist eine gute Frage. Ein Weg die Antwort zu finden ist, einige der populärsten Level zu untersuchen und herauszufinden, was sie laufen läßt.

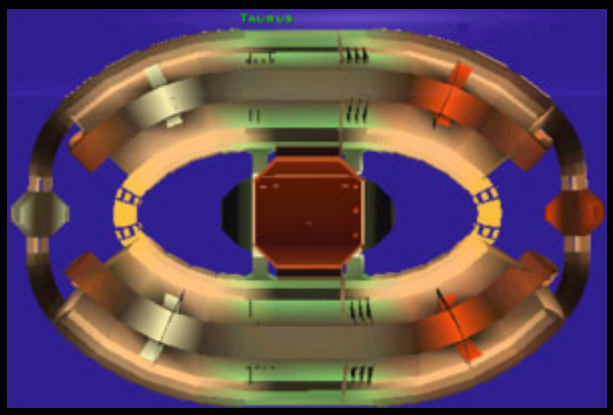
Für Descent 3, einige der Levels die den Test der Zeit online zu sein bestanden haben sind **Burning Indika 3**, **Pyroglyphic**, die Levels aus dem **Fury**-Set (von Outrage), **Skybox** und sogar mein eigenes **Earthshaker 2001** hat zeitweise ganze sechs Server zur selben Zeit (und so wenig wie einen oder zwei ☺) Was ist mit diesen Levels was sie so populär macht?

The Donut Theory

Examining the first 3 mentioned above...Burning Indika 3, Pyroglyphic, and the levels in the Fury set reveals an interesting discovery (especially the Fury levels). It is my belief that, what I call the "Donut" design, is responsible in large part for the playability and fun factor in some of these levels. Not to say that the levels look like giant donuts, but think about this...in the Fury set, Levels 1, 3, and 5 all have a central battle room, and an outlying series of passages that circle it. Pyroglyphic is somewhat similar (though more elaborate and less symetrical) in that there are a series of passages that more or less surround a central area or two. Burning Indika is a little less "Donut-ish" also, though it too may fit into this theory somewhat. I mean no offense when I refer to a level as a Donut, as none of these levels are that simple! Any level that loosely fits the following description will fit into my Donut Theory heh.

Die Donut-Theorie

Die Betrachtung der ersten drei oben erwähnten Level, **Burning Indika 3**, **Pyroglyphic** und die Level im **Fury**-Set läßt eine interessante Entdeckung erkennen (speziell die Fury-Levels) Ich glaube, daß das was ich die 'Donut-Theorie' nenne für den größten Teil der Spielbarkeit und dem Spaßfaktor einiger dieser Levels verantwortlich ist. Die Levels sehen nicht gerade wie große Donuts aus, aber denke darüber nach... im Fury-Set haben die Levels 1, 3 und 5 alle einen zentralen Raum für Gefechte, und eine Serie von außen liegenden Passagen die ihn umrunden. Pyroglyphic ist da ähnlich (allerdings ausgeklügelter und weniger symmetrisch) darin, daß da eine Serie von Passagen ist, die mehr oder weniger ein oder zwei Zentralgebiete umgeben. Burning Indika ist auch etwas weniger donut-mäßig, obwohl es auch irgendwie in diese Theorie reinpaßt. Ich will niemanden beleidigen wenn ich von einem Level als Donut spreche, da keines dieser Level derart simpel ist! Hey, jedes Level das irgendwie in die folgende Beschreibung paßt, paßt in meine Donut-Theorie.



The Donut Theory™ - Ein Level mit einem zentralen Gefechtsbereich das mehr oder weniger von einer Serie kleinerer Passagen mit einer gewissen Komplexität umgeben ist. Taurus, das Level 5 aus Outrage's Fury-Serie, ist ein erstklassiges Beispiel. Das rötliche Zentralgebiet hat einen einfachen Ring von Tunnels, die es einkreisen.

Nun, ich bin kein Psychologe also kann ich nicht genau sagen warum diese Theorie funktioniert (wenn sie wirklich funktioniert). Ich kann aber raten ☺

- Es ist einfach, über das zentrale Gebiet von einer Seite des Levels zur anderen zu gelangen.
- Spieler können sich in mehr als einer Richtung bewegen und trotzdem denselben Zielort erreichen (über den äußeren 'Kreis'). Navigieren erfordert weniger nachdenken, faktisch kann man sich überall im Level in dieselbe Richtung drehen... mehr Brainpower kann zum eigentlichen Kämpfen verwendet werden.
- Das Level ist einfach zu behirnen - Du bist nie weit vom Hauptgebiet weg.

- Das Level ist actionreicher, da es kompakter ist. Vergleiche wie lange es braucht, im Fury-Level 5 (Taurus) quer durchs Level zu kommen mit der Zeit die es braucht Level 2 (Halfpipe) zu durchqueren, welches ein langes, gerades Level ist. Zum Geschehen zu kommen geht in einem Level wie Taurus schneller, welches wirklich wie ein Donut geformt ist ☺
- Spricht beide an, Dogfighter und Tunnelratten.

Die Donut-Theorie ist eine persönliche Theorie die ich über Leveldesign habe. Natürlich gibt es da draußen sehr wohl andere populäre Level, die nicht in dieses Design passen. Aber der Donut scheint mir so offensichtlich, daß er seinen eigenen Abschnitt hier verdient hat.

Hinter dem Donut

Also ok, Donuts können eine gute Idee für Anarchy-Levels sein oder auch nicht. Allerdings wollen wir nicht, daß uns alle unsere Levels an den lokalen Donut-Shop erinnern! Was also können wir noch machen, um unsere Level spielbar und interssant genug zu machen, um die Leute dazu zu bringen für 'Mehr' wiederzukommen?

Bleiben wir bei der Sache mit dem Layout. Die meisten von uns sind mit Pyroglyphic von Skyrat vertraut. Ein großartiges Level. Hübsche Texturen und perfekte Beleuchtung geben dem Level eine großartige 'Stimmung'. Aber da sind andere Level, die genausogut oder vielleicht sogar besser aussehen (ich werde diese Level hier nicht erwähnen oder irgendetwas Negatives über IRGENDEIN Level sagen) aber die werden nicht gespielt. Es ist nicht das Aussehen allein, was ein Level groß macht. Burning Indika 3 verwendet eine Mischung aus Nicht-Custom-Texturen, D2-artiger Architektur, nicht viel Eyecandy im Weg, und doch ist es wahrscheinlich das meistgespielte Multiplayer-Level auf PXO! Also was haben Pyroglyphic und Burning Indika gemeinsam, was sie so verdammt populär macht?

Die Level sind kompakt und ein Spieler kann schnell von einem Bereich zum Anderen gelangen... Die Action ist deswegen schnell. Da ist mehr als ein Weg rein und raus für jeden Raum, und gewöhnlich mehr. Da sind Orte, wo ein Spieler hinter einer Ecke während eines Gefechtes Deckung nehmen kann und schnell hinter dem Opponenten auftauchen kann. Wenn Du einen Spieler um eine Ecke kommen siehst, könnte er/sie sich zurückziehen oder eine alternative Route nehmen um Dich zu umkreisen und hinter dir aufzutauchen. Du weißt einfach nie, wo du hinschauen sollst.

Die Tunnels sind groß genug um darin zu kämpfen, aber klein genug daß Frag-Missiles und Concussion-Waffen effektiv sind und ihnen schwer auszuweichen ist. Meist ist es einfacher einen Spieler mit Laser in einem Tunnel zu treffen, da die Spieler hier weniger Platz zum Dodgen haben. Manchmal streue ich meine Quadlaser einfach einen Gang entlang, wissend daß zumindest einige der Schüsse ihr Ziel treffen werden. Dogfighting in einem großen Raum oder Außengebieten kann bei einer langsamen Connect beinahe unmöglich sein. Ein Haufen Spieler mit Analogmodems hängen aus diesem Grund an den Tunnels. Sehr enge Bereiche können dem Spieler ein klaustrophobisches Gefühl geben, plus daß manchmal die Wände so nahe sind daß man den Ausweg nicht so einfach sieht.

Die offenen Bereiche dieser Level sind groß genug zum Dogfighten, aber nicht so groß daß es für die Spieler schwierig ist einander zu treffen. Beides, Dogfightingbereiche und Tunnels im Level zu haben, wird Spieler beider Typen anziehen. Den Polygoncount nieder zu halten während man eine eindrucksvolle Architektur hat, ist sehr wohl möglich. Es ist viel netter, ein gut gebautes Level zu spielen daß auch das Auge anspricht.

Ästethik

Die Art wie ein Level aussieht, kann seine Spielbarkeit beeinflussen. Die Verwendung von Textur/ Beleuchtungskombinationen die es schwierig machen andere Schiffe zu sehen, oder einfach hart für das Auge sind, werden den Level ernsthaft beeinträchtigen. Das Verwenden von vielen Blinklichtern, animierten Texturen oder derart hellem Licht daß sogar Ray Charles zwinkern müßte sollte vermieden werden. Das heißt nicht, daß Du solche Dinge nicht in deinem Level haben solltest, übertreib' es nur nicht... Piloten können durch zuviel Eyecandy schneller ermüden.

Die richtige Textur/Beleuchtungskombination wird schonender für die Augen sein, Spielerschiffe werden einfacher zu sehen sein (wenn gewollt), und die Leute werden Dich mehr mögen ☺ Eine Art die Dinge unter Kontrolle zu halten, ist es den Level so zu texturieren, als wäre es ein echter Ort. Du wirst nicht an einen industriellen Ort gehen, und dort grüne Blinklichter, eine gelbe Decke, blaue Streifen am Boden und anderes schrilles Zeug sehen. Es wird eher weniger spektakulär, mit einfachen, metallischen Texturen und in einer recht faden Farbgebung sein. Aber vielleicht denkst Du an eine andere industrielle Einrichtung als ich ☺ Sei natürlich kreativ... es IST trotz allem ein Spiel.

Part II

Veränderung ist gut... manchmal

Ok, Du hast also nun ein gut gelayoutetes Level, es ist beleuchtet und texturierst wie Du es willst, niemand hat sich über Anfälle durch's Ansehen von irgendwelchen verrückten Texturen über ausgedehnte Zeiträume beschwert, was noch?

Eine koole Sache beim Game-Editing ist daß Du die Freiheit hast, beinahe alles was Du willst mit dem Spiel zu machen. Du kannst stärkere, oder sogar total customisierte Waffen machen. Du kannst Deine eigenen Bots und Spielerschiffe machen! Du kannst Deine eigenen Custom-Ereignisse skripten, etwa einen Schalter, der einen Raum mit Feuer anfüllt, um Opponenten zu frittieren. Dein erster Impuls mag sein diese 'gott'-ähnliche Macht dazu zu verwenden, die größte Bombe, das schnellste Schiff und die meisten Skripten zu entwickeln die Du kannst. Bevor Du allerdings irgendwelche Änderungen machst, gibt es ein paar Dinge die Du berücksichtigen solltest.

Versuche nicht anders zu sein nur um anders zu sein. Die Leute mögen Veränderung nicht zwingend. Füge schwerwiegende Änderungen nur dann hinzu, wenn Du denkst daß das wirklich zur Spielerfahrung beiträgt. Stelle sicher, daß Du Deine Ideen für Custom-Schiffe und -Waffen sehr sorgfältig durchdenkst. Du willst kein Schiff kreiren, welches derart schnell ist, daß die anderen zwei oder drei dagegen nicht die geringste Chance haben. Und, wenn es zu verschieden von dem ist was die Leute zu fliegen gewohnt sind, tun sie es sich vielleicht nicht an es zu erlernen. Speziell da das Schiff nur in EINEM LEVEL verfügbar ist, nämlich Deinem! Ein Custom-Schiff zu inkludieren kann oder kann nicht dem Erfolg eines Levels helfen. Die Leute mögen es meistens, daß ihr Lieblingsschiff überwiegend unverändert bleibt... "Hey, warum ist meine Pheenie so langsam?!"

Dasselbe gilt für Waffen. Ich habe einige recht verrückte Waffen gesehen, mit denen Leute aufgekommen sind. Einige davon sind vielversprechend, andere entweder gröblich overpowered oder einfach albern. Ich erinnere mich, daß ich einen Impact Mortar getestet habe, der solange abgeprallt ist bis er schließendlich jemanden getroffen hat.

Die Earthshaker die wir entwickelt haben scheint ihre Popularität zu halten, oder zumindest das Level wo sie drin ist. Diese Waffe war aus bestimmten Gründen erfolgreich. Der Hauptgrund ist daß viel der Spieler die Earthshaker noch aus Descent 2 kennen. Die Shaker-Level waren online sehr populär, speziell bei hochpingigen Verbindungen, da Genauigkeit nicht so ein großer Faktor bei diesen Raketen war... sie zu dodgen, darum ging es. Viele Leute vermißten die Shakers... jeder der sie gut für D3 re-designt hat, würde auf einen möglichen Erfolg ausblicken können.

Aber wenn sie schlecht gemacht worden wäre, würde sie elendig versagt haben. Eine Website wie GameEdit zu haben um sie zu fördern hat der Shaker auch nicht weh getan.

Durch viel testen und troubleshooten konnten wir mit einem passablen Design aufwarten. Es gibt ein paar kleine 'Probleme' mit der Rakete, aber insgesamt ist sie gut für alle drei Schiffe ausbalanciert. Jede Schiffskapazität für diese Rakete wurde verändert, um die Geschwindigkeit und Stärke für jedes Schiff auszugleichen. Das langsamste Schiff hat die höchste Shaker-Kapazität, während das schnellste nur zwei Shaker tragen kann. Spieler können in jedem Schiff erfolgreich sein.

Der Level selbst ist nicht perfekt... Es gibt ein paar unausgerichtete Texturen und ein paar andere Sachen die ich zu fixen vergaß. Ich habe trotzdem an meinen Prinzipien für Texturierung und Beleuchtung festgehalten... geschmeidige Texturen, die eine nette Kulisse abgeben in der Du Deinen Opponenten siehst, Beleuchtung die nicht zu hell oder zu dunkel ist, obwohl durchgehend eine breite Vielfalt an Illuminierung im Level ist. Es gibt einen Mix aus Raumgrößen und Designs.... ein paar Flecken zum Verstecken, weite Tunnel, enge Tunnel. Es ist interessant zu sehen, wie bestimmte Spieler an bestimmten Bereichen im Level abhängen, abhängig von ihrem Spiel-Stil.

Eine gut ausgedachte Custom-Waffe kann ein Level erfolgreich machen... 15 aufällige aber hastig designte und unbalancierte könne es ruinieren. Denke Deine Ideen sehr sorgfältig durch.

Deinen Level promoten

Also hast Du nun den perfekten Anarchy-Level gebaut... ein paar gut ausgedachte Custom-Waffen, ein nettes, kompaktes Level das aussieht wie von einem Profi gebaut. Das wird wohl nicht schiefgehen, oder? Nun...

Damit die Leute Dein Level spielen, müssen sie wissen daß es existiert. Die erste, offensichtliche, Sache ist es das Level an alle Level-Archive einzureichen, die Dir einfallen. Einige werden über Dein Level in ihrer News-Seite berichten, andere werden es einfach zu ihren Archiven hinzufügen ohne überhaupt eine Erwähnung. Das wird Dir ein wenig Publizität verschaffen, aber vielleicht nicht genug. Wenn Du willst daß Dein Level populär ist, mußt Du ein wenig mehr Selbstpromotion betreiben.

Ein guter Weg Dein Level zu promoten, ist es anzukündigen wenn es beinahe fertig ist. Laß die Leute vorzeitig wissen, daß deine fantastische Neuschöpfung bald fertig ist. Du willst daß die Leute über Dein Level bescheid wissen, bevor es überhaupt released ist. Je mehr Leute darüber sehen und hören, desto mehr werden sie es wiedererkennen und interessiert daran sein, wann es fertig ist. Übertreib's nur nicht.

Du kannst Deine eigene kleine Website oder 'Projekt-Seite' anfangen, die Screenshots und Beschreibungen Deines Levels bietet. Es ist eine gute Idee zu warten, bis Dein Level beinahe komplett ist bevor Du Deine Promotions-'Kampagne' startest. Die Leute werden nach einer Weile davon krank 'Coming soon...' zu hören. Schicke Presseaussendungen zu den Spieleseiten (wie GameEdit.com), verlinke sie vielleicht mit Deiner Seite um Screenshots Deiner bald-veröffentlichten-Arbeit zu zeigen. Wenn Du keine Site hast, schicke Bilder zu allen Sites die sie mit ihren News posten werden.

Gehe in die Bulletin Boards und poste News über Dein aufkommendes Level, mit Links zu Screenshots. Du könntest dadurch auch etwas wertvollen Rat von Leuten bekommen. Wenn Du bereit Bist es zu betatesten, lade alle Deine Kumpel ein zu joinen und deren Kumpels. Du wirst nützlichen Rat und Kritik bekommen, plus etwas mehr Promotion für Dein Level.

Schließlich, dies könnte der wichtigste Weg sein Dein Level zu promoten. Wenn es kurz vor dem Releasedatum Deines Levels ist, versuche Zeit auf einem oder zwei guten Dedicated Server anzuberaumen. Es ist entscheidend, daß Dein Level in der Serverliste gehostet gesehen wird, sodaß die Leute einfach joinen und es autodownloaden können, auch wenn sie noch nie davon gehört haben. Versuche soviel Zeit wie möglich an einem Server zu bekommen. Wenn die Leute Deinen Level auf einem populären Server gehostet sehen, mögen sie sich denken 'Gut, wenn Server xyz den Level hostet, muß er recht gut sein'. DwnUnder war so nett, Earthshaker für eine oder zwei Wochen zu hosten. Zur Zeit als er es entfernte, hatten andere Server aufgeschlossen und es gehostet. Wenn es nicht auf seinem Server gewesen wäre, wäre es vielleicht gestorben... ich kann nicht genug betonen, wie wichtig ein Dedicated Server für den Erfolg eines Levels ist. Die Serveradmins sind die Götter des Online-Spielens, speziell für D3... krieche ihnen entsprechend in den Arsch. Die meisten die ich kenne sind echt koole Leute.

Bereit...?

Ok, hier ist die Checkliste:

- Level-Layout ist gut und spaßbringend, Texturierung und Beleuchtung sind perfekt balanciert. Ein Profi wäre neidisch.
- Custom-Waffen sind gut durchdacht, nicht übertrieben und bringen etwas ins Spiel.
- Custom-Schiffe sind gut modelliert und texturiert, außerdem gut miteinander ausbalanciert.
- Ich habe eine Website, und all die Newssites und Webboards wissen daß ich komme!
- Ich habe einen schnellen Dedicated Server, der bereit ist mein Level für eine Woche zu hosten.

Ich bin bereit meinen Level zu releasen! Ist da sonst nochwas was ich zuerst machen sollte?

Wenn du pingelig sein willst, hier sind ein paar andere Ideen:

- Release Dein Level an oder nahe einem Wochenende. Nicht jeder ist wie Du und spielt jeden einzelnen Tag Descent 3. Viele Leute haben ihre freie Zeit am Wochenende und dann spielen sie, dann durchsuchen sie die Sites nach News und neuen Levels (ich habe Statistiken die das beweisen). Wenn Dein Level samstags in allen diesen Listen an der Spitze steht, sind die Chancen besser daß es mehr Leute sehen werden. Wenn Du ein Level montags releast, könntest Du am Samstag bereits von den 'new releases'-Listen verdrängt sein.
- Stelle sicher daß es fertig ist. Es gibt nichts irritierenderes als alle paar Wochen neue Versionen desselben Levels downloaden zu müssen. Versuche Version 1.1, 1.2, Updates, etc. zu vermeiden, baue stattdessen einfach ein neues Level :)
- Sei nett zu den Levelarchiv-Leuten (wie mich :)). Die meisten von uns wollen alle News die wir kriegen können. Anstatt uns 'hier ist mein neuer Level, der ist spitze, poste ihn' zu emailen, versuche ein paar Details in Deiner Mail und Deiner Textfile zu geben, was denn Deinen Level nun großartig macht. Wenn Du eine Custom-Waffe hast, beschreibe Deine Gedanken dahinter und erkläre warum Du denkst die Leute sollten sie versuchen. Wenn Du das kannst, ist es wahrscheinlich, daß Du eine gute Waffe hast. Wenn die Leute hunderte oder sogar tausende Level aussieben müssen, wird jede Information die Du bereitstellen kannst ihre Entscheidung einfacher machen.
- Rufe Deine Großmutter an, Du weißt daß sie sich freut von Dir zu hören.

Viel Glück

Sogar der beste Level kann igoriert werden und in einem Levelfriedhof verloren gehen. Glück spielt hierbei auch eine Rolle. Wenn Du allerdings alle Möglichkeiten abdeckst, wird es eine Chance darauf haben das ultimative Ziel der Level-Unsterblichkeit zu erreichen. Wenn's es nicht richtig schafft, werde nicht zu frustriert. Lerne aus jeder Kritik die Du erhältst und baue ein anderes. Ich weiß nicht was sonst noch hinzuzufügen wäre. Ende.

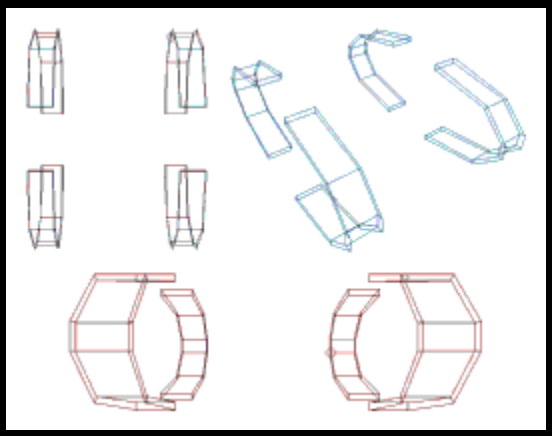
110 - Eye Candy

Robo

Kannst Du dir ein Level vorstellen mit nackten Wänden, schlechter Texturierung und fadem Look das gleich eine gute Review bekommt? Nope. Kannst Du Dir vorstellen, daß es PXO übernimmt und der nächste Hit-Level wird? Unwahrscheinlich. Kannst Du Dir vorstellen daß Du das selbst magst? 'Jo.' (Verdammt, jetzt hast Du den Witz ruiniert 😬)

Ohne gute Ausblicke wird Dein wahrscheinlicherweise in den bodenlosen Abgrund der verlorenen Level gezogen werden... (aka meine ersten vier Levels). Wenn Dein Level nette Anblicke hat, kannst Du sicherlich erwarten daß es einen bestimmten Erfolg erzielt, wenn Du es korrekt releast natürlich ;) Was dieses Tut machen wird, ist Dir ein paar Beispiele von 'Eyecandy' zu zeigen... Der Ausdruck der für Sachen verwendet wird die Du in Deinem Level erzeugst und die es so gut aussehen lassen oder so einzigartig machen. Etwas Eycandy ist fein, wenig Eyecandy ist schlecht, und zuviel Eyecandy kann dich mit partieller Blindheit zum Optiker schicken 😜

Fangen wir an...



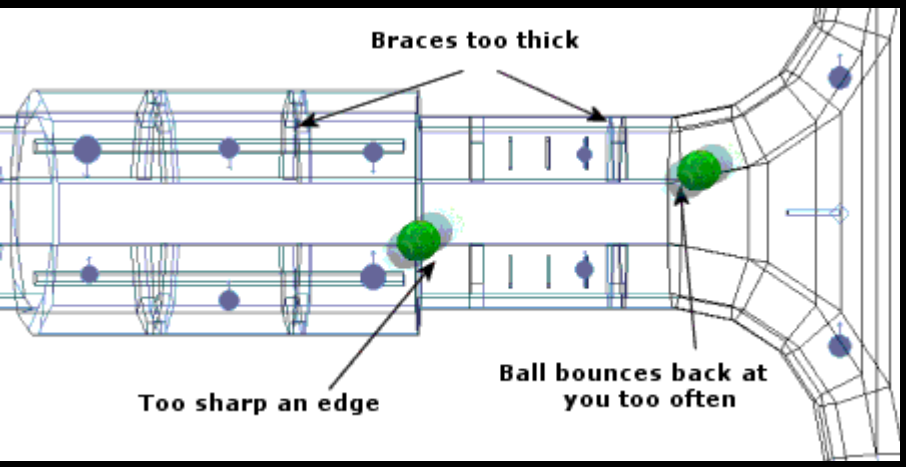
Die Wandverstrebung:

Viel in Dodgeball verwendet und in Dodgeball Enhanced verbessert, auch in all den Outrage-Leveln verwendet, mit unterschiedlichen Designs. Wandstreben sind ein sehr effektiver Weg, ein Level für jedermanns Standards gut aussehen zu lassen, wenn Du sie korrekt machst.

Wenn zu dünn, sind sie einfache Stücke Eyecandy. Zu dick, dann kommen sie einem in die Quere, aber an einem bestimmten Punkt werden sie perfekt werden, ich versprech's! 🙂

Sie sind beides, Fluch und Segen für Leveldesigner. Sie sind gut um sich dahinter zu verstecken und Waffenfeuer abzulenken, aber in Monsterball können sie ernsthafte Probleme verursachen. Wie unten gezeigt:

(Dodgeball Enhanced new wall braces)

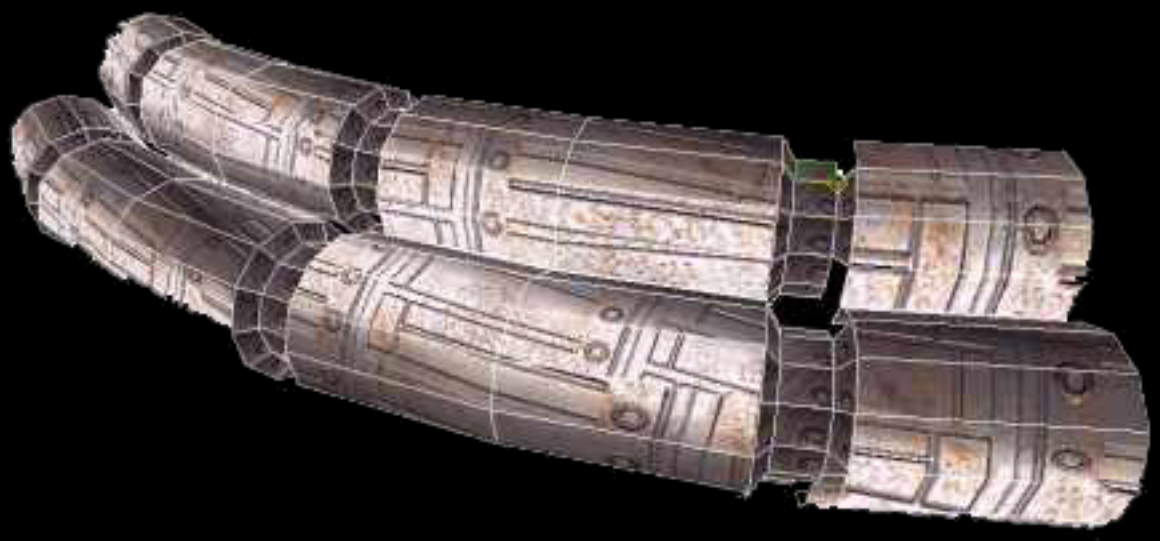


Das Rohr:

Einer der effektivsten und einfachsten Eyecandy-Designs. Das Rohr hat die Macht Dein Level eigenartig realistisch aussehen zu lassen. Ich denke da an Level 2, den Novak Gefängnis-Komplex. In der säuregefüllten Mine hat es zerbrochene und gesprungene Röhren die durch den Fels schneiden, was eine schicke Szenerie macht.

Sie sind auch flexibler als Wandverstrebungen. Du kannst sie leichter benden und verdrehen, und Du kannst auch Vertiefungen einsetzen. Du kannst an einer Röhre eine gläserne Sektion erstellen mit texturierten Faces drinnen, was einen Effekt von säuregefüllten Röhren bringt. Du kannst sogar komplexe Röhrenformen lathen, wie die Röhren (bild) mit Vertiefungen darin. Die Möglichkeiten gehen in die tausende! ☺

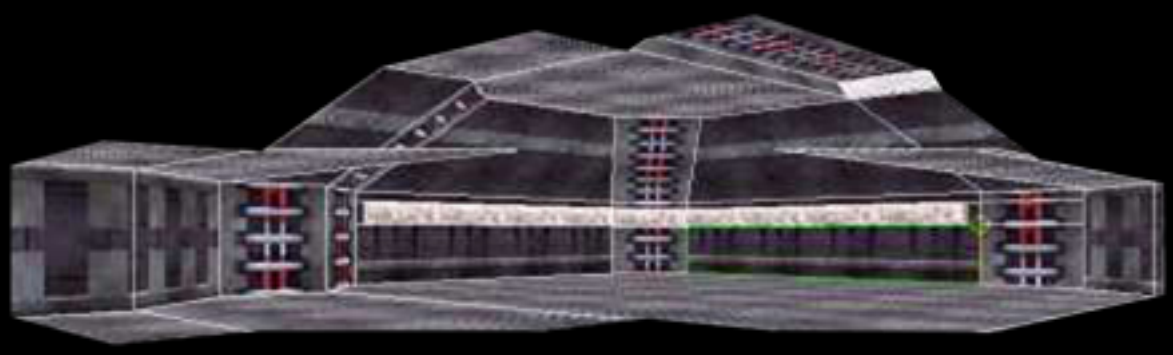
Aber nochmal, paß auf, sie können den Leuten wirklich auf die Nerven gehen wenn Du zuviele hast. Willst Du neben Napalm-Blobs auch noch Röhren dodgen? Ich würde es mal versuchen ☺



Animierte Texturen:

Sie können ein Gefühl von Komplexität in ein Level bringen, was Dich davor bewahrt soviel Eyecandy zu erzeugen. Aber sie könnenauch zusammenarbeiten. Der Raum (bild) Aus dem Outrage-Level 'The Core' (nun, ich habe ihn eigentlich erneut von Grund auf gebaut als mir langweilig war, perfekte Kopie - hehe) hat eine sich bewegende Textur in einer Art 'Wandvertiefung' was es 'technischer' aussehen läßt. Um das noch zu toppen, haben sie ein Halo um die Textur dazugetan. Fantastische Arbeit.

Ich mag es, animierte Texturen zu verwenden um raumfahrtzeitalter-mäßige Themen zu erzeugen. Ich mache Energiestrahl-Reaktoren, Beleuchtung, Teleporter, sogar Toaster ☺ Versuche nur, sie zu Deinem Vorteil zu verwenden, abhängig davon was für eine Art Level Du baust.



Der Spew (sag 'eeeeewwww'):

Ahem. Spew wird von Spew-Emittern emittiert (O_o). Ein Spew-Blob kann ein Energiepartikel sein, Feuer, Wasser, Dampf, Rauch oder sogar Schnee. Der Spew im Bild (Bild) paßt nett in die Szene... Ein großer Gefechtsraum mit vielen Frag-Turrets die auf Dich schauen, und Feuer daß aus den Wänden kommt - perfekt!

Ich muß nur noch sagen, der Pyro bin nicht ich, da ich im Traum nicht daran denken würde in diese Richtung zu fliegen und ich fliege sowieso eine Skipnix (Robo wird nicht von Turrets ins verlängerte Kreuz getreten, anders als dieser Kerl...)

111 - Waffenbalance

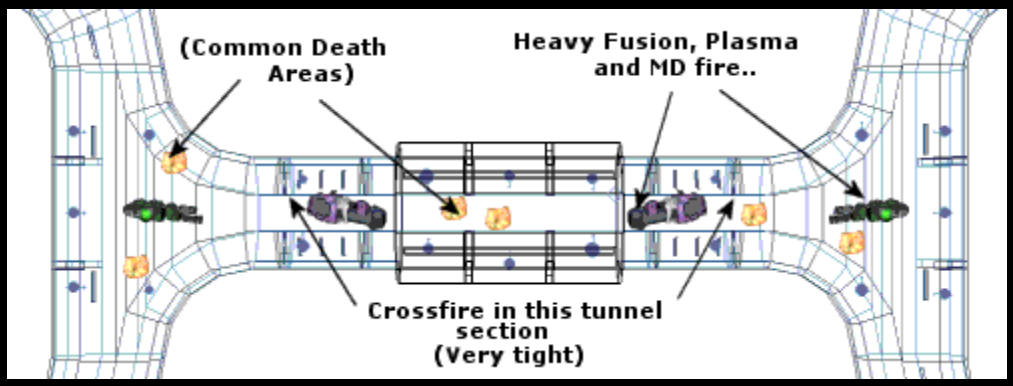
Robo

Waffenbalance kann der Unterschied zwischen einem großartigen Level mit fantastischem Gameplay und Top-Reviews und einem mit armseligem Gameplay und Reviews das Du lieber nicht sehen willst. Ich scherze nicht ☺ (oder doch? *ahem*)

Ich habe dieses Problem selbst aus erster Hand in einem meiner Level erlebt. Es heißt Dodgeball, ich bin sicher daß Du etwas hier irgendwo gesehen hast, daß damit zu tun hat. Es hatte sehr ernsthafte Waffenbalanceprobleme, welche easy-Kills im Hauptkorridor verursacht haben, der die jeweiligen Seiten des Levels verbindet. Das war eigentlich ein richtiges Chaos aber es war für manche zu einfach ☺

Hier ist eine Art 'Schlachtplan' der Problemzonen, der Problem-Waffen und der Ort wo du höchtwahrscheinlich in die Hölle gefraggt wirst. Hübsch illustriert 😁

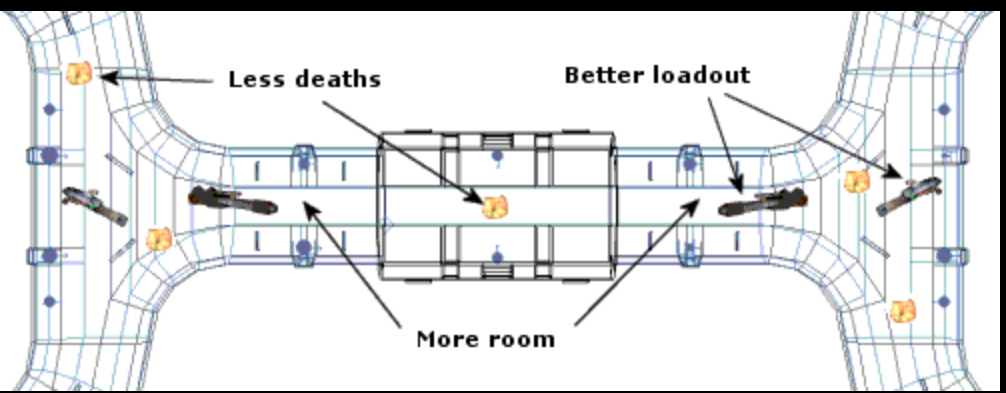
Du kannst ganz einfach die Gründe sehen. Erstens, der Tunnel ist extrem schmal, zweitens sind da zuviele 'spray-&-pray'-Waffen und drittens, sind da zuviele Waffenspawnpunkte um dieses Gebiet herum.



Ich nenne das die **Flaschenhals-Theorie** :)

Die Flaschenhals-Theorie kommt ins Spiel wenn Du einen engen Tunnel hast, und dieser der einzige Weg quer durch den Level ist. Auch häufen sich die Waffen auf beiden Seiten, so daß das Gebiet ein großer Schlachtplatz wird. Wenn Du es während eines intensiven 4-vs-4-Spieles namens 'Schmeiß den Mortar' durch den Haupttunnel in Dodgeball schaffst und mit mehr als **10% Schild** 'rauskommst, kriegst Du von mir eine Medaille und ich verkünde das dann 😜

Die Flaschenhals-Theorie ist immer noch Teil der Revision, Dodgeball Enhanced, aber nicht so markant. Es gibt nicht länger irgendwelche Orte zum decken und snipen, es gibt weniger Waffenspawnpunkte und die Waffenbeladung wurde geändert. Anstatt eines reichlichen Angebots an Smarties, Plasma, Homings und Fusion, hast Du nun eine neue Beladung an Vauss, MD, Frags, Super Laser und Concussions. Die Geometrie ist allerdings die selbe, siehe links:



Schlußendlich entscheiden der Spielmodus und die Eigenschaften des Levels welche Waffen am besten sind. Am besten gibst Du vielleicht ein paar Frags in ein enges Level, aber vielleicht keine Smarties. Es ist eine schlechte Idee, eine Plasma in ein Outdoorlevel hinzuzugeben, man wird nichts treffen. MD oder Vauss wären eine weit bessere Wahl.

Hier ist eine Liste von empfohlenen Waffenladungen für einige Spiel-Modi:

- **Anarchy**: Alles, es hängt von Deinem Level ab. Nimm nur nicht zuviel von etwas.
- **Team Anarchy**: Das gleiche, wirklich ☺
- **Capture The Flag**: Gib viele Secondaries in den Bereichen um die Bases, allerdings nicht zuviel - und eine gute Menge an Primaries in den Dogfighting-Gebieten.
- **Monsterball**: Strange. Viel Vauss und MD kann dem Level einen Gefallen tun - oder es einfach ruinieren. Super Laser und Concussions sind ebenfalls meine Favoriten in Monsterball-Levels.

Kurz und bündig, aber akzeptierbar ☺

Versuche einfach die Bottleneck-Theorie anzuwenden, wenn Du sie kennst, wird das dem Gameplay Deines Levels helfen. Betatests werden wahrscheinlich die meisten Deiner Bugs ausbügeln ☺

112 - Warum Missionen bauen?

Robo

Viele Leute fragen mich, warum noch immer D3-Levels bauen, nach fast vier Jahren? Nun, die Antwort ist so einfach wie "ich mag's einfach." oder "Spaß"; tatsächlich ist es eine relativ komplizierte Antwort... Es gibt viele verschiedene Gründe warum ich und viele andere das noch immer machen, und nun werde ich Dir ein paar Beispiele geben...

Während meines D3-Lebens tendiere ich das Zitat zu verwenden "Du kriegst keinen sicheren Erfolg geschenkt, Du kämpfst für Dich allein.", und das funktioniert sehr gut für mich. Übersetzt in Leveldesigning-Begriffen bedeutet das daß Dir der sichere Erfolg nicht mit einem einfachen Level gegeben wird, nur eines einfaches, flachwandiges, langweiliges Level... Du mußt es jedesmal härter versuchen, Dich durch Deine Fehler verbessern, Tips und Tricks auffinden, Deinen eigenen Arbeitsplan optimieren um effizienter zu sein. Wenn Du nicht glaubst, daß Du das machst, nach allem habe ich zwölf Levels und eineinhalb Jahre gebraucht um das richtig hinzukriegen ☺

1.

Ich denke, das dürfte das wichtigste aller meiner Argumente sein, also kommt es als erstes. Alle Spiele sterben langsam weg, Du mußt das eines Tages realisieren. Obwohl Du und ich das nicht glauben wollen, Descent3 könnte in zehn oder 15 Jahren lange weg und von allen vergessen sein, außer den Fanatikern wie Dich und mich ☺

So wie ich das sehe, um Leute an D3 interessiert zu halten, mußt Du die Community aktiv halten. Kannst du sagen, wieviele Leute D3 in seiner Box sitzenlassen haben, weil sie wenig Aktivität in der Community gefunden haben und dann darauf vergessen haben? Vielleicht viele. Meine Theorie ist, um eine schrumpfende Community so lange als möglich zu erhalten, müssen die Leute weiterhin am Levelbauen dranbleiben. Das hat in bestimmten Punkten keinen großen Effekt, aber es verzögert den Tod einer Community genug.

Das ist einer der Hauptgründe warum ich "Robo's D3Edit Tutorials" gemacht habe. Zu versuchen die Leute mehr dazu zu kriegen unbedingt D3Edit zu lernen und diese überaus wichtigen Level zu bauen. Ich muß dazu sagen, wenn D3Edit ein anderes Programm wäre, Descent3 würde jede Woche einen Haufen Levels bekommen. Die einfache Tatsache ist, es ist kein einfaches Programm zu lernen. Vieles ist nicht offensichtlich und schwer zu erkunden.

2.

Mein nächster Grund ist es den Leuten zu helfen. Ich startete vor drei Jahren mit ein paar Boxen los, mit zufälligen Texturen, klaffenden Löchern in den Wänden, und ich wußte nicht wie ich irgendetwas testen sollte, was ich gemacht habe. Dann entdeckte ich Tutorials und begann meine Level zu bauen. Über drei Monate hinweg kreirte ich vier Levels (offiziell drei) und dachte ich bin nicht stolz auf sie, Du kannst sagen sie sind etwas besonderes für mich :) Dann nach einem Jahr wurde ein Level namens "Dodgeball" geboren. Das brachte mir große Zuversicht ein und brachte mir noch mehr Erfahrung. Ich lernte dann, aus meinen Fehlern noch mehr zu lernen als zuvor. Nach einem Monat wurde meine feinste Arbeit, "Dodgeball Enhanced", released. Die Leute mochten es (und mögen es noch immer), und es wird manchmal von Lawrence 'DwnUndr' Britt gehostet ;)

Nun bin ich ein Level-Designer für Orbital Design Studios. Arbeite mit einem Programm das VIEL komplexer als D3Edit ist. In einem Jahr oder zwei räkle ich mich vielleicht im Geld, aber sogar wenn wir versagen (und wir werden nicht :)) werde ich das in meinem Lebenslauf halten für zukünftige Jobs. Ich könnte einen Modellierer-Job bekommen oder in eine große Software-Company kommen - man weiß nie!

Meine Sichtweise ist, sieh wo ich angefangen habe und sieh wo ich heute bin. Dank ein paar einfachen D3Edit-Tutorials, Level-Erfahrung und Hilfe von meinen Kumpels arbeite ich nun für eine Firma

Wo könnte es dich hinbringen?

3.

Vergnügen und Erfahrung. Vielleicht der wichtigste Aller Gründe ist Vergnügen und Erfahrung. Das knüpft auf viele Arten an #2 an. Level-Designing, obwohl es von außen schwierig, kompliziert und unmöglich scheint, ist eigentlich (beinahe) recht einfach und vergnüglich. Es ist genau wie seine eigene Welt zu kreieren, auf gewisse Art 'Gott' zu sein. Du sagst wo alles hinkommt, was zu tun ist, und Du kannst Deine Subjekte nach belieben vernichten

4.

Ich denke, mein letzter Punkt ist's kool auszusehen, aber ich bin sicher wie Du das zu machen hast

Inspiriere Leute, Lerne die Wege von D3 und D3Edit, scheue Dich nicht die Grenzen herkömmlichen Denkens zu überschreiten, und vor allem - Sei kreativ! Und viel Glück!

Abschnitt K – Single Player

Was man noch braucht, um spannende Solomissionen zu machen

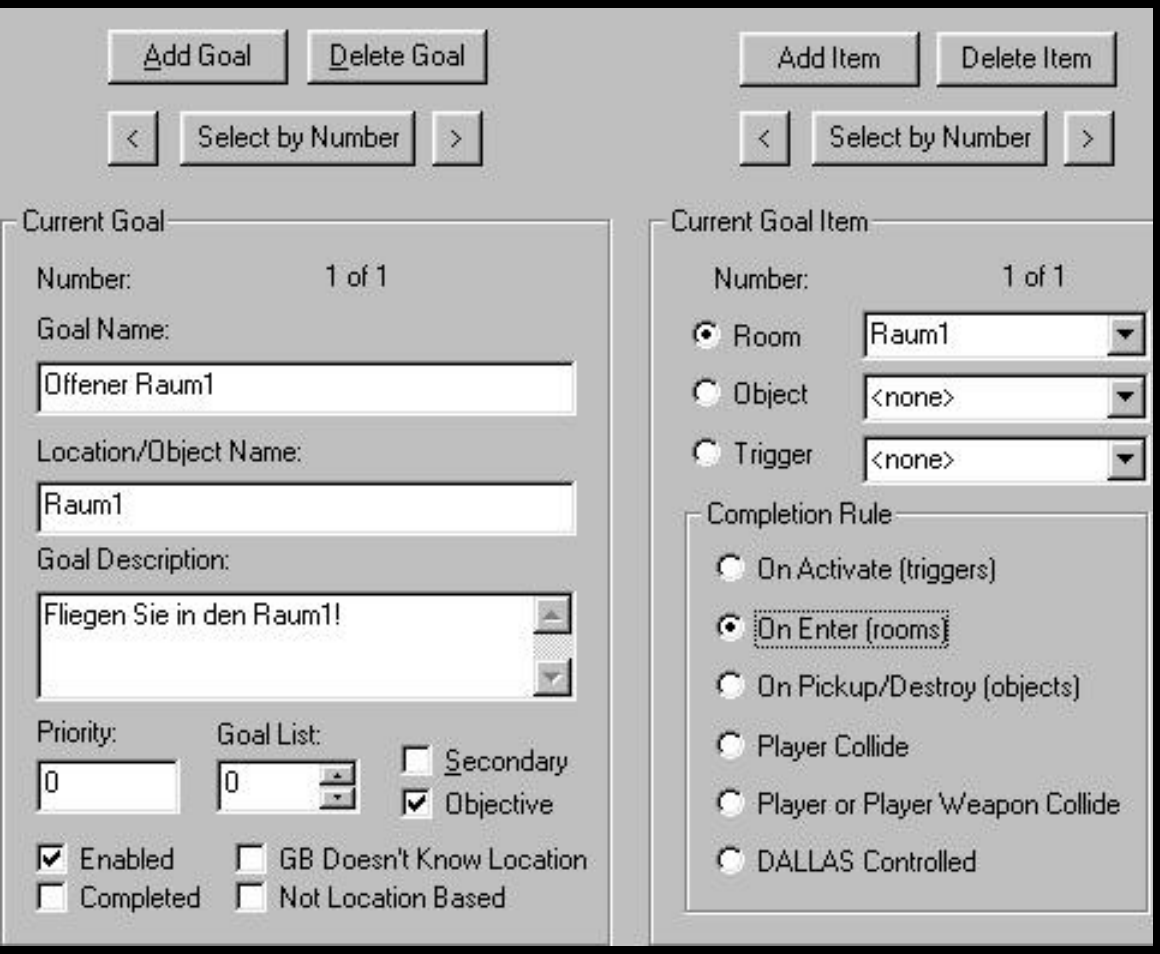
113	**Level-Ziele**	Aufträge im SP	Blackshark	397
114	**Roboterzentren**	Es werden Bots!	Blackshark	398
115	**Pfade**	Bots wohin schicken	(LL)Atan	400
116	**BNode-Tutorial**	Damit der Guidebot nicht immer hängenbleibt	Kyouryuu	404
117	**BNodes setzen**		(LL) Dark	406
118	**Player-Respawnpunkte - toprog**	toprog	xxx	410
119	**Loadscreen einbinden**	Das Standard-Ladebild ersetzen	Robo	413
120	**Levelbriefing**	Eine Story in den Level bauen	RagilRal	415

Toc

113 - Level-Ziele

Blackshark

Hallo, ich erkläre Euch jetzt, wie man Ziele in den Level einbaut. Weil wir uns nicht so viel Arbeit machen wollen, einen neuen Raum zu erstellen, klickt auf das leere Blatt oben links im Menü (File/New...) und auf "Default Room". Nun geht in die World View-Ansicht und stellt"Textured with Outline" ein. Das aktuelle Face ist mit "Outrage Test" beschriftet. Wählt Euch eine andere Textur aus, ich habe die "**Brown Volcano Rock**" genommen.
Drückt auf **Einfg** auf der Tastatur. Jetzt klickt Ihr auf "Room Properties" im Menü "Window" und betätigt den Knopf, auf dem "None" draufsteht. Wir nennen den Raum "Raum1".



Anmerkung: Um Objekte oder Roboter zu benennen, klickt Ihr einmal drauf und seht oben rechts im Current Level Objekt Fenster (Objekt Bar), was ausgewählt ist. Drückt den Knopf auf dem der Name des Objektes steht und gebt dem Objekt, oder Roboter einen Namen. (Objekte benennen)

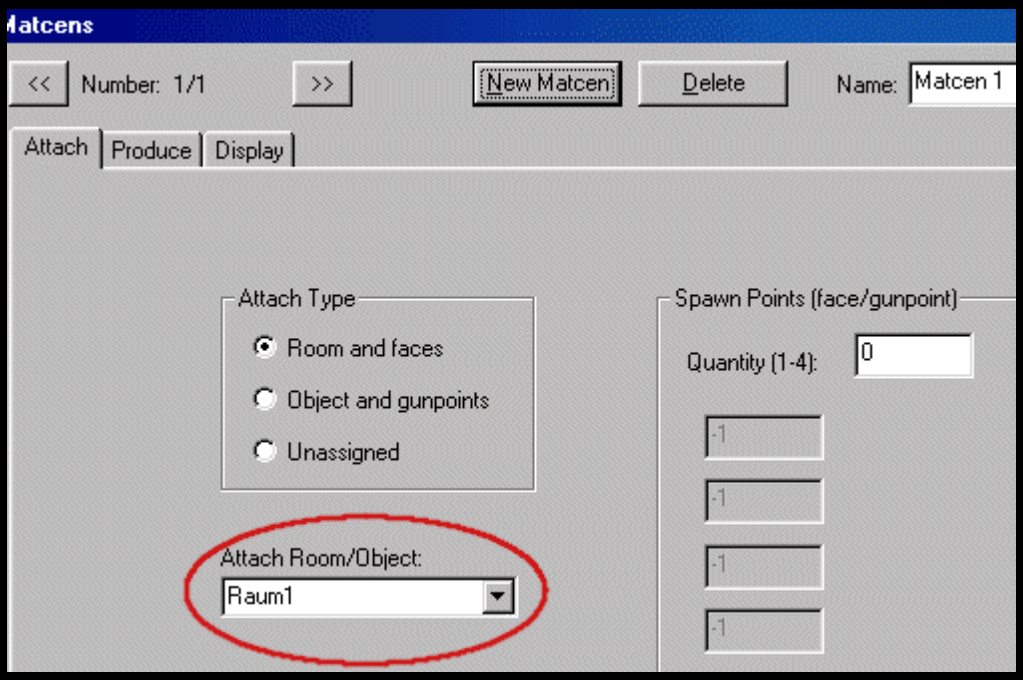
Nun öffnet das Goal Fenster im Menü "Window" unter "Goals", drückt "Add Goal" und füllt die zwei Spalten und das leere Textfeld wie im Bild aus. Das Missionsziel soll sein, einen Raum zu erreichen, bzw. zu betreten. Also, was muss wohl angeklickt werden??? Ist klar, "On Enter (rooms)".

Speichert den Level und berechnet die Lichter. Dann speichert ihn als eine MN3-Datei ab und startet Descent3. Wenn Ihr den Level spielt und den Raum betretet, werdet Ihr sehen, was passiert! Ich jedenfalls liebe das!!! Wenn Ihr noch Fragen oder Probleme habt, schickt mir eine Email, ich versuche Eure Fragen zu beantworten.
Ich werde Ziele auch noch etwas näher erklären, was z.B. "**Not Location Based**" heißt. Schreibt mir, und die Antwort wird Euch bald erreichen.

Tschüß
Euer Manuel

114 - Roboterzentren

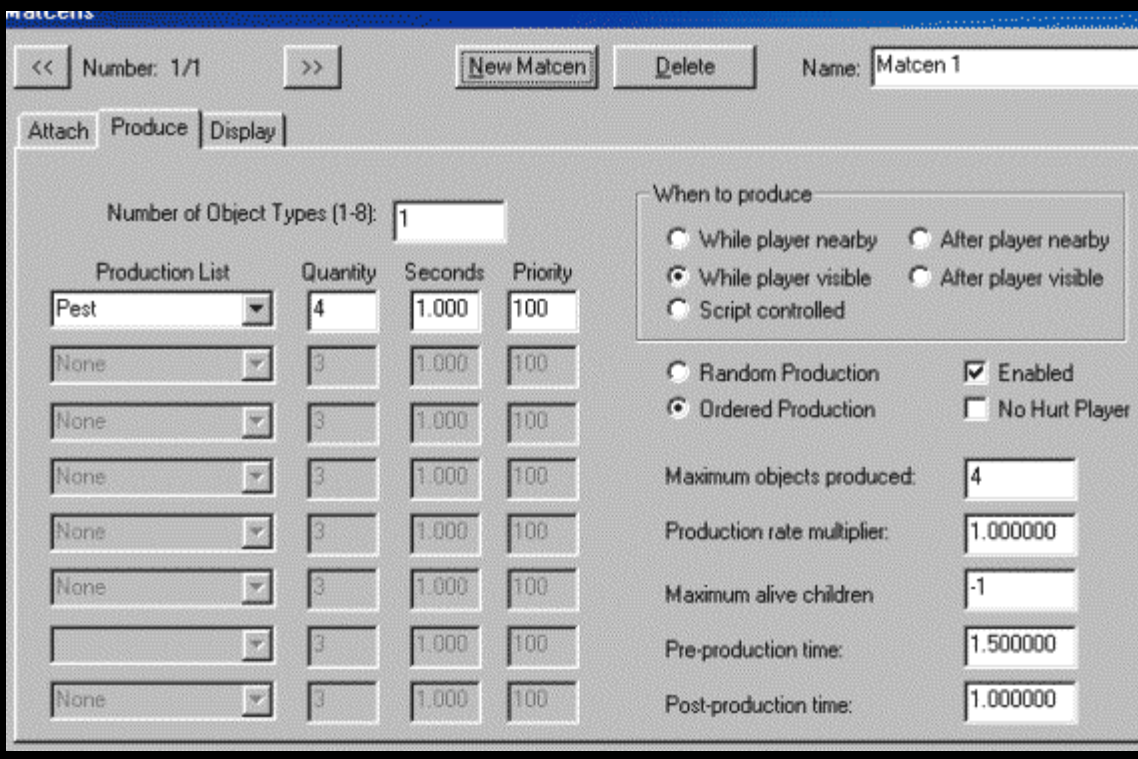
Blackshark



Hallo! Ich werde Euch hier erklären, wie man Roboterzentren baut. Also, klickt auf "New" und "Default Room". Geht in die World View Ansicht und wählt Euch eine beliebige Textur aus. Dann drückt **Einfg** auf der Tastatur oder im Menü "Room", "Add room at current room". Jetzt geht im Menü "Window" auf "Room Properties" und benennt den Raum als Raum1 (Knopf "none" anklicken). Um ein "**Matcen**" (so wird ein Roboterzentrum in der Editor-Sprache genannt) zu erstellen, drückt wieder den Menü "Window" und dann "³". Drückt "**New Matcen**" und füllt alles aus wie es im Bild oben zusehen ist.

Unter dem "Attach Type" wird eingestellt, in welchem Raum, an welchem Punkt oder Objekten Roboter gebaut werden sollen.
Unter "Display" könnt Ihr angeben, welche Grafik und welches Geräusch während des Bauens verwendet werden sollen. Zunächst müssen wir angeben was und wie viel an Robotern oder Powerups produziert werden soll.

Schaltet das Menü "**Produce**" an und füllt alles aus, wie im Bild rechts.



Damit Ihr auch Eure eigenen Roboterzentren erstellen könnt, also nicht wie dieses, werde ich jetzt ein paar Begriffe erklären:

Quantity	Damit kann man einstellen, wie viel von jeder Art eines Roboters oder eines Powerups produziert werden sollen.
Number of object types	Dort kann man einstellen, wie viel und was für verschiedene Arten von Robotern hergestellt werden sollen.
While player visible	Das bedeutet, dass sich die Roboter bilden sollen, während Ihr sichtbar seid.
While player nearby	Das Roboterzentrum fängt an zu produzieren, wenn Ihr nah an der Quelle dran seid.
After player visible und After player nearby	Damit ist gemeint, dass nachdem das Schiff sichtbar bzw. nachdem es nah an der Quelle dran war, die Produktion der Objekte beginnt.
Script controlled	Man muss mit dem DALLAS Script-Editor umgehen können, wenn man z.B. angeben will, dass die Produktion starten soll, wenn eine Tür geöffnet wird, oder anderes.
Maximum objects produced	Dort kann man einstellen, wie viele Objekte INSGESAMT produziert werden sollen.

Mit dem Rest habe ich mich noch nicht beschäftigt, aber Ihr könnt ja mal ausprobieren was passiert, wenn man "Random Production" anklickt. Zum Schluss berechnet den Level und speichert ihn als MN3-Datei ab. Fertig!! Wenn Ihr noch Fragen oder Probleme habt, schickt mir eine Email, ich versuche Eure Fragen zu beantworten.

Viel Spaß beim Bauen!

115 - Pfade

(LL)Atan

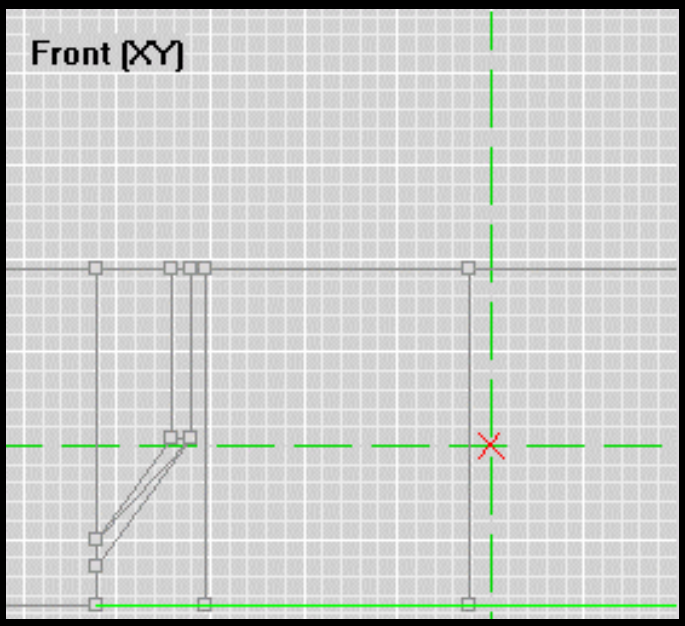
Diese Anleitung zeigt wie mit D3Edit Pfade angelegt werden. -

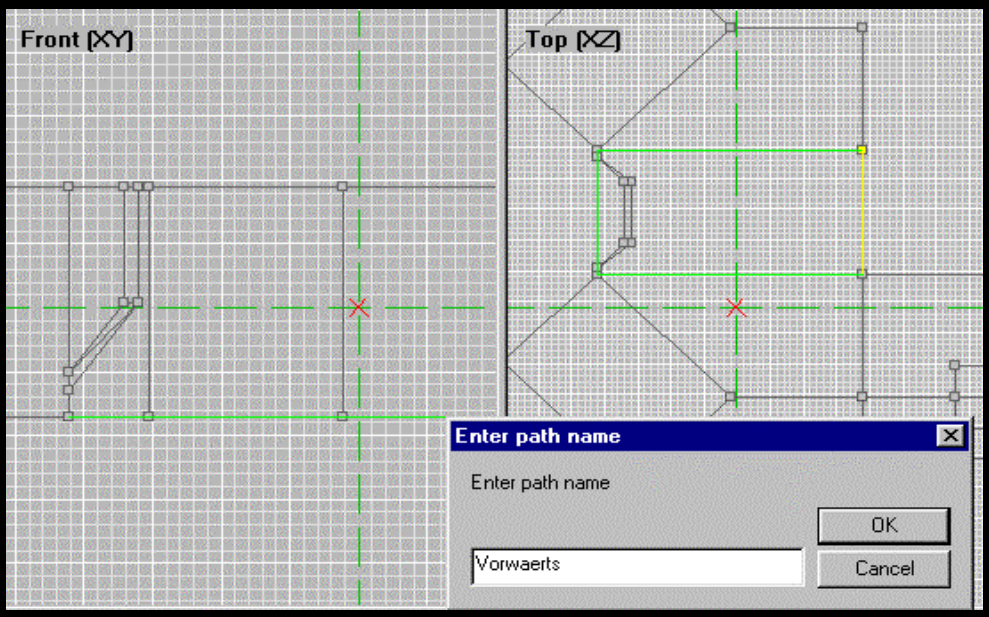
Öffnet Euren Level und wechselt in die aktuelle Raumansicht. (Current Room View)

Schaltet 'Textured View' oder 'Textured with Outline View' Ansicht ein.

Nun den Pfad-Modus einschalten (**Strg-H**):

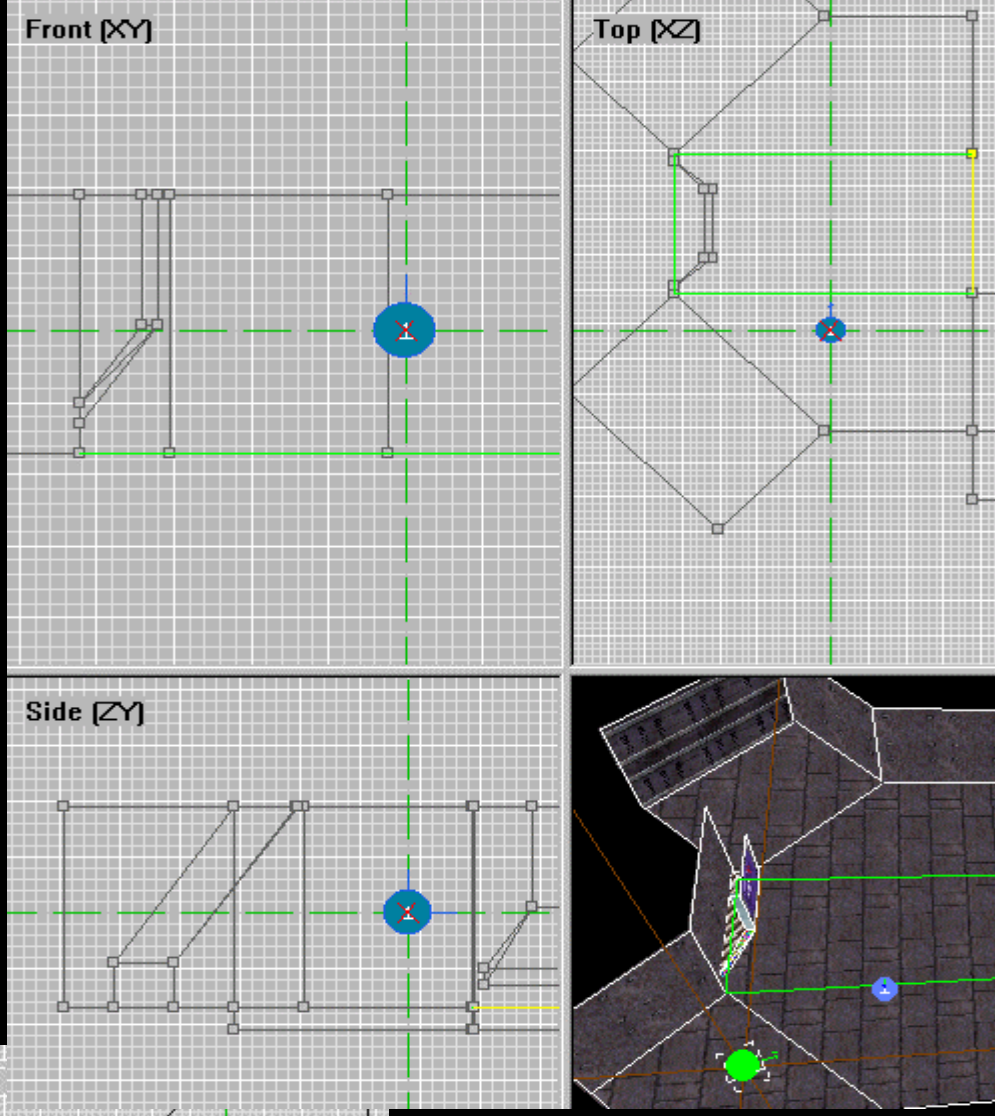
Das **X** mit **Strg + Pfeiltasten** an die gewünschte Position bringen.
Eines der Front-, Top-, Side-Fenster muß aktiv bleiben.

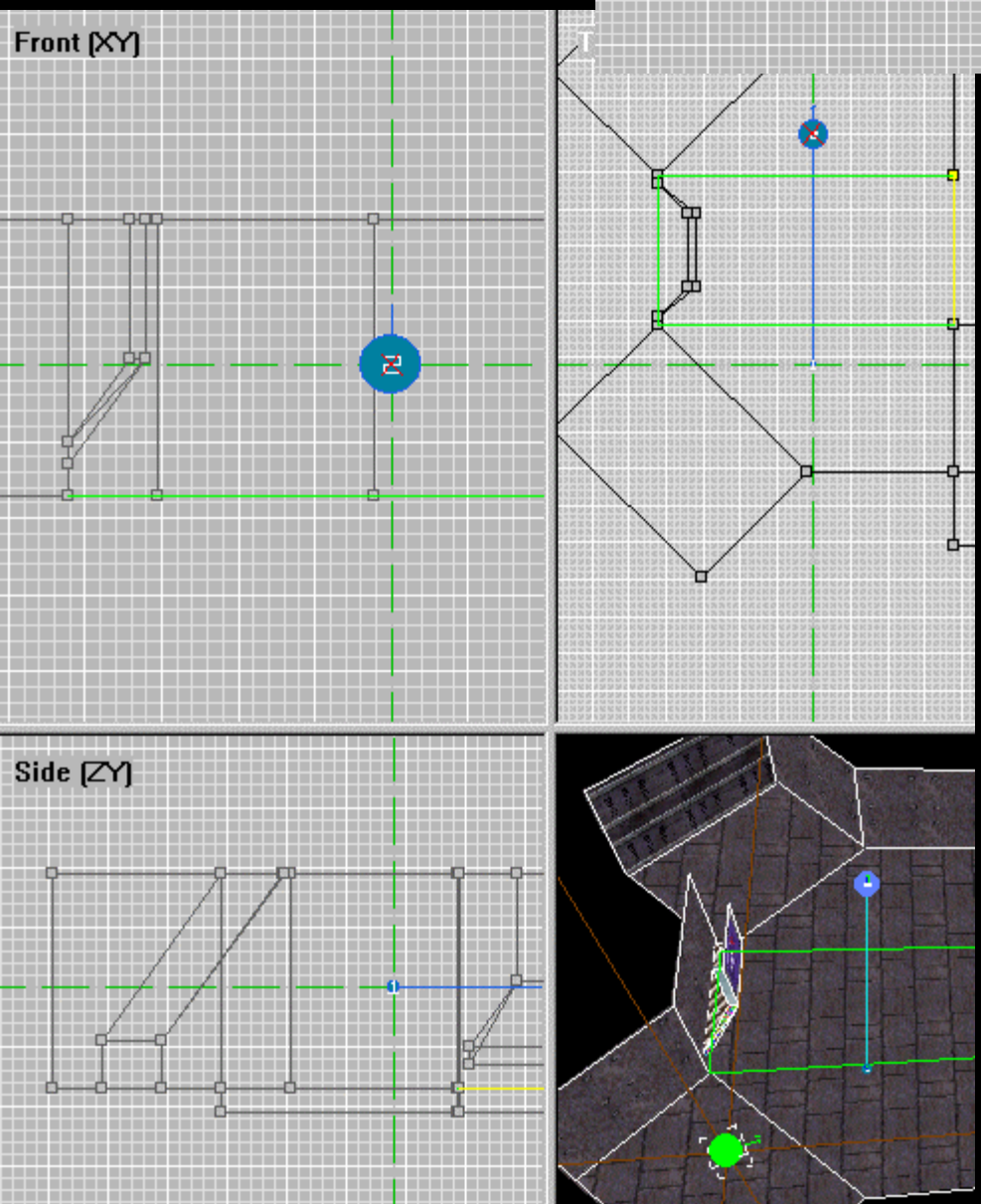




SHIFT+Einfg um einen Pfad anzulegen.
In dem nun erscheinenden Eingabefeld den gewünschten Pfadnamen eingeben. (links)

Nach der Eingabe wird der Pfad mit einem einzelnen Pfadknotenpunkt (Nr.1) an der Zielposition (**X**) erzeugt.

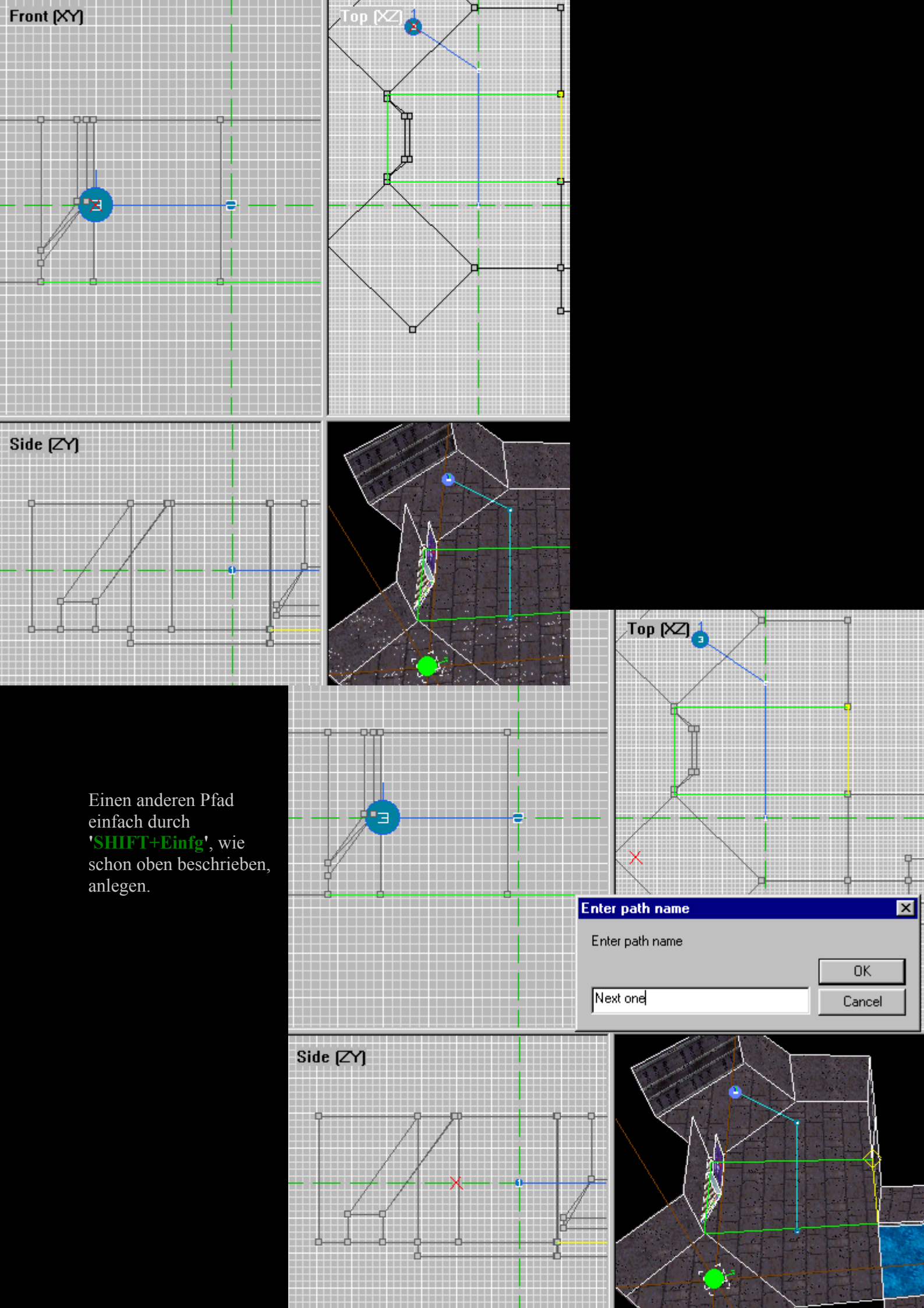




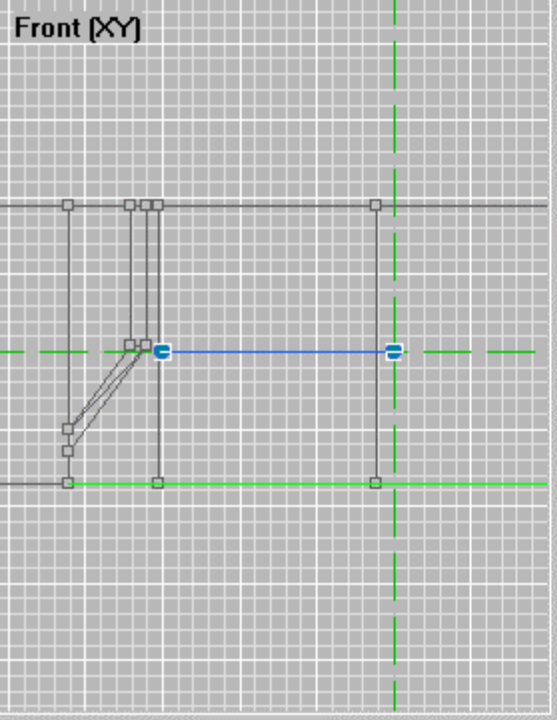
Der neue PfadKPkt wird jeweils nach dem aktuellen PfadKPkt eingefuegt.
(Dafür muss ein PfadKPkt aber erstmal aktiv sein (blauer Kreis)).

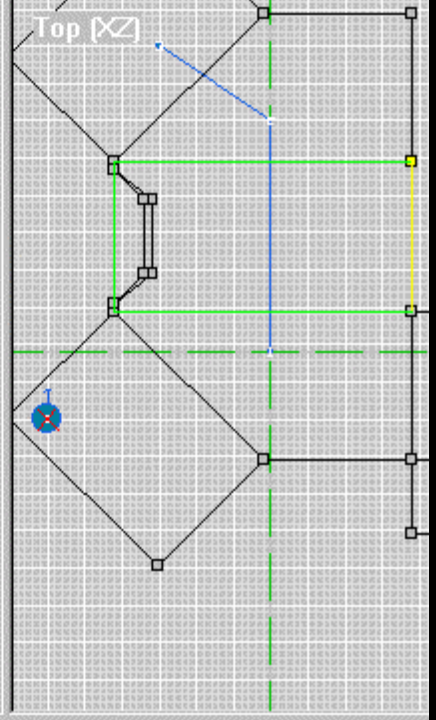
Um einen PfadKPkt zum aktuellen PfadKPkt zu machen, einfach in die Nähe des gewuenschten klicken oder mit '**N**' und '**SHIFT+N**' durch die PfadKPkte gehen um den richtigen anzuwählen.
(Dafuer muss ein PfadKPkt aber erstmal aktiv sein (blauer Kreis)).

Das '**X**' mit '**Strg' + Pfeiltasten** an die näheste gewünschte Position bringen.
Mit '**Einfg**' kann man nun weitere Pfadknotenpunkte eingeben. (Nr...2..3...usw)

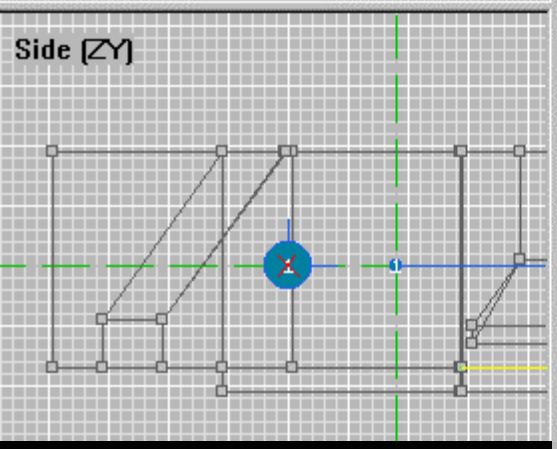


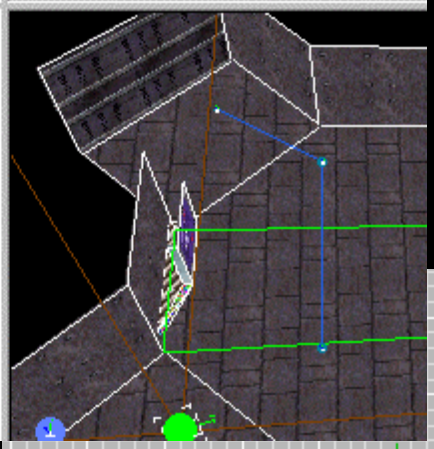
Einen anderen Pfad einfach durch '**SHIFT+Einfg**', wie schon oben beschrieben, anlegen.





Wenn mehrere Pfade im Level sind, dann kann man mit '**H**' und '**SHIFT+H**' die Pfade anwählen.





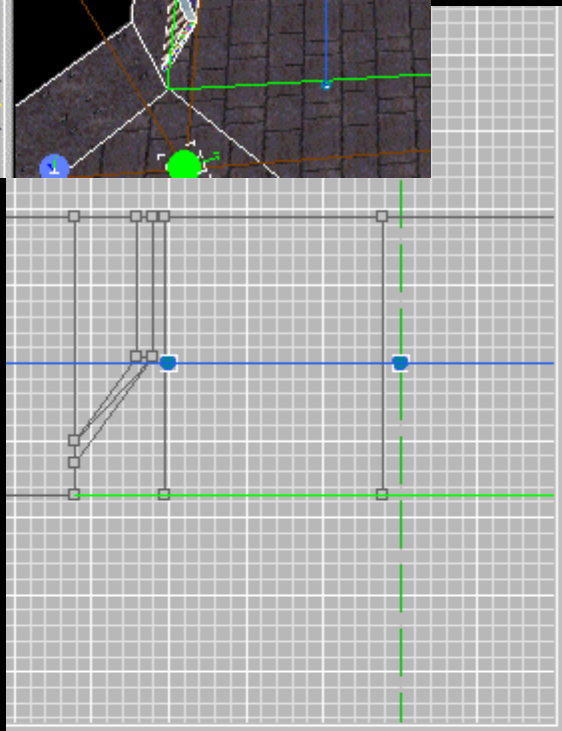
In der Statusleiste oder im Nachrichten Fenster (1.1 Beta 8) kann man sehen welchen Pfad,
Punkt man gerade angewählt hat.

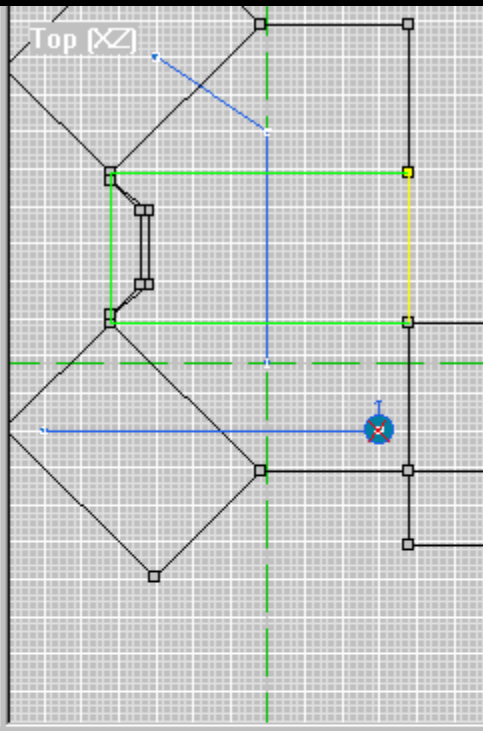
Path 1, Node 2 selected. Path Name = Next one

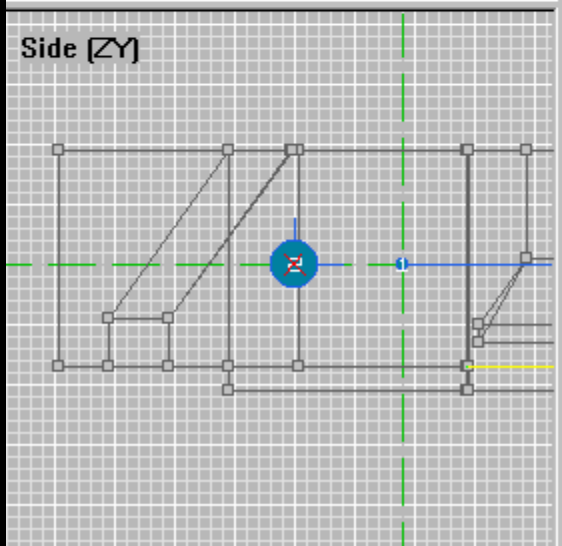
Bearbeiten kann man die PfadKPkte im Pfad-Modus und dann wie üblich mit den Nummernblock Tasten. Eines der 2d-Fenster muss dafür aktiv sein.

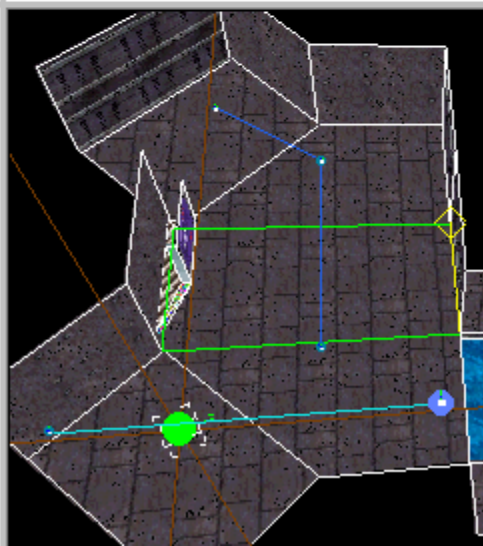
Löschen der PfadKPkte ist mit '**Entf**' möglich. Ist der letzte PfadKPkt eines Pfades erreicht, fragt der Editor nach ob der Pfad gelöscht werden soll.

Viel Erfolg
(LL)ATAN









116 - BNode-Tutorial

Kyouryuu

überarbeitet

Lerne hier was zur Hölle BNodes sind und wie man sie verwendet. Dieses Tut wurde im Jänner 2003 überarbeitet.

Einleitung

Damals, in D1 und D2, als alles aus Cubes aufgebaut war, war die Navigation in den Levels nicht schwierig, sogar für die Robots. Setze denGuideBot in ein Level und er weiß automatisch wohin. Setze einen Bulk Destroyer in ein Level und setze sein Behaviour auf Snipe, und er wird die besten Orte dafür finden. Laß mich nicht vom Thiefbot anfangen. Es gab praktisch nichts, das Du den Robots sagen mußtest wohin sie gehen sollen.

In Descent 3 liegen die Dinge komplizierter. Die KI der Spiel-Feinde und des GuideBots könnte schlauer sein wenn sie Dir erst begegnet - aber um diese riesigen Minen zu navigieren, brauchen sogar die Bots eine Karte. Das ist es wofür die BNodes eigentlich da sind. BNodes sind ein Leitsystem für die KI - eine Serie von Wegpunkten denen die KI in ihrem Bestreben folgen kann, von Punkt A nach Punkt B zu gelangen.

Die KI ist sehr genau bei den BNodes. Wenn keine in der Mine sind, bestehen Chancen daß sich Deine Bots wie komplette Vollidioten verhalten werden. Mehr noch, der GuideBot wird nicht kapieren wie man von einer Objective zur nächsten kommt. BNodes sind essentiell für Singleplayer- aber dienen keinem Zweck in Multiplayer-Levels.

Bevor Du beginnst

Abhängig von der Größe Deines Levels, können BNodes eine recht zeitintensive Aufgabe sein, speziell da D3Edit anscheinend regelmäßig crasht, wenn man sie manipuliert. Speichere oft! Bevor Du anfängst, da sind ein paar Szenarien die ausgiebig behandelt werden sollten um BNodes so einfach als möglich zu machen, und auch ein paar Warnungen.

- Erstens, Habe niemals Portale die nicht zugänglich sind. D3Edit setzt eine BNode an jedes Portal! Die KI ist nicht schlau genug um zu verstehen daß ein Portal nicht zugänglich ist und wird eher beharrlich versuchen ins Hindernis zu rammen, als einen Weg herum zu finden.
- In D3Edit gibt es ein paar unvermeidbare Situationen wo der Guidebot sich einfach dumm verhält. Besonders, wenn Du mehr als ein Portal in einem Raum hast, aber nur eins passiert werden kann. Zum Beispiel, nehmen wir an wir haben einen Raum mit sechs Portalen, aber fünf davon sind per Skript mit Glas blockiert. Anstatt das freie Portal zu suchen, wird der Guidebot in die verglasten rammen. Manchmal kapiert er es wenn Du ihn beschießt oder ihn dazu bringst Dir zum freien Portal zu folgen. Dann wird der Guidebot die korrekte Serie an BNodes zusammenlegen und von da an losfliegen. Natürlich beides negiert den Sinn darin, den Guidebot von Anfang an zu haben. Ich bin nicht sicher, aber ich habe so ein Gefühl daß Outrage ausgefeiltere Vorhaben mit den BNodes hatte als die die Original-D3-Levels designten, als die stumpfe Jedes-Portal-Kriegt-Eine-BNode-Strategie die verwendet wird.
- Räume sollten alle ihre Portale recht zugänglich von anderen Portalen haben. Das heißt, baue keinen 360°-Ring-Raum. Optimiere nach zwei 180°-Räumen oder vier 90°-Räumen. Das ist ohnehin schlaue Level-Optimierung.

BNodes verwenden

Zurück zur anstehenen Aufgabe. Nehmen wir an, Du hast einen Singleplayer-Level und Bist bereit Deine BNodes einzusetzen. Das solltest Du tun, wenn Dein Level fast fertig ist - mach es nicht, solange Du noch daran baust. Klicke P um das Path-Paneel zu öffnen. Wir wollen die BNodes manipulieren, also klicke auf den AI Paths (BNodes)-Radiobutton.

Du könntest Dich fragen, für was wird der andere Pfad-Typ verwendet? Diese Game paths werde für Aktivitäten verwendet wie Roboter auf spezifischen Pfaden durch die Mine zu führen, wo ich will daß sie konsequent einem sehr wohlüberlegten Pfad folgen. Sie werden auch verwendet, um die Kamera für die Ingame-Cinematik zu führen.

Klicke auf Create **AI Path Nodes**. Ein kleiner Dialog sollte danach aufgehen (wenn nicht, öffnet ihn der Button **Verify Nodes**). Der Dialog sagt Dir, was alles falsch ist dabei wie die Sache geendet hat nachdem D3Edit die BNodes auto-eingefügt hat. Idealerweise willst Du hier eine 'VERIFY OK!'-Nachricht, aber eher als nicht wirst Du eine Liste von Problemräumen bekommen und den fragliche BNodes die Probleme haben. Der Schlüssel ist, daß Du sicherstellen mußt daß jede BNode zumindest eine andere in der Mine erreichen kann. Du kannst keine von denen einfach herumliegen lassen, vom System abgeschlossen.

Zum Schluß, die Basisstrategie ist es entweder existierende BNodes in der Room View zu bewegen sodaß sie die anderen erreichen, oder auf dieselbe Art neue BNodes zu erzeugen. Nachdem Du eins davon gemacht hast, klicke auf den Auto Edge Room-Button (aktualisieren) um zu sehen ob alles funktioniert hat. Wenn nicht, wiederholen. Es gibt allerdings ein paar Vorsichtsmaßnahmen:

Um einen BNode auszuwählen, mußt Du in der Room View Textured with Outline eingestellt haben und mit der Maus auf den BNode klicken. Du mußt auch im Path-Mode sein (**Ctrl-H** oder auf den Button in der Toolleiste klicken). BNodes können nicht über die 2d-Ansichten angewählt werden. Allerdings können sie, wenn sie erst einmal angewählt sind, darin bewegt werden.
Um einen BNode zu bewegen selektiere ihn und verwende die Tasten auf dem Ziffernblock um ihn zu verschieben.
Verwende nicht mehr BNodes als das absolute Minimum. Andernfalls wird die KI verwirrt.

Halte das im Hinterkopf und Du wirst in Kürze ein BNode-Meister sein!

117 - BNodes setzen

(LL) Dark

Der Editor versucht an jedem Raumportal und in der virtuellen Mitte des Rooms ein Bnode zu setzten und diese dann miteinander zu verbinden.
In einem runden oder rechteckigen Raum ist das kein großes Problem. Sobald der Raum aber eine etwas ungewöhnlichere Form aufweist, oder Einbauten hat, scheitert der Editor regelmäßig. Außerdem ist es vielleicht auch nicht gewünscht an jedem Portal ein Bnode zu haben weil das Portal zB. Ein Fenster ist oder in einen Raum führt den das GuideBot gar nicht finden soll, oder, oder, oder....
In all diesen Fällen muss man per Hand eingreifen.

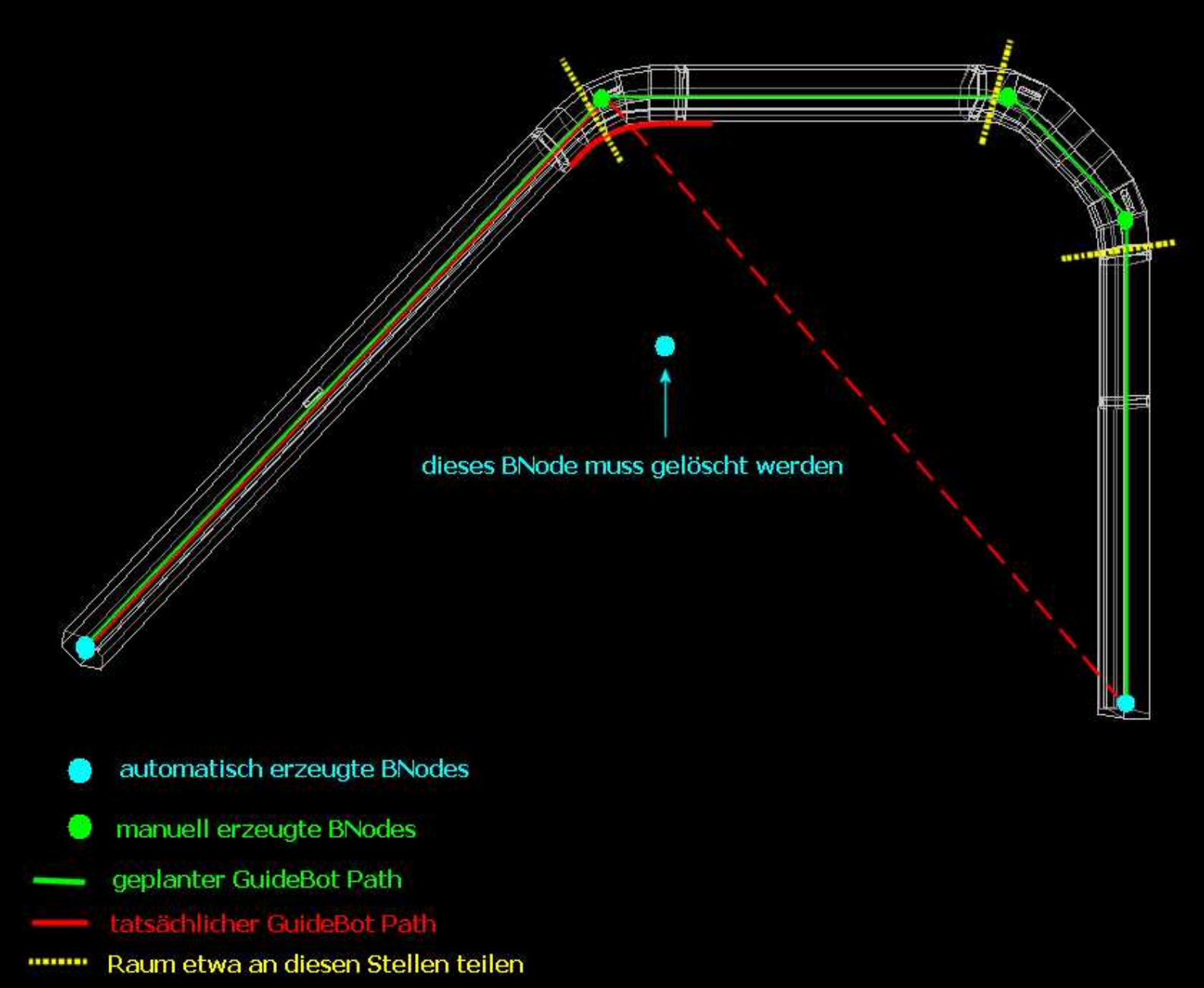
Wie das am besten geschieht soll hier erklärt werden.

Räume

Das Wichtigste zuerst:
Es ist von größter Wichtigkeit die Räume so zu bauen das sie keine Verwinkelungen, tiefere Ausbuchtungen oder größere Krümmungen aufweisen. Hat man solche Raumstrukturen, dann muss man diese in Einzelräume zerlegen die man dann wieder durch Portale miteinander verbindet.

Warum ?
Eigentlich sollte das GuideBot sich von Node zu Node durch den Raum hangeln. Das tut es im Idealfall auch. Es fliegt vom Node des Eingangsportals zum Node in der Mitte des Raumes und von da aus zum Node des Ausgangsportals. Von da geht es ab in den nächsten Raum. Dies ist, wie gesagt, der Idealfall – gibt es leider eher selten so.
In der Regel hat man mehrere Nodes im Raum verteilt und vielleicht auch mehr als nur zwei Portale. Dann tritt eine eher unangenehme Eigenschaft des GuideBot zu tage. Man bemerkt dass das GB ein faules Stück ist. Es versucht den kürzesten Weg zum nächstgelegenen Node zu finden und schert sich einen Dreck darum ob diese Nodes direkt miteinander durch Edges verbunden sind. (Edges sind die Verbindungen zwischen den Nodes) Es spielt für das GB auch keine Rolle ob sich eventuell eine Wand zwischen diesen Nodes befindet. Das Ergebnis ist dann, dass das GB stur vor einer Wand hängt und nicht weiter kommt.

Zur Veranschaulichung mal ein Bild:



Wie man sieht, wird ein BNode in der virtuellen Raummitte erzeugt. Den Editor interessiert es dabei nicht ob die Raummitte nun innerhalb oder außerhalb des eigentlichen Raumes liegt. Solche, außerhalb der Levelstruktur liegenden, BNodes müssen gelöscht werden. Als nächstes fügt man per Hand neue BNodes ein. Dies geschieht immer so, das es eine direkte Sichtverbindung zu einem anderen Bnode gibt. Im Bild sind diese BNodes grün dargestellt. An sich hätte man nun einen durchgehenden AI-Node-Path durch den ganzen Raum hergestellt und könnte zufrieden sein. Leider würde das GuideBot aber den rot eingezeichneten Path nehmen und somit an der rot gekennzeichneten Wand hängen bleiben.
Abhilfe schafft hier nur, den Raum ca. an den gelb gestrichelten Linien in 4 Einzelräume zu teilen und danach durch Portale wieder miteinander zu verbinden. Führt man dann die automatische AI-Node-Erstellung aus, wird kaum nachträgliche Handarbeit erforderlich sein und auch das GuideBot wird problemlos seinen Weg finden.

Merke:
Die Krümmung eines Ganges sollte nie größer als 45° sein.
Bei sehr breiten Gängen kann sie maximal 90° sein. (aber nicht empfehlenswert)
Tiefe Raumausbuchtungen vermeiden. Solche Ausbuchtungen besser als separaten Raum anfügen.

Portale

Standardmäßig versucht der Editor an jedem Portal ein BNode zu setzen. **Das setzen der BNodes ist dabei jedoch von der Textur des Portals abhängig!**
Auch wenn man bei einem Portal die Textur nicht sieht, so ist sie ja dennoch da. (wähle ein Portal an und schaue in die Textur Bar) Damit der Editor am Portal ein BNode setzt, muss dieses Portal die Textur *Palmleaf1* haben. (In Atans letzter Editorversion wird die Portaltextur automatisch dahingehend geändert) Hat ein Portal eine andere Textur, dann wird KEIN BNode gesetzt. Somit kann man ganz einfach darauf Einfluss nehmen ob an einem Portal ein BNode gesetzt wird oder nicht. Dies funktioniert auch nachträglich. Wird ein BNode gesetzt, dann immer in der Mitte des Portals.

Ausführung Schritt für Schritt

Level erstellen und sämtliche Objekte einfügen.

1. Alle Räume überprüfen ob sie die oben genannten Anforderungen erfüllen. Müsste nach Erstellung des AI-Node-Systems ein Raum gesplittet werden und/oder ein neuer Raum hinzukommen, würden alle BNodes des Levels wieder gelöscht werden und man müsste von vorne anfangen!
2. ALLE Portale überprüfen ob sie die oben beschriebenen Anforderungen erfüllen.
3. Bei älteren Editorversionen muss man darauf achten das alle Portale die Textur *Palmleaf1* haben (außer die, die keine BNodes bekommen sollen).
4. Bei der neuesten Editorversion ist es umgekehrt, hier muss man darauf achten das alle Portale die kein BNode bekommen sollen, diese Textur NICHT haben. (da die Portaltexturen beim erzeugen des Portals automatisch auf *Palmleaf1* gesetzt wurden)
5. Nun erst das AI-Node-System erzeugen!
   Diese geschieht folgendermaßen:

   a. Die *Path Bar* öffnen und dort AI Paths (BNodes) auswählen.
   b. Klicke nun auf Create AI Path Nodes.
   c. Es werden nun im gesamten Level in allen Räumen automatisch BNodes erzeugt und versucht diese miteinander durch Edges zu verbinden.Danach wird dir eine Liste angezeigt in der steht in welchen Räumen das nicht geklappt hat und welche BNodes dort nicht miteinander verbunden sind.
   d. In diesen Räumen musst du nun auf jeden Fall selbst tätig werden.
      Es empfiehlt sich aber DRINGEND auch ALLE anderen Räume zu kontrollieren.

Wähle den Raum aus an dem du arbeiten willst und gehe in die Room View.
Solltest du keine AI Paths oder BNodes sehen, dann klicke mit der **rechten** Maustaste ins Fenster und setze bei Display AI-Nodes ein Häkchen.

Wechsle mit **Ctrl+H** in den Path mode oder betätige dazu den Button
Nun kannst du mit **N** die Nodes einzeln durchschalten (mit **Shift+N** geht's rückwärts).
Du kannst sie natürlich auch in der *Path Bar* mit den Pfeil-Buttons unter BNode durchschalten.
Hier wird dir auch angezeigt welches Node gerade aktiv ist.
In der Room View wird das aktive Node vergrößert und in den Koordinatenfenstern gelb angezeigt.Schaue nun als erstes ob sich ein, oder mehrere Nodes, außerhalb des Raumes befinden.
Da sie manchmal schwer zu finden sind, schalte zu diesem Zweck einfach mal alle Nodes durch.
Hast du ein solches Node gefunden, dann lösche es, indem du in der *Path Bar* auf **Delete** klickst.
Lösche auch die Nodes, die zu dicht an einer Wand platziert wurden. (oder verschiebe sie)
Solche Nodes können meist nicht mit den anderen Nodes durch Edges verbunden werden.
Nachdem du alle derartigen Nodes entfernt hast, klicke in der Path Bar auf **Edge cRoom Nodes**.

Nun wird versucht alle restlichen Nodes des Raumes miteinander zu verbinden.
Wenn das klappt, dann ist dieser Raum fertig und du kannst dich dem nächsten zuwenden.
Wenn es nicht klappt, oder du mit der Anordnung der Nodes noch nicht zufrieden bist, dann ist es wahrscheinlich notwendig weitere Nodes einzufügen.

Setze dazu im XY oder ZY Koordinatenfenster die gestrichelte grüne Nulllinie mit **Ctrl+linke Maus** auf die Raumhöhe auf der das Node später stehen soll.
Dann setze im XZ Koordinatenfenster (Draufsicht) per Mausklick das rote X an die Stelle wo das Node eingefügt werden soll und drücke dann auf der Tastatur **Insert** oder klicke in der *Path Bar* auf **Insert**.
Nun erscheint möglicherweise eine Abfrage ob das neue Node in die aktuelle Edge (die ist gelb) integriert werden soll oder nicht. Wenn du nicht gerade eine Edge trickreich umleiten willst, dann kannst du diese Abfrage immer mit Nein beantworten.
Füge deine neuen Nodes nun so ein daß sie immer in Sichtverbindung mit einem anderen Node stehen. Achte auch darauf daß sie nicht zu dicht an einem Face platziert sind (5 Units Abstand reichen normalerweise).
Die Nodes müssen so angeordnet sein das sich von Node zu Node eine durchgehende Verbindung von einem Portal zum anderen herstellen lässt.
Darüber hinaus können natürlich noch weitere Nodes im Raum angeordnet werden um Einbauten zu umgehen oder um kleinere Einbuchtungen zu erreichen.
Die Nodes lassen sich auch verschieben indem man sie in einem Koordinatenfenster anwählt und dann mit **Numblock 8/6/4/2** bewegt.Sie lassen sich nicht durch Faces hindurch schieben.
Wenn alle Nodes deiner Meinung nach richtig platziert sind, dann betätige wieder *Edge cRoom Nodes*.

Wiederhole das obenstehende so lange, bis du mit dem Ergebnis zufrieden bist und der Raum nicht mehr als fehlerhaft aufgelistet wird.

Hinweis:
Falls dein Level ein Terrain besitzt und sich dort auch external Rooms befinden, werden in der Verifyliste wahrscheinlich BNode Errors für Räume angezeigt die es bei dir gar nicht gibt.
(zumindest wirst du sie nicht finden)
In Atans neueren Editorversionen steht dann dort *No path from x to y in terrain regions ....*
Diese Errormeldung kannst du getrost ignorieren.

Manchmal lassen sich Nodes an Portalen einfach nicht verbinden.
Verschiebe diese Nodes einfach um 2-3 Units vom Portal weg. (zur Not auch weiter)
Danach sollte es klappen.

Hast du Levelteile die durch Terrain von einender getrennt sind, dann achte darauf das eine direkte Sichtverbindung besteht. Ansonsten wird der GB den Weg nicht finden.
AI Paths im Terrain lassen sich mit der letzten mir bekannten Editorversion (V1.1 Beta9 Atan 0.3_3_N - no Public) leider noch nicht erzeugen. Diese Funktion ist aber bereits in Vorbereitung und wird vielleicht irgendwann einmal zu Verfügung stehen.
(mittlerweile geht das)

Viel Spaß beim Levelbauen
(LL) Dark

118 - Player-Respawnpunkte - toprog

xxx

überarbeitet

Pumo
DBB Ace
DBB Ace

Posted Thu Sep 15, 2005 7:19 pm View user's profile Reply with quote Send private message
Well, i have some time playing with D3 Edit and i've made some simplistic levels.

However, i have this doubt.
How can i add multiple Single-Player starts to work like in the Single Player Campaigns: When you die, you start in the nearest place from where yo died.

So, how can i do it? Confused

DCrazy
DBB Benefactor
DBB Benefactor

Posted Thu Sep 15, 2005 7:45 pm View user's profile Reply with quote Send private message
You add waypoint objects where you want the player to respawn after dying. They used to be on the Sound bar; OtherOne moved them to some other toolbar -- object, I believe -- in one of the "beta" builds. Once the player passes a waypoint, they respawn at that waypoint if they die.

___

How do I script a checkpoint, you know, where you respawn after you die. What should I do?

Any solutions?

DCrazy
DBB Benefactor
DBB Benefactor

Posted Sat Jul 03, 2004 4:53 am View user's profile Reply with quote Send private message
For future reference, D3Edit topics go in Descent Development, not the Coders' Corner.

And there's no scripting required for waypoints. Just add one to the level (they're on the Object bar I believe, they used to be on the Sound bar a while ago). When the player runs over the waypoint, D3 will remember. When the player dies, he will respawn at the last waypoint he touched.

Sapphire Wolf
DBB Captain
DBB Captain

Posted Sat Jul 03, 2004 8:27 am View user's profile Reply with quote Send private message
thank you

DarkFlameWolf
DBB Captain
DBB Captain

Posted Sun Jul 04, 2004 9:04 pm View user's profile Reply with quote Send private message
there's gotta be more specific than that. Is the waypoint like a portal or an object that the player must get NEAR to activate?

Sapphire Wolf
DBB Captain
DBB Captain

Posted Sun Jul 04, 2004 9:36 pm View user's profile Reply with quote Send private message
DCrazy wrote:
For future reference, D3Edit topics go in Descent Development, not the Coders' Corner.

And there's no scripting required for waypoints. Just add one to the level (they're on the Object bar I believe, they used to be on the Sound bar a while ago). When the player runs over the waypoint, D3 will remember. When the player dies, he will respawn at the last waypoint he touched.

Actually, they're on the sound bar. I Think

DarkFlameWolf wrote:
there's gotta be more specific than that. Is the waypoint like a portal or an object that the player must get NEAR to activate?

Hmmmmm... Good Question.

I think Gameedit.com should include a description about waypoint.

Arbitar
DBB Ace
DBB Ace

Posted Sun Jul 04, 2004 11:44 pm View user's profile Reply with quote Send private message
you heard the man Skyalmian Wink

Interceptor6
DBB Ace
DBB Ace

Posted Mon Jul 05, 2004 5:29 pm View user's profile Reply with quote Send private message
You only need to go in the room with the waypoint to activate it. It is an object, placed in the level. It is on the sound bar (for some unknown reason).

DCrazy
DBB Benefactor
DBB Benefactor

Posted Tue Jul 06, 2004 11:51 am View user's profile Reply with quote Send private message
Are you guys running the absolute latest version of D3Edit? I could have sworn that at some point OtherOne moved the Waypoint button to the Object bar.

OG3 used waypoints as well. If you died in the middle of a race, you respawned at the last waypoint you touched. Unfortunately, collisions with waypoints are never reported to the mod, so we had to use "special" room flags (S1, S2, etc.) -- we called them checkpoints -- to determine where players

were, and waypoints to respawn them at certain locations. This limited the number of checkpoints in a level to 5 + the starting line (room S1).

[edit]You only need to go into the room with the waypoint in it? No wonder collisions with waypoints are never noted, because they never happen...[/edit]

119 - Loadscreen einbinden

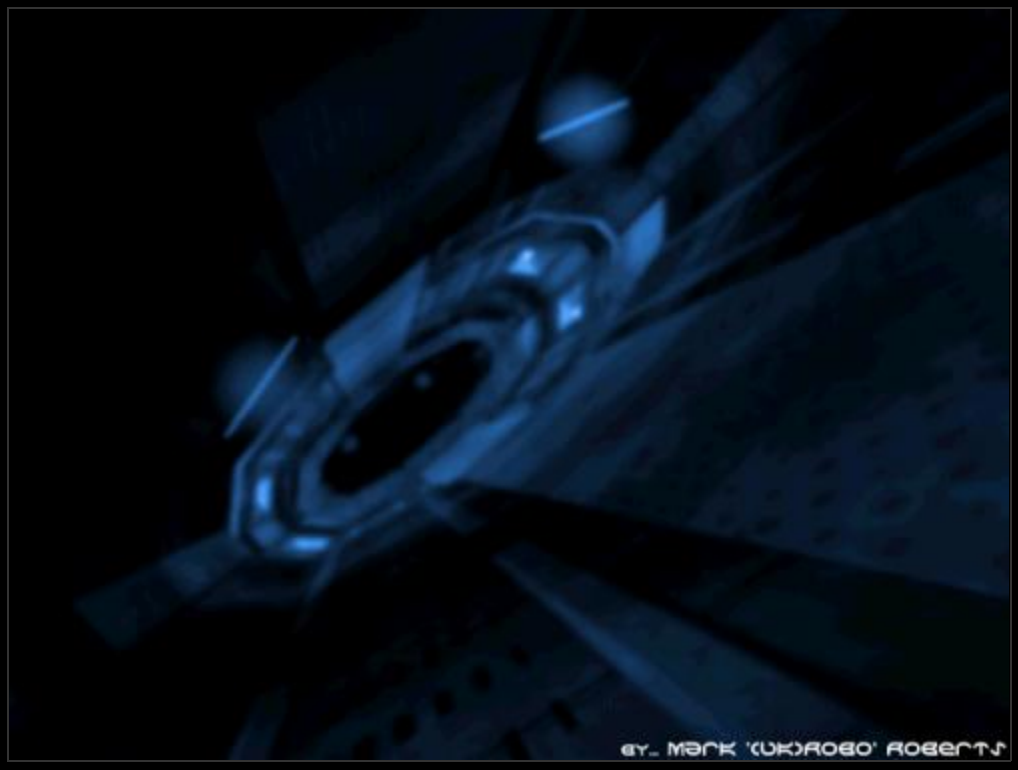
Robo

überarbeitet

Wenn Du einen Level auswählst beginnt er zu laden. Du wirst vielleicht irgendein Bild sehen wenn das passiert, das ist ein Loadscreen. Normalerweise setzen die mich in eine Trance, was nur zeigt wie leicht ich zu beeindrucken bin ☺

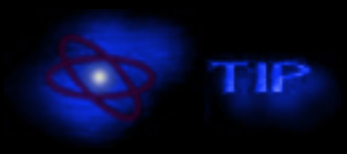
Einen Loadscreen zu erzeugen ist für mich einer der erfreulichsten Teile an der Levelkreation. Ich mag es kreativ zu sein, nette Farbschattierungen und koole Effekte einzubauen. Und ich inkludiere immer einen Namen. Schau Dir diesen Loadscreen an:

Da bin ich ☺ Dieser Loadscreen ist das Resultat aus 30 verschiedenen Dateien, das sind eine Menge Farben. Der Level sieht nicht so aus, natürlich, es ist nur ein wenig Kunst. Was Du **nicht** in diesem Bild sehen kannst, ist was Descent 3 damit macht. Ich habe einen Gaußschen Weichzeichner angewendet. Am äußeren Rand sieht es wie ein einfacher und normaler Weichzeichner aus, aber in D3 werden die Farben in einem netten Effekt aufgesplittet.



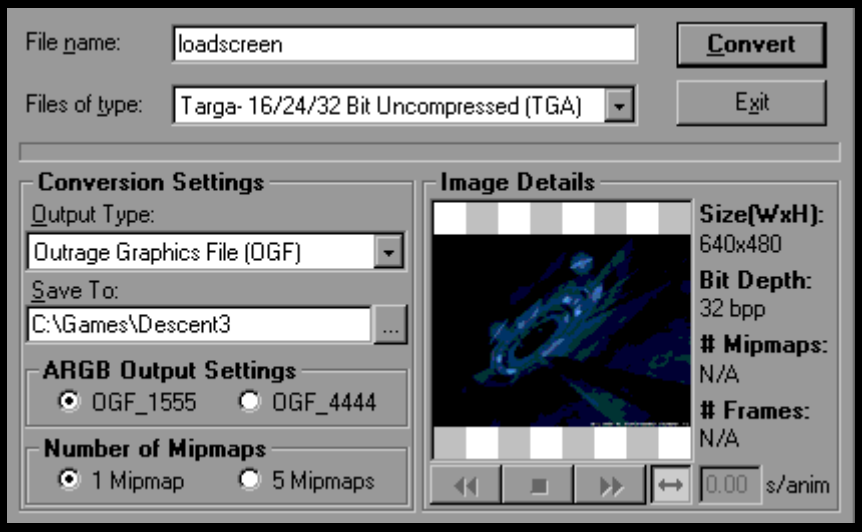
Das ist eine Art 'Outrage-Style'-Loadscreen , weil alle Outrage-Levels diesen Effekt in ihren Loadscreens haben.

Ich arbeite in Paint Shop Pro, da ich Photoshop nicht habe, aber alles was einen Gaußschen Weichzeichner machen kann geht. Dein Bild muß eine 640x480er Bitmap sein. Hast Du den Prozeß des kreativen Schaffens abgeschlossen, wende Den Gaußschen Weichzeichner an.



Du hast Glück, wenn Du die Stärke des Weichzeichners verändern kannst. Ein Bild könnte in D3 mit einer zu hohen/zu niedrigen Einstellung nicht richtig aussehen. Stelle die Werte höher wenn das Bild helle Farben hat, kleiner wenn es ein dunkles ist.

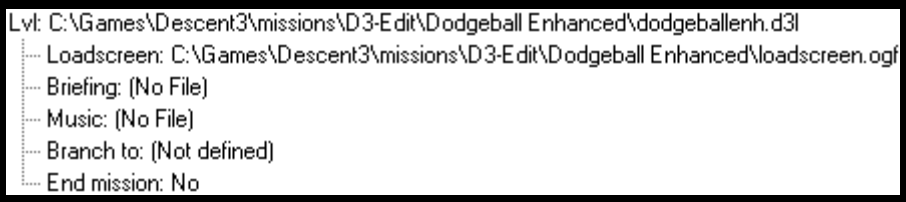
Dann füge Deinen Namen ein. Wir machen das jetzt, da wir nicht wollen daß er weichgezeichnet wird. Wenn Du fertig bist, speichere das Bild als **unkomprimiertes 24-Bit .tga**. Nun ans Konvertieren.



Descent 3 selbst kann keine `.tga`-Datein für die Loadscreens verwenden, Du mußt sie ins `.ogf`-Format konvertieren. Dazu kannst Du das OGF-Tool (mit D3Edit inkludiert) verwenden oder das fantastische Programm namens D3 Image Tool (siehe Tools-Sektion).

Finde Deine Datei und Stelle die Optionen so wie im Bild links ein, stelle Dein eigenes Output-Verzeichnis ein.

bschließend müssen wir spezifizieren wo Dein Loadscreen in der `mn3`-Datei ist. Verwende den Level Properties-Dialog im MN3-Editor, es sollte dann etwa so aussehen:



Wenn Du den Quicktest verwendest, inkludiere Deinen Loadscreen erst mal in die Dateiliste. Dann doppelklicke im unteren Bereich von Quicktest unter Level Setup/Ordering: auf Loading: und wähle Deinen Loadscreen aus.

Speichere Dein `mn3`... und teste es!

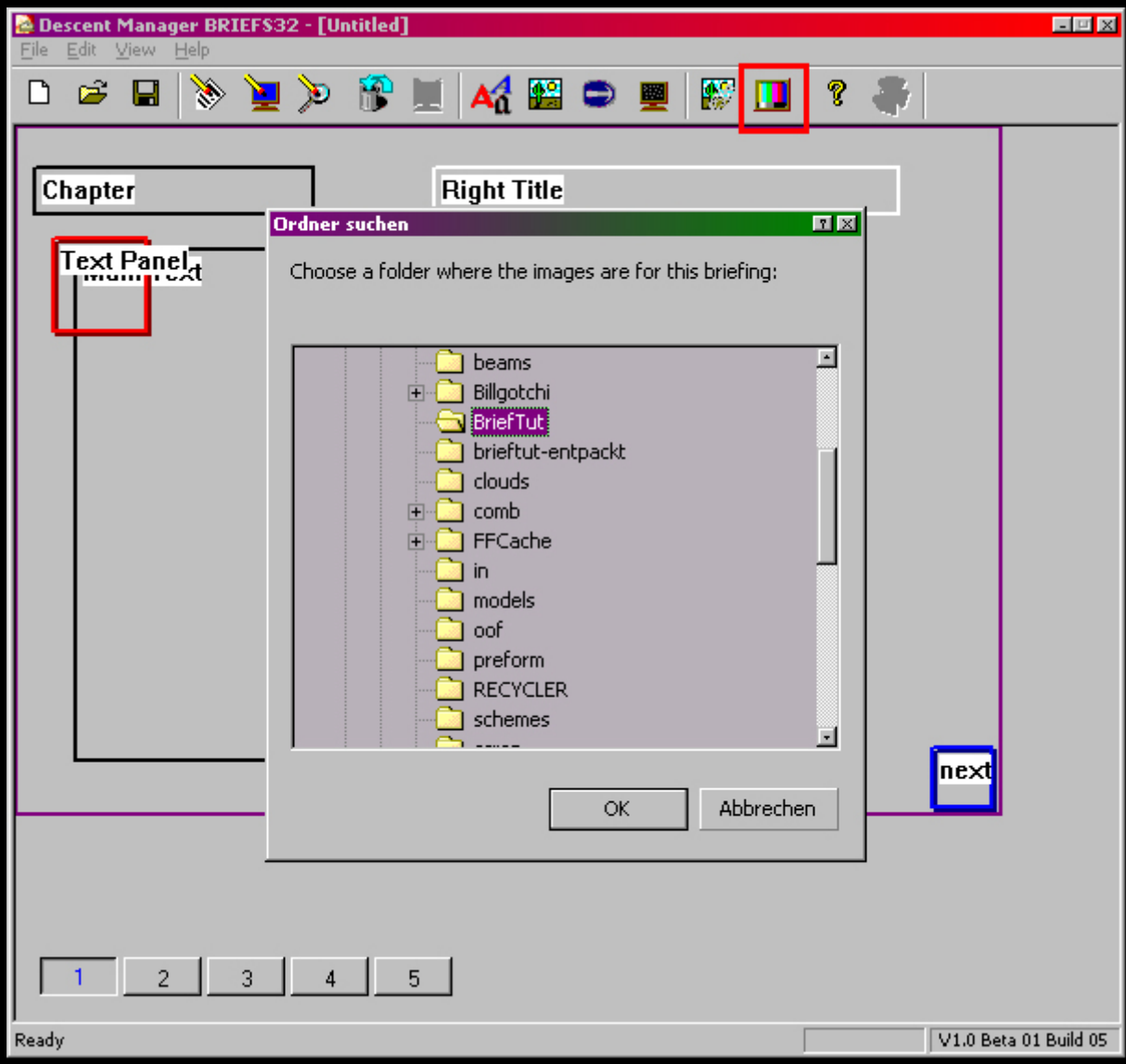
120 - Levelbriefing

RagilRal

Ein Singleplayer-Level kann mit einem Levelbriefing nochmals aufgewertetet werden. Hier könnt Ihr dem Spieler zusätzliche Atmosphäre verschaffen, indem Ihr Ihm einen Kontext präsentiert, in dem die Mission spielt.
Levelbriefiings könnt ihr mit dem Tool Briefs32 erstellen, welches Teil des Descent Manager ist aber dennoch einzeln installiert werden kann; siehe Sektion 'Liste aller D3 Tools'.

Briefs32 einrichten

Installiert es Euch und entpackt anschließend die `brieftut.exe`, entweder mit Winrar oder indem Ihr sie ausführt.
Startet Briefs32, und öffnet `tutorialsample.brf` aus der entpackten `.rar`: Ihr werdet gefragt, wo die Bilder zum Briefing liegen. Dies muß man leider jedesmal machen, egal ob man ein Briefing öffnet oder neu erstellt.



Habt ihr das eingestellt, ändert sich in der Ansicht zunächst mal nix. Ihr müßt noch den rot eingerahmten Button klicken, dann wird alles dargestellt.

Die nummerierten Buttons im unteren Bereich des Fensters repräsentieren die einzelnen Bildschirme des Briefings.

Bestätigt die Dialogfelder besser immer per Mausklick auf den OK-Button, die Enter-Taste wird nicht immer genommen ☺

Die Praxis

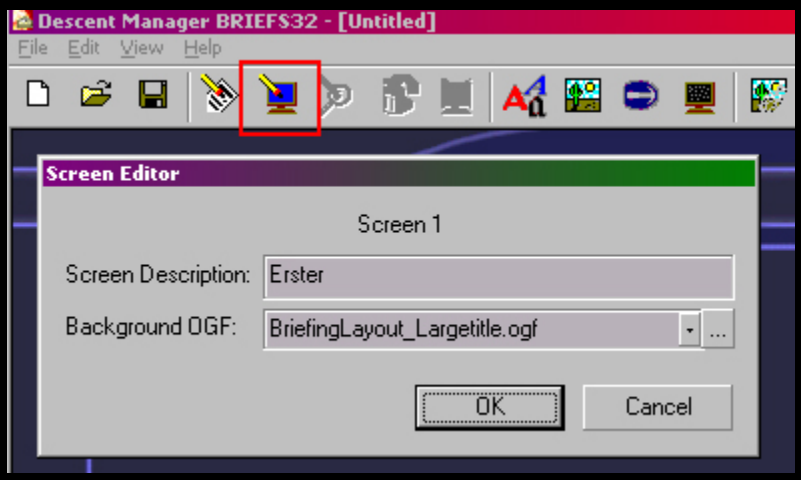
Wenn Ihr mit der Maus über einen Button verharrt, bekommt ihr eine kurze Beschreibung angezeigt; das meiste ist so eigentlich recht selbsterklärend.
Aber, wer hätte das gedacht, soeinfach ists dann auch wieder nicht ☺ wer die Grafiken für Briefings nicht komplett selbst erstellen will, der muß aus der `d3.hog` zunächst mal die Grafiken extrahieren, die benötigt werden. An diesem Punkt werden die meisten von Euch bereits irgendwo einen Ordner mit den `ogf`'s aus der `d3.hog` haben, und hier ergibt sich bereits die nächste Schwierigkeit: man kann leider nur ein Verzeichnis für die Bilder angeben. Das bedeutet, entweder Du schiebst Deine Bilder in dieses Verzeichnis, oder Du holst Dir die `ogf`'s für Buttons, Hintergründe und Rahmen in jenes Verzeichnis, wo Du Dein Briefing zusammenstellst. Ich habe eine Liste der Original-`ogf`-Dateien zusammengestellt, welche für ein Briefing relevant sind. Es ist dies die 'Infrastruktur' eines Briefings, also die oben angesprochenen Rahmen, Buttons usw. Ich hoffe, sie ist vollständig ☺

```
Back.ogf
BackFocus.ogf
Backglow.ogf
BackGlowFocus.ogf
bluepanel.ogf
bluepanel2.ogf
bluepanel3.ogf
botarrow.ogf
botarrowglow.ogf
botarrowglowleft.ogf
botarrowleft.ogf
BriefingLayout.ogf
BriefingLayout_GiantTitle.ogf
BriefingLayout_GiantTitle2.ogf
briefinglayout_largealpha.ogf
briefinglayout_largetitle.ogf
Briefsideswoosh.ogf
Forward.ogf
ForwardFocus.ogf
ForwardGlow.ogf
ForwardGlowFocus.ogf
Quit.ogf
QuitFocus.ogf
Quitglow.ogf
QuitglowFocus.ogf
recon.ogf
reconglow.ogf
robots.ogf
robotsglow.ogf
Secobjlayout.ogf
```

Die zu Briefs32 beigelege Beispieldatei verwendet ausschließlich eigen-erstellte Bilder.

Die verwendeten Grafiken

Diese müssen als `ogf` vorliegen, und Ihr erstellt sie aus `tga`'s mit einem Alpha-Kanal, wo ihr - speziell bei Buttons - Abrundungen oder Ähnliches transparent macht, sodaß man den Hintergrund nicht an der Grafikgrenze abgeschnitten bekommt. Man kann aber auch 'normale' ogf-Bilder verwenden.
Ihr habt 530 mal 370 Pixel Platz um eure Briefings zu gestalten, dies deswegen, da das Briefing ja in die Oberfläche von D3 eingebettet wird und der Rahmen auch seinen Platz benötigt.

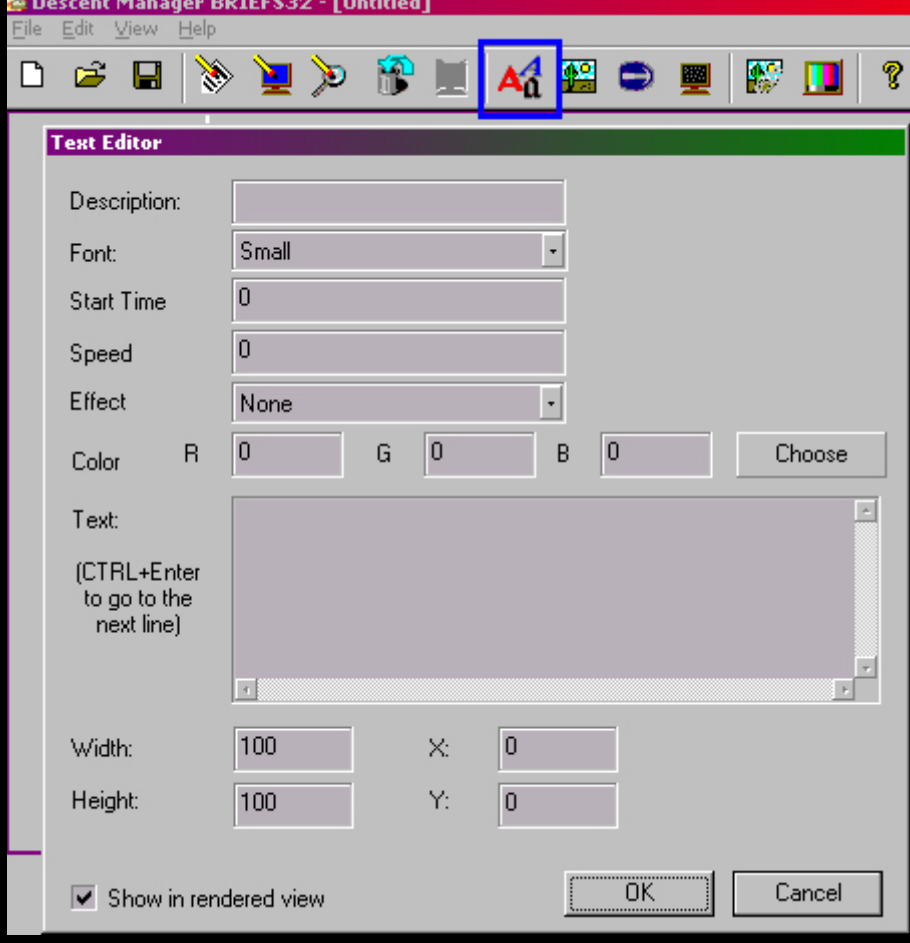


Hintergrund setzen

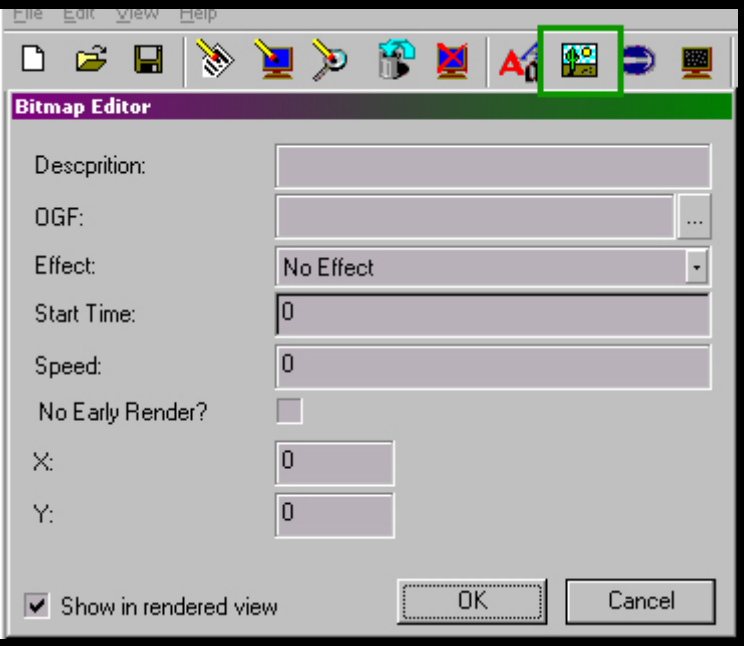
Zunächst mal benötigt Ihr einen Hintergrund, in dem Ihr das Briefing reinsetzt; ihr stellt diesen ein, indem ihr auf den rot eingerahmten Button klickt und im daraufhin erscheinenden Dialogfeld einen Titel angebt als auch eine Hintergrundgrafik.

Texte einsetzen

Durch Klick auf den blau umrandeten Button erzeugt Ihr ein Textfeld. Hier könnt Ihr neben dem eigentlichen Text noch ein paar Dinge einstellen, wichtig ist Breite/Höhe. Diese kann nur hier gesetzt werden, im Gegensatz zur Position, welche auch per Drag&Drop festlegbar ist.
Neben der Formatierung des Textes legt Ihr hier auch fest
(einzelne Parameter aufschlüsseln)



Grafik einbauen



Ihr klickt den grün eingerahmten Button und bekommt das folgende Dialogfeld:

unter OGF: müßt ihr allerdings nicht nur über den Button mit den drei Punkten die Datei anwählen, sondern auch noch per Hand den Dateinamen eures Bildes eintragen - sonst funktionierts nicht!

Beim laden der Grafik wird der Rahmen auf die Größe des Bildes angepaßt.

Neue Seite

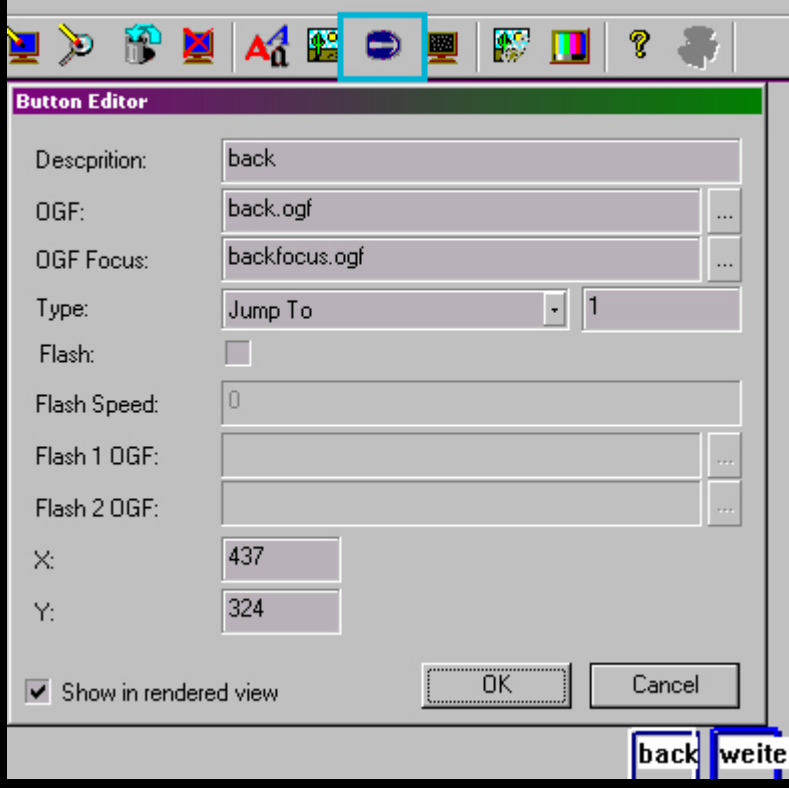
Seid Ihr mit dem Layout der ersten Briefing-Seite zufrieden, fügt ihr mit dem rechts gezeigten Button einen neuen 'Screen' an, und wieder lauern Fallstricke.... Ihr solltet Euch klar sein was Ihr wollt, denn man kann in Briefs32 nur an die letzte Seite anfügen! Löschen allerdings läßt sich natürlich jede Seite (Button links), ihr Inhalt ist dann allerdings futsch; wenn man also draufkommt daß man zwischen Screen Nummer 3 und 4 doch noch eine Seite haben will, sind die Seiten 4 und folgende verloren. Daher, erst mal hinsetzen und austüfteln wie das Ganze eigentlich ausschauen soll.

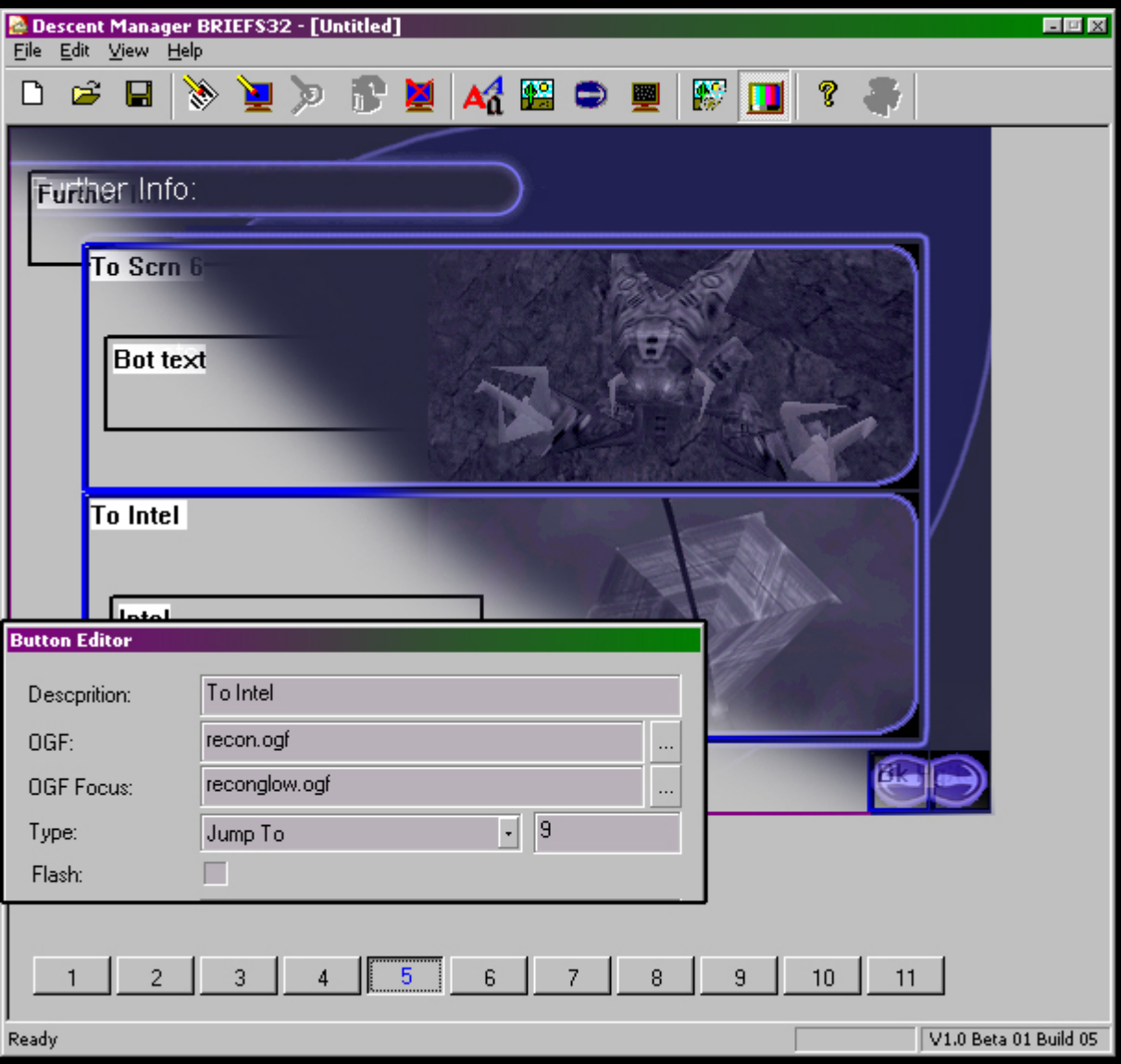
Navigation

Im Briefing muß man sich auch bewegen können, zu Diesem Behufe erstellt man Buttons. Klickt ihr den türkis eingerahmten Knopf...

(Parameter erklären)

Unter Type gebt Ihr an, wohin der Button führt. Eine 'Zurück'-Funktion müßt Ihr mit Jump To realisieren, ihr verweist dann einfach auf die vorige Seite.



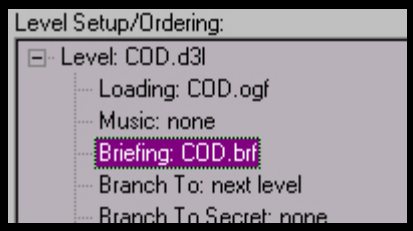


Die Missions-relevanten Informationen in den Originalen Briefings sind über Buttons aufstrukturiert, sodaß der Spieler gezielt etwas anwählen kann.

Wenn das Briefing dann steht, fehlt nurmehr noch eines:

Briefing ins Level einbauen

Geht denkbar einfach. Zunächst müßt ihr im Quicktest mal die Dateien des Briefings in die Dateiliste aufnehmen, und zwar die `.brf` inklusive aller zugehörigen Grafiken so diese nicht aus dem Originalfundus stammen. danach öffnet im unteren Bereich des Quicktesters den Zweig für euer Level, rechtsklickt auf Briefing: none, wählt Edit und wählt dann euer Briefing aus. Habt Ihr die `.brf` nicht in der Dateiliste, könnt ihr hier nix einstellen!



Tja, im Großen und Ganzen wars das auch schon.
Die Kunst ist es jetzt das zur Verfügung stehende auch zu nutzen, also ran an den Speck

Abschnitt L – Meisterklasse

Wer bis hierher gekommen ist, wird die folgenden Infos zu schätzen wissen

121	**Türen optimieren**	Vom Umgang mit defekten Türen	Ragil Ral	421
122	**Dynamisches Licht**	Kaputtbare Lichter!	Ragil Ral	425
123	**Gruppendateien**	CopyPaste im großen Stil	Ragil Ral	428
124	**Customs, nochmal**	...aber endgültig	Ragil Ral	432
125	**Bauen mit Objekten**	Damit geht's gleich leichter	Ragil Ral	433
126	**Mercenary-Bots verwenden**		Ragil Ral	438
127	**Schiffswahl in Singleplayer**	toprog	Ragil Ral	441
128	**Scripting & Matcen**	Auf Ereignisse von Matcenters per Skrip reagieren toprog	Ragil Ral	445
129	**Räume teilweise skalieren**	Wenn nur ein Teil vergrößert werden soll	Ragil Ral	450
130	**Zusammengesetzte Objekte inprog**		Ragil Ral	452

Toc

121 - Türen optimieren

Ragil Ral

Ihr kennt mittlerweile das Problem, wenn man irgendwo eine Tür anbaut wird das Face wo sie angebaut wird total zerschnitten. Es werden zwar minimal neue Faces erzeugt, aber unnötig viele Verts. Desweiteren ist es nicht ganz einfach einen Türraum ordentlich zu texturieren, da das Türobjekt einem meist die Sicht versperrt.
Ganz zu schweigen davon daß der Einbau von Stock-Doors meist T-Joints und andere Nettigkeiten ins Level einführt... also was tun?

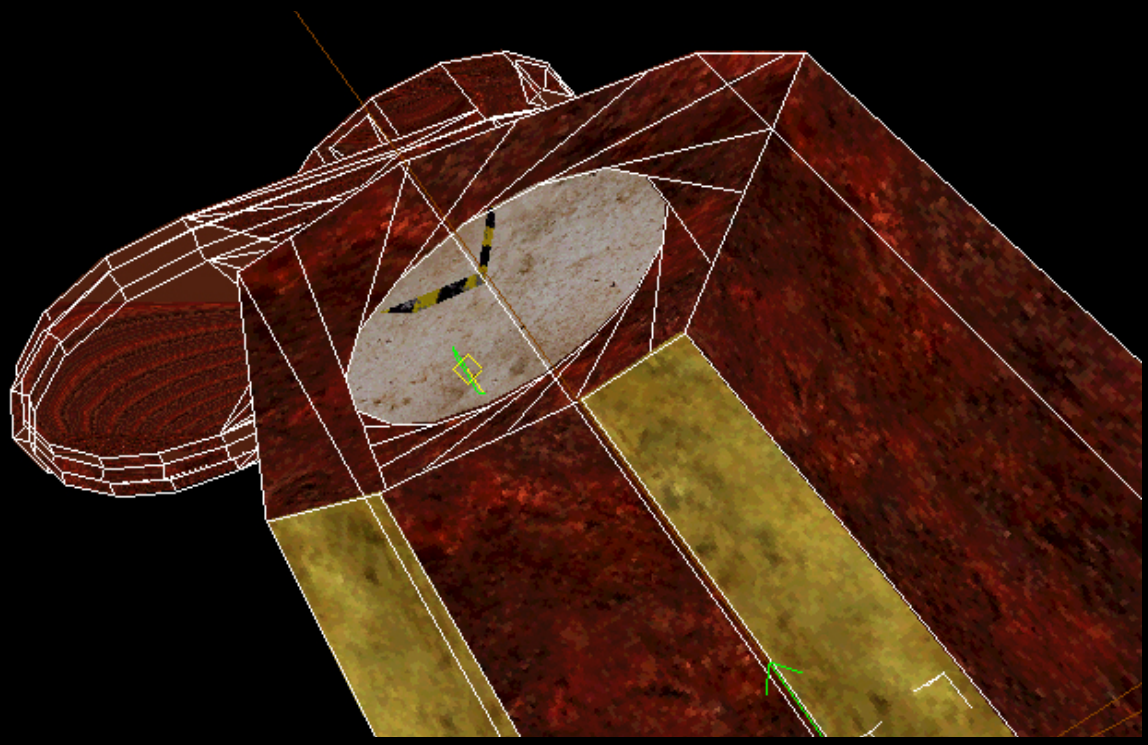
Vorher Nachdenken

...ist natürlich immer hilfreich ☺ Wenn ihr irgendwo im Level eine Tür wollt richtet die Geometrie bzw. die Architektur entsprechend darauf aus. Sirians Methode, Faces an denen Türen angepappt werden, derart einzufassen daß die Zerschneidung beim Einfügen die Umgebung nicht zu sehr beeinflußt, ist immer eine gute Idee, bedingt jedoch daß man eben vorher schon weiß wo man eine Tür einbauen wird. Aber oftmals ist ja Planung nur der Ersatz des Zufalls durch den Irrtum...

Im Nachhinein

Es kann aber natürlich immer vorkommen, daß man erst im Verlauf des Levelbauens draufkommt daß man irgendwo doch noch eine Tür braucht, sei es weil man's beim Planen vergessen hat oder weil man improvisierend baut. Es gibt zwei Möglichkeiten:
-Dank der Face-Schnittfunktion könnt Ihr ein Polygon wo eine Tür hinsoll vor dem Einfügen derselben derart zerschneiden, daß die Zersplitterung beim Attachen der Tür die Umgebung nicht komplett umbaut.
-Habt Ihr die Tür einfach angepappt müßt Ihr die Faces welche sie umgeben optimieren, quasi zusammenräumen; es hängt von der Tür ab wie man das lösen kann. Je mehr Verts das Frontface einer eingebauten Tür hat, desto mehr 'Splitter' ergeben sich beim Einfügen; denkt hier etwa an die Tür 'PTMC Covert H', welche acht Edges am FF hat, oder an die Secret Doors, welche recht rund sind und sogar 16 Edges haben. Ich werde diese Tür hier beispielhaft behandeln, da sie große Probleme beim Einbau verursacht.

Beginnt mit **Strg-N** und erzeugt ein Standard-Level mit Standard-Raum. Vergrößert diesen Standardraum ein wenig. Dann fügt eine 'seoulsecretdoorround' (es ist genau die letzte in der Liste) ein. Ihr bekommt dann sowas:

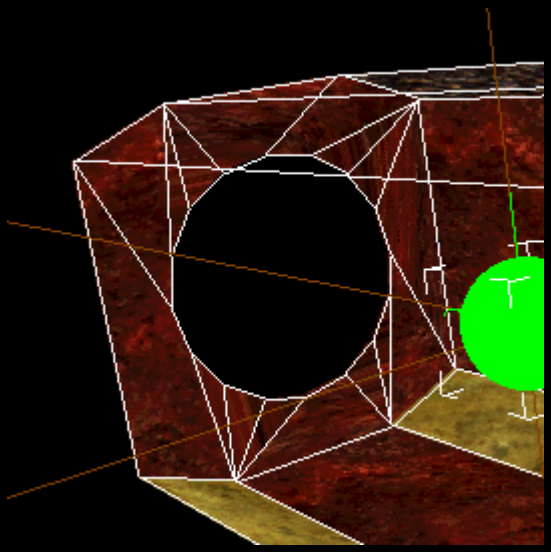


Das Einfügen der Tür hat das Face an dem sie angesetzt wurde komplett zerschnipselt. Dann führt Verify Mine aus: Ihr bekommt 16 Duplicate bzw. unused Verts und drei T-Joints, die Shell wird natürlich als 'bad' moniert. Die UV-Werte der Türraum-Texturierung sind *irgendwie*.

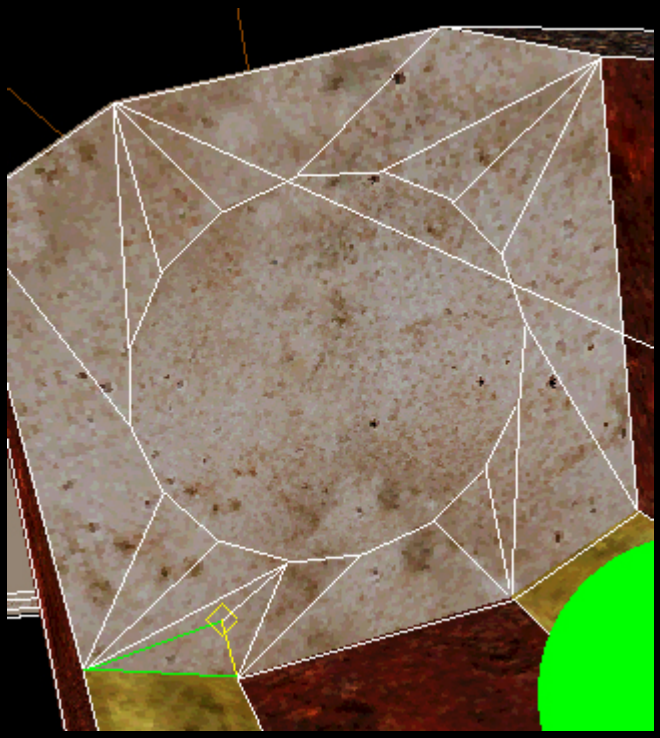
1.: Optimierung der Anbaustelle

Wieder habt Ihr hier zwei Möglichkeiten, ihr könnt entweder mit der Snap-Vert-Funktion die Verts zusammenfassen, was allerdings die Texturausrichtung zerstört, oder ihr splittet mit der Split Current Face-Funktion die Faces und faßt sie anschließend mit **Strg-Shift-LKlick** wieder (optimiert) zusammen, so bleibt die Texturausrichtung erhalten falls Ihr den Vor-Raum bereits fertig texturiert hattet. Letzteres ist aber leider nicht immer möglich, und außerdem langwieriger.

Verts zusammenfassen:
Laßt die Verts der Splitterfaces an Verts der angrenzenden Faces schnappen und macht Remove Extra Verts. Aber wie Ihr sehen könnt sind die UV-Werte nun ziemlich verbogen.



Faces splitten und reorganisieren:
Dies verlangt Euch ein wenig Hirnschmalz ab, erhält aber die Texturenausrichtung. Ihr müßt Euch überlegen von welcher Richtung aus Ihr beginnt. Nach getaner Arbeit führt den Remove Verts-Magneten aus.

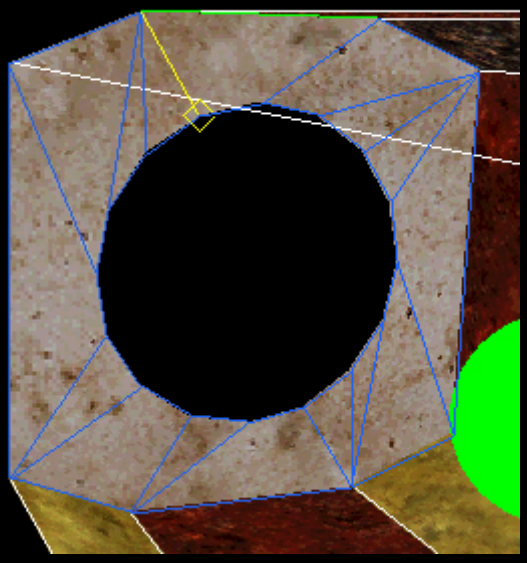


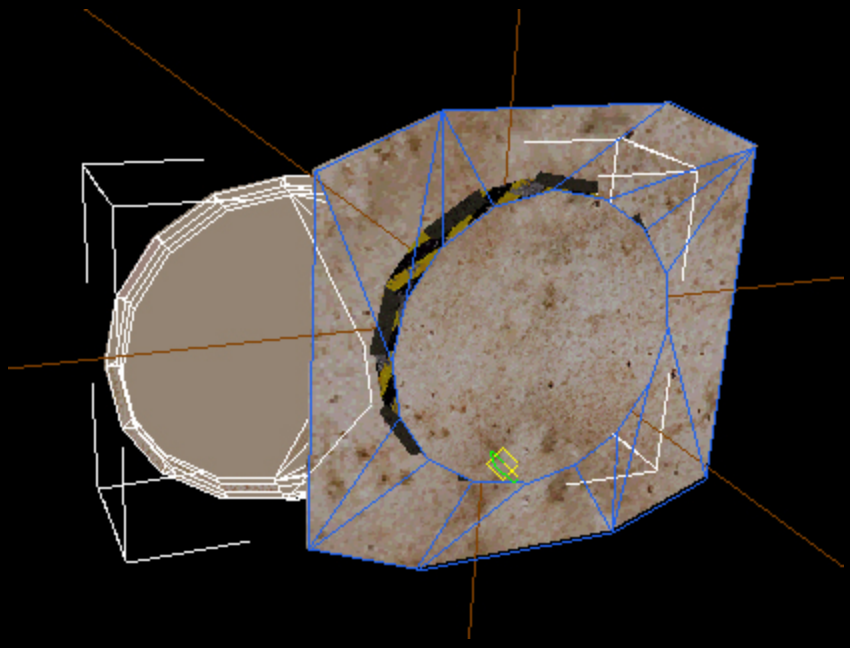
Wie Ihr sehen könnt, landet man mit der zweiten Methode hier an einem Toten Punkt; der Bereich um das current Face herum kann nicht mehr ohne Snap Verts weiter optimiert werden. Da jedoch die meisten Umgebungsfaces bereits optimiert sind, kann man hier ruhig einige Male mit Snap Verts arbeiten, da ja über **Strg-LKlick** die UV von den Nachbarn übernommen werden kann.

2.: Optimierung & Texturausrichtung des Türraums

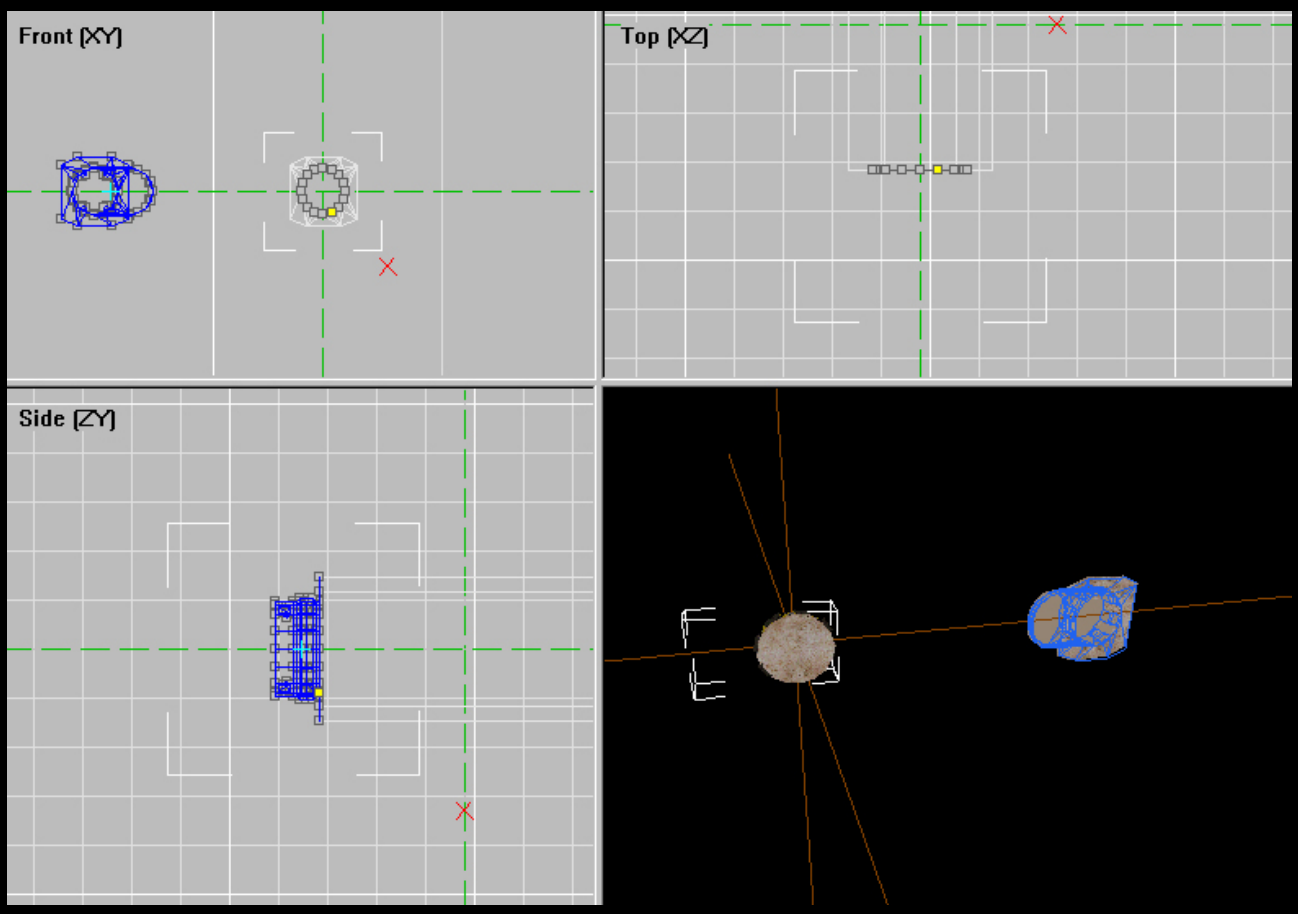
Zunächst würdet Ihr Euch in der World View mit **Strg-LKlick** die Ausrichtung eines Faces in den Türraum holen. Ihr müßtet hier - in der World View - derart hineinzoomen, daß Ihr Euch 'innerhalb' des Tür-Objektes befindet, um eins der angrenzenden Faces zu erreichen. Das finde ich aber unpraktisch, weswegen ich eine andere Methode favorisiere - Ihr macht ganz einfach folgendes:

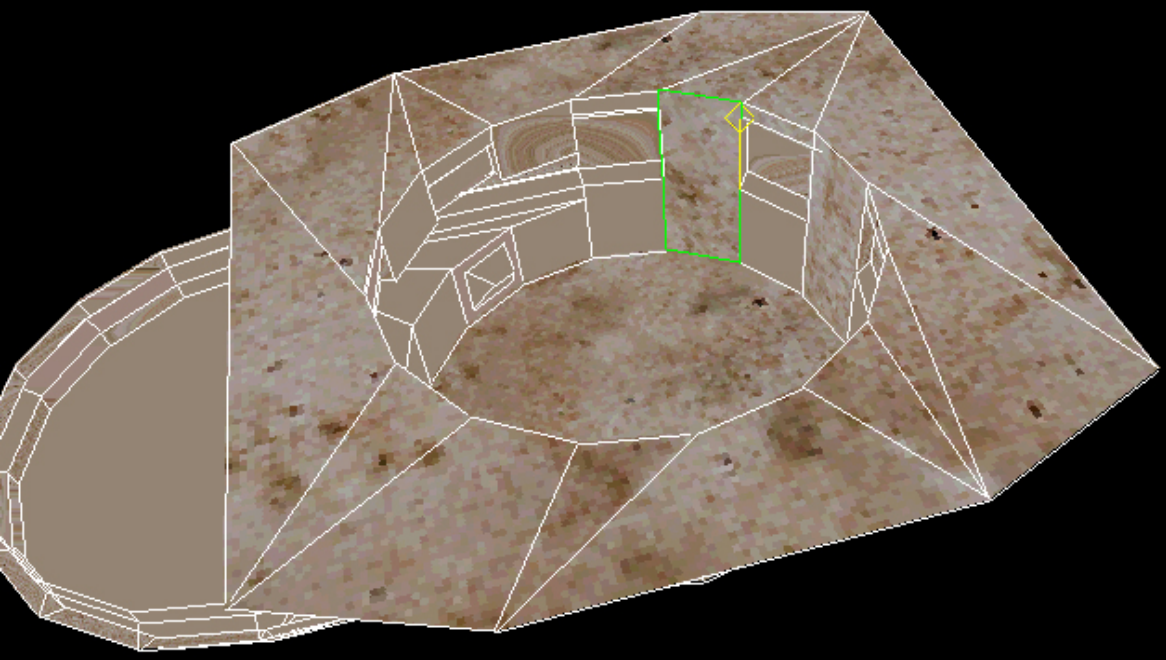
a) Geht in den Vorraum und kopiert die - ausgerichteten - Umgebungsfaces der Tür. Geht dann in den auszurichtenden Türraum und fügt dort die Faces mit **Strg-Shift-V** ein. Danach führt Remove Extra Verts aus.





b) Ihr habt nun die Faces mit der gewollten UV-Ausrichtung im Türraum, doch abermals ist das Tür-Objekt im Weg. Markiert nun alle Faces außer dem/der Portal/e, schneidet sie aus und fügt sie wieder mit **Strg-Shift-V** an Ort und Stelle ein. Danach geht in eine der drei Ortho-Ansichten, setzt das Grid mit 7 (In der Zahlenreihe im Alphablock) auf 100 Units und verschiebt die eben eingefügten Faces in irgendeine Richtung:



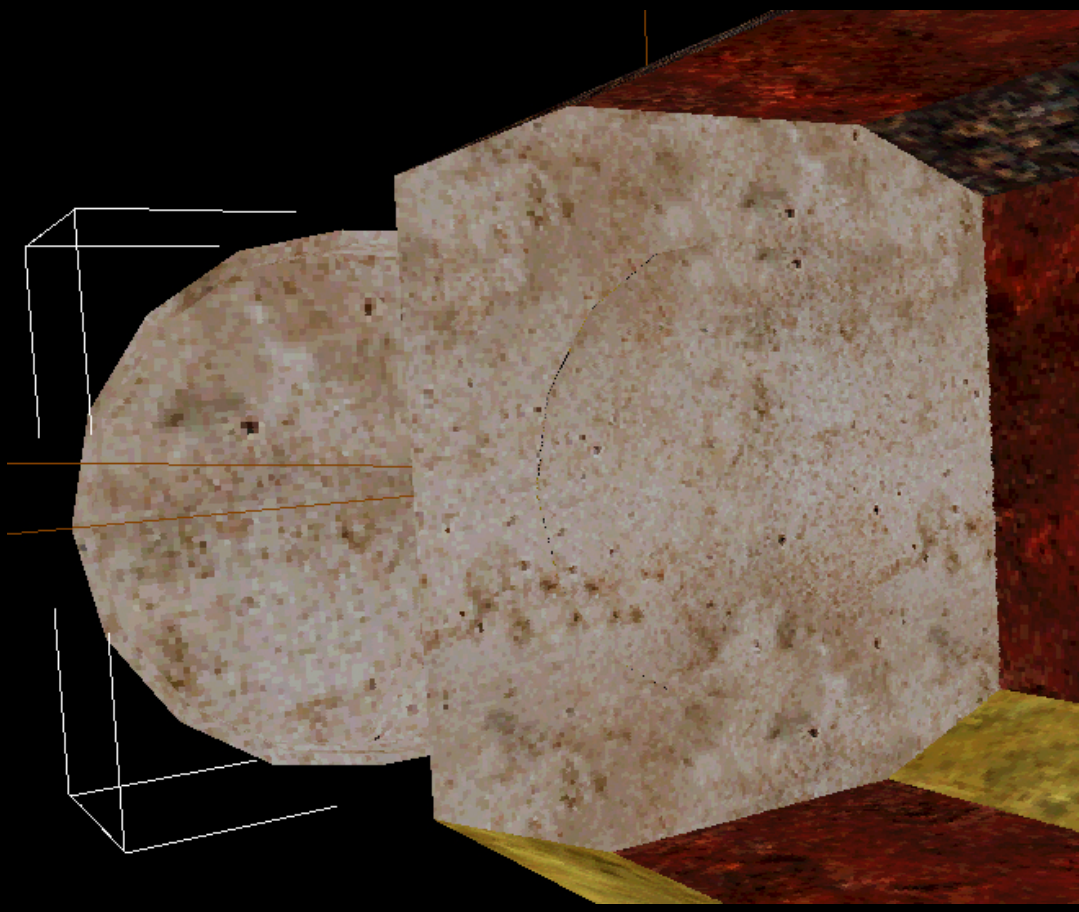


Nun könnt Ihr ganz entspannt die Ausrichtung der Texturen vornehmen.

c) Wenn Ihr damit fertig seid, löscht die Faces welche aus dem Vorraum stammen. Die Tür-Shell aus dem Beispiel hier muß aber noch repariert werden, sie enthält ja noch T-Joints. Außerdem ist eine Optimierung angesagt. Beseitigt also alle Unschönheiten.

d) Markiert danach die gesamte Face-Gruppe (vergeßt die Einbauten nicht, falls Ihr die mitverschoben habt) und schiebt sie wieder die 100 Units zurück. Danach noch einmal Remove Extra Verts ausführen. Verify sollte für den Türraum nun keine Fehler mehr melden.

Das wars auch schon.



Wenn Ihr eigene Türen baut, ist es eine gute Idee - besonders dann, wenn das Frontface viele Verts hat - einen Rahmen zu machen.

Wie manche von Euch eventuell bereits festgestellt haben, sind einige der 'mitgelieferten' Türen in D3, naja, ahäm, sagen wir mal nicht ganz fehlerfrei. Wenn man jetzt nicht jedesmal ein und dieselbe Tür fixen will, bei jedem Mal wenn man sie einbaut, {toprog}

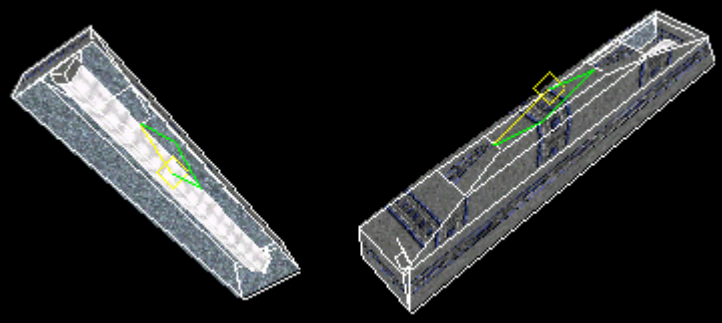
122 - Dynamisches Licht

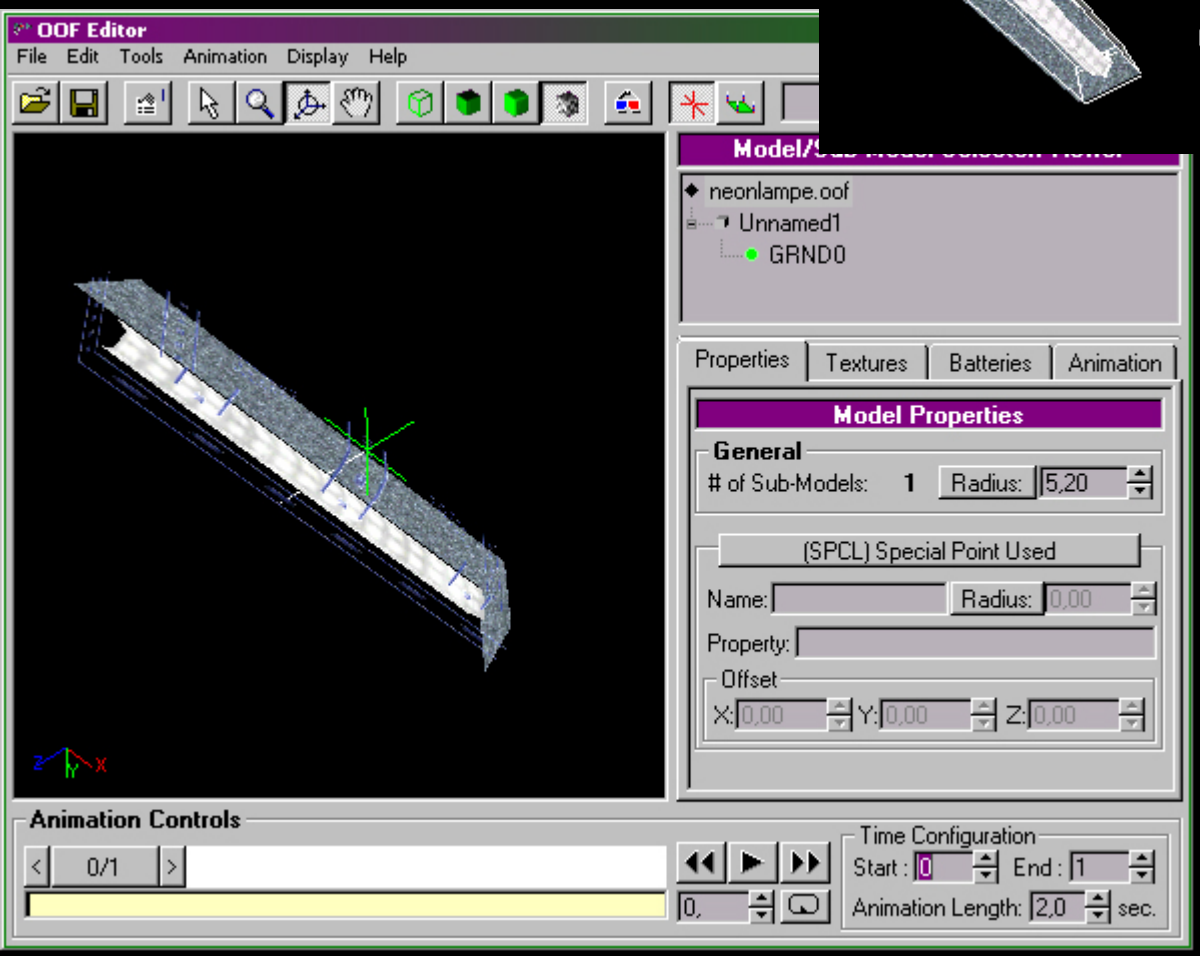
Ragil Ral

Innerhalb der Minen lassen sich ja aufregende Licht-und-Schatten-Spiele veranstalten; doch die Außenwelt wird nur von den Satelliten beleuchtet. Wenn man versucht am Terrain irgendwo Lichter hinzubauen, strahlen diese kein Licht ab und werfen auch keinen Schatten.

Es gibt aber Abhilfe!

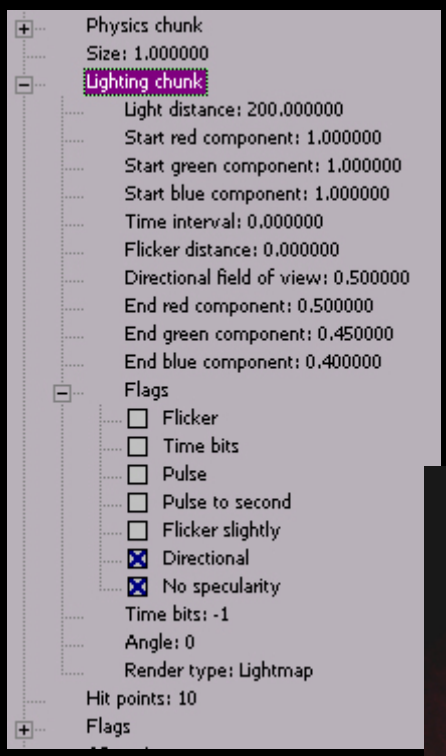
Descent 3 erlaubt es nämlich Objekten Licht abzustrahlen. Dies wird in der Level-`.gam` festgelegt; Ihr braucht also ein Objekt das leuchten soll und eine Level-`gam` um die Lichtverhältnisse des Objektes festzulegen.
Erstellt Euch also zunächst mal ein Licht-Objekt; baut es in D3Edit, setzt ein Face ein daß dann zum GRND werden soll (im rechten Bild das current Face), und wandelt es mit dem OOF-Editor in ein Objekt um.





Nun wirds ein wenig komplizierter.

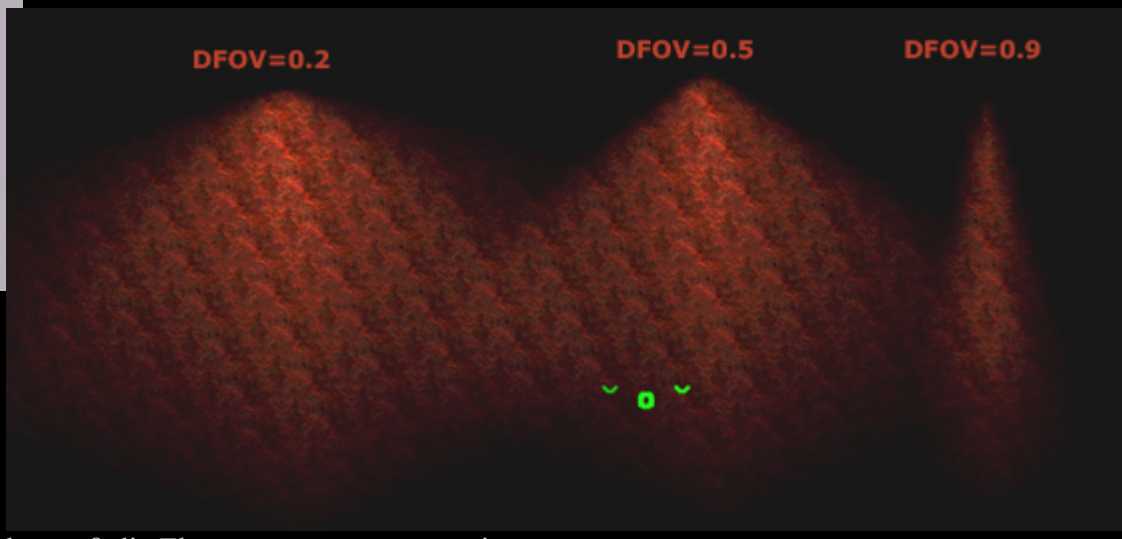
Jedes Objekt hat in seinem `.gam`-Eintrag einen 'Lighting Chunk', der die Lichtverhältnisse festlegt. Dort werden alle relevanten Werte angegeben, wie Reichweite des Lichtes, Farbe und ob es in eine bestimmte Richtung strahlt. Der Ursprung des Lichtes ist jener Punkt, an dem im D3Edit die Achsen entspringen. Die wichtigsten Parameter sind im einzelnen:



Light distance: Reichweite des Lichtes in Units.

Start red/green/blue component: Lichtfarbe am Ursprung; in Anteilen, zwischen 0 und 1.

Directional field of view: Wenn das Licht eine Richtung haben soll, wird hier jener Bereich angegeben
{toprog - herausfinden was genau hier angegeben wird. papacat spricht ja von cosin.}

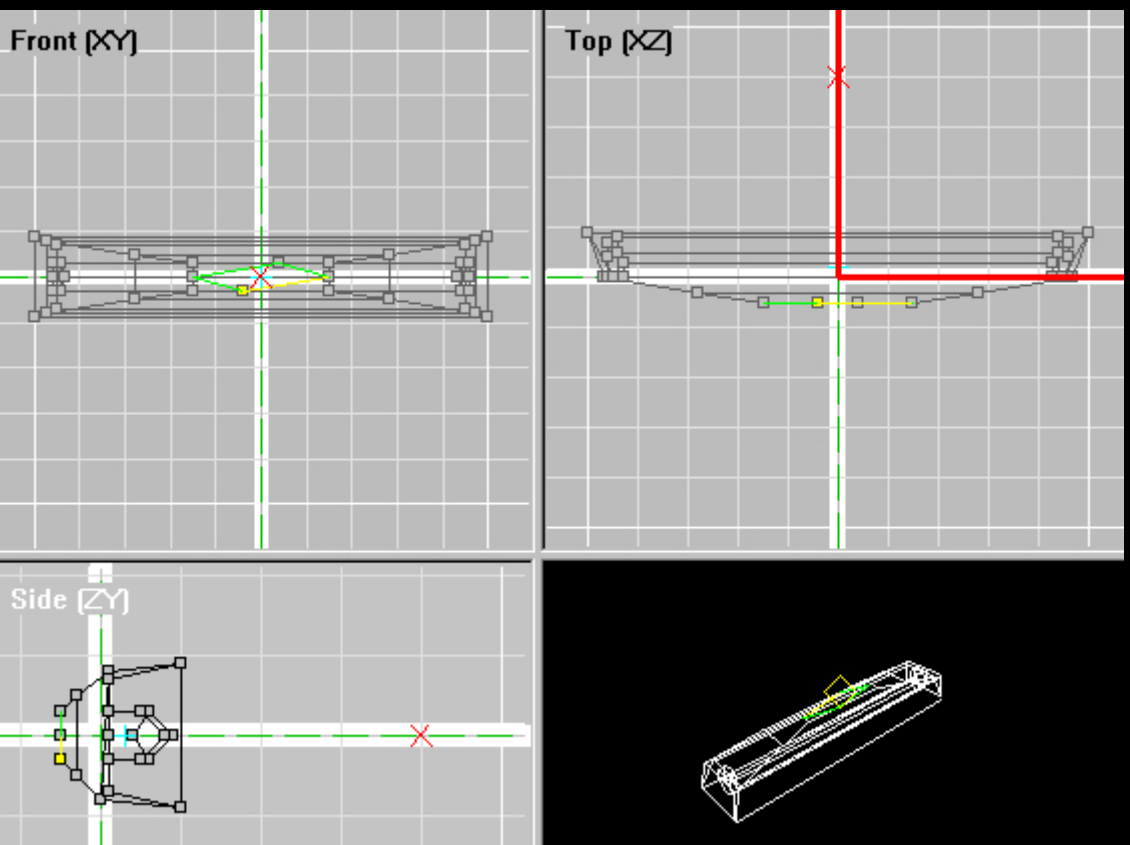


. Wenn ihr hier was angebt, muß die Flag Directional gesetzt sein.

Wie Ihr rechts sehen könnt wird der Rand des Lichtkegels mit kleineren DFOV-Werten weicher.

End red/green/blue component: Lichtfarbe am Ende der Light distance. Auch diese Werte sind in Anteilen anzugeben.

Flags:

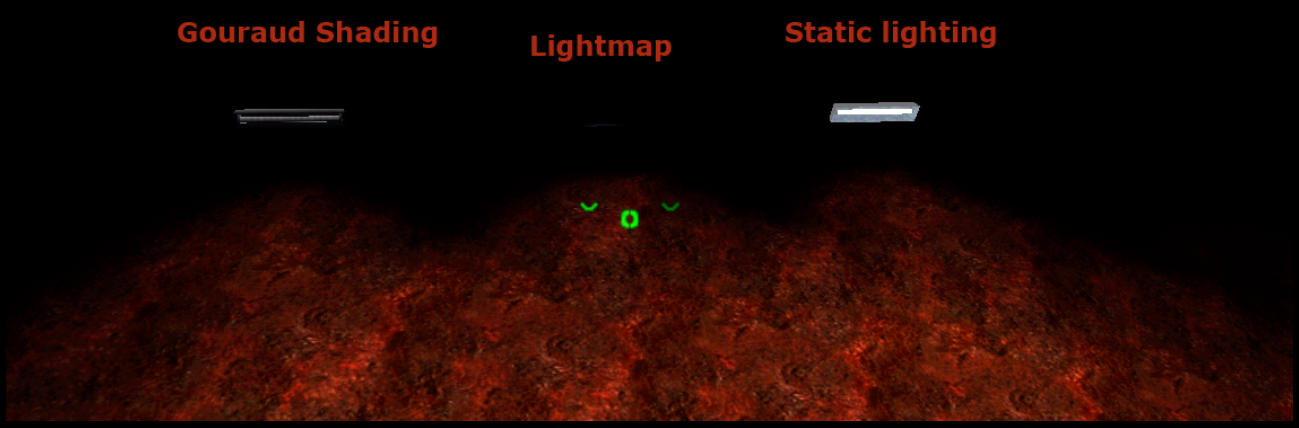


Flicker: Läßt das Licht flackern.

Directional: Wenn Ihr ein Richtungslicht wollt, müßt ihr diese Flag setzen. Die Abstrahlrichtung des Lichts ist parallel zur Z-Achse, von negativ nach positiv; ihr müßt also euer Objekt beim Erstellen in D3Edit entsprechend ausrichten. (links) Ist

diese Flag gesetzt, könnt ihr unter Directional field of view (s.o.) bestimmen, welchen Winkel der Lichtkegel haben soll.
Pulse läßt das licht pulsieren, Ihr müßt dann unter Time interval den Rhytmus in Sekunden angeben.

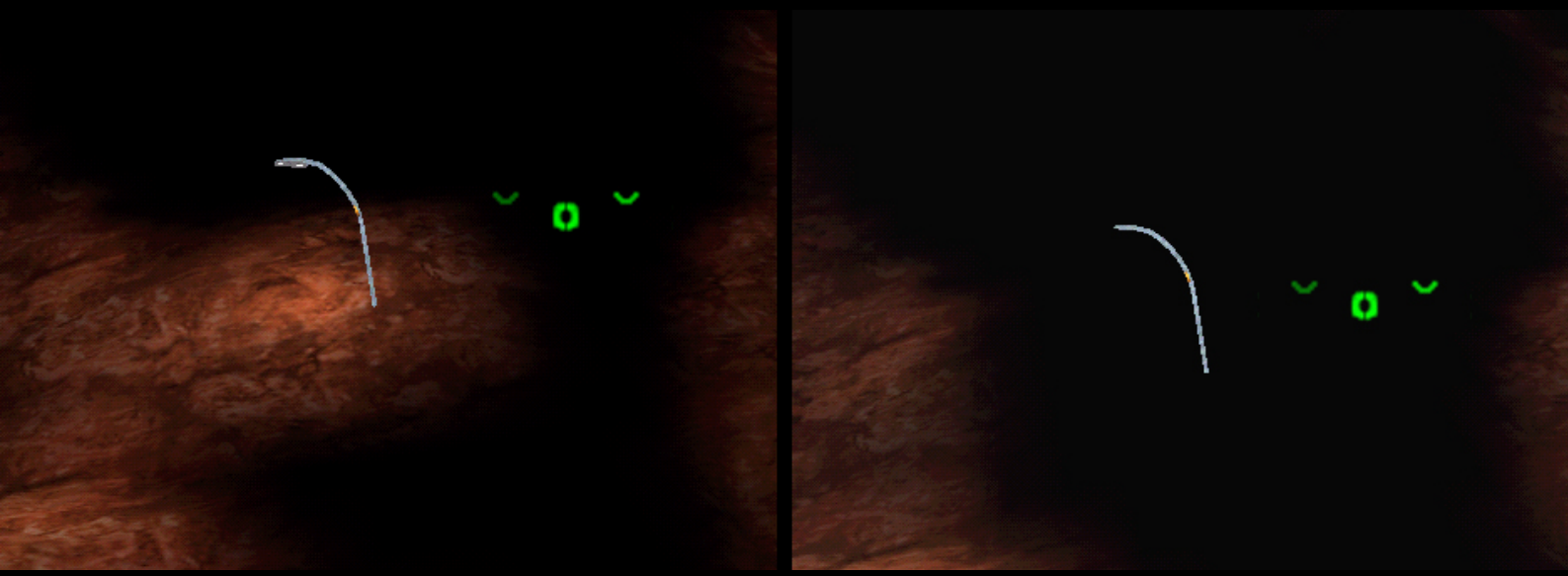
Flicker slightly hat bewirkt, daß garkein Licht mehr abgestrahlt wurde.



Render type: Gibt an, wie das Lichtobjekt ingame gezeichnet werden soll.

Mit den anderen Flags habe ich mich noch nicht ausreichend auseinandergesetzt; da passieren teilweise recht komische Sachen.
Wenn ihr im Lighting Chunk etwas Ungültiges an- bzw. eingebt, wird das Objekt ingame entweder nicht angezeigt oder es gibt kein Licht ab.

Mit dynamischem Licht kann man aber noch was ganz anderes machen: Lichter, die man



ausschießen kann, wie in Descent 2 😬

Im Bild oben seht ihr ein dynamisches Licht am Terrain, in einem Bereich der von einer Erhebung im Terrain beschattet wird; links vor, rechts nach dem Kaputtschießen des Lichtobjektes.

Die Sache hat jedoch auch einen Haken, denn solch lichtwerfende Objekte sind waschechte FPS-Killer. Ich habe in einem Level zum Ausprobieren 18 solcher Dinger reingestellt und hatte durchwegs, überall im Level, so um die 20...30 FPS. Nachdem ich alle Lichter ausgeschossen hatte waren's wieder ordentliche 200 und mehr. Allerdings ist mein Rechner auch schon etwas älter.

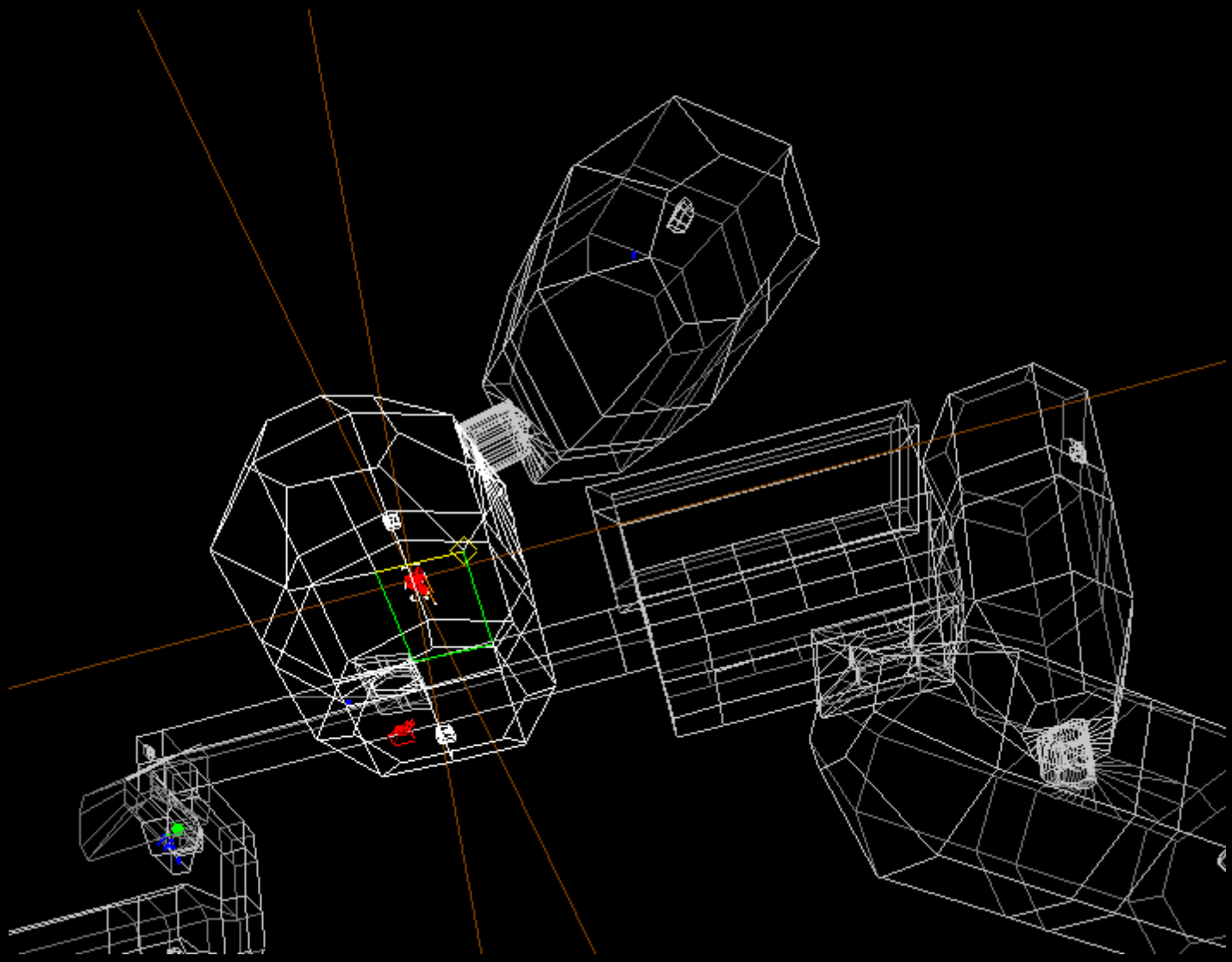
123 - Gruppendateien

Ragil Ral

Vielleicht ist es Euch schon mal passiert, daß Ihr einen Teil der Mine woanders haben wolltet oder daß ihr aus dem Valid MP-Bereich herausgebaut habt.
Bei älteren D3Edit-Versionen mußte man da noch diese Räume löschen, vorher eventuell wegspeichern da man an ihnen herumgeschraubt hat, erneut einfügen und dann alle Objekte und Türen nochmals einsetzen. Für diesen Fall gibt es nun die Möglichkeit, Gruppen von Räumen in eine Datei auszulagern und wie einen Raum zu attachen, inklusive aller Objekte und Türen. Dies eignet sich auch dafür, Konstruktionen aus mehreren Räumen mehrmalig zu attachen.
Klingt aufwendig, doch Atan hat hier einen sensationellen Job erledigt.

Ausgangslage

Ihr habt also einen Level wo Ihr mehrere Räume woanders haben, oder diese Räume mehrmals verwenden wollt.

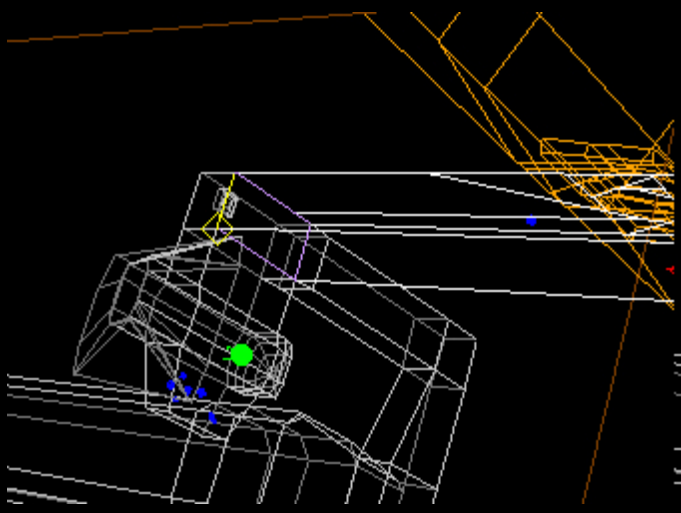


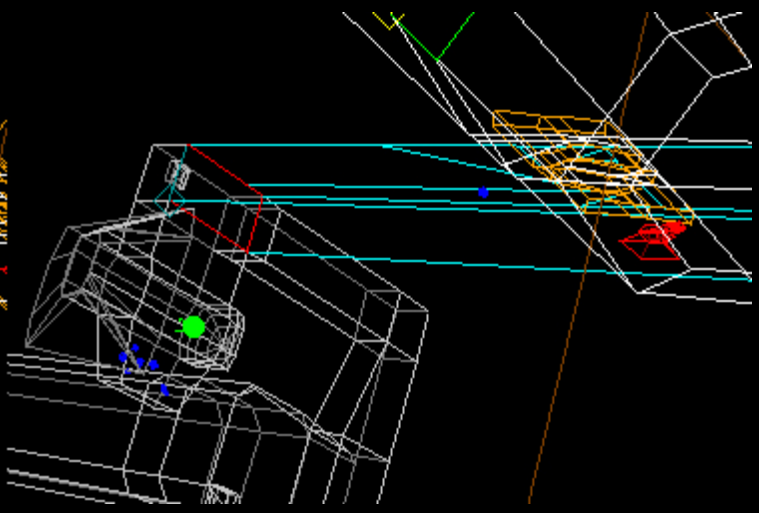
Wie Ihr seht, enthalten die Räume bereits Objekte und sind über Türen verbunden. Und nun kommt Trick grp:

Markieren der gewünschten Räume

Bleibt in der World View und klickt einen der Räume an. Danach drückt **Space**: die Statuszeile sagt dann Room xx selected. Macht das für alle Räume, um die es Euch geht. Selektierte Räume werden dann in Orange gezeichnet (außer ein selektierter Raum ist aktiv):

Danach müßt Ihr entscheiden, mit welchem Face die Gruppe dann attached werden soll. Bewegt Euch zu diesem Face, klickt es an und drückt - immer noch in der World View - **M**: Die Statuszeile sagt dann Room:Face:Edge:Vertex x:x:x:x marked. Dieses Face darf auch ein Portal sein.

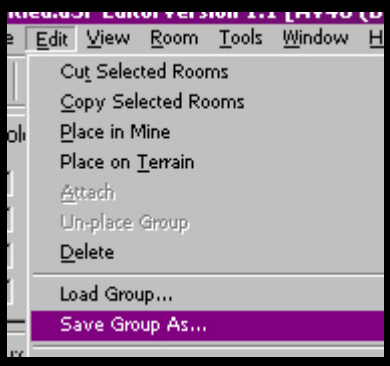




Im oberen Bild ist links der Raum aktiv, in dem das Face - hier ein Portal - markiert wurde, rechts ein anderer Raum. Beachtet die Farbcodierung: Der Raum mit dem markierten Face wird türkis, das markierte Face rot dargestellt wenn er nicht der aktive ist. Ebenfalls rechts, der aktive Raum ist einer der selektierten, er wird weiß dargestellt.

Kopieren bzw. Ausschneiden

Nun müßt ihr entscheiden ob Ihr die Räume woanders oder nochmals haben wollt. Wenn Ihr sie umpflanzen wollt drückt **Strg-X**, ansonsten **Strg-C** - immer noch in der World View: Die Statuszeile meldet x rooms copied to group (wenn Ihr sie kopiert) bzw. x rooms cut to group wenn ihr sie aussschneidet. Danach geht Ihr nach Edit -> Save Group As... und gebt dem Dingens einen Namen, klickt auf Speichern. Es wird eine Datei mit der Endung `.grp` geschrieben.

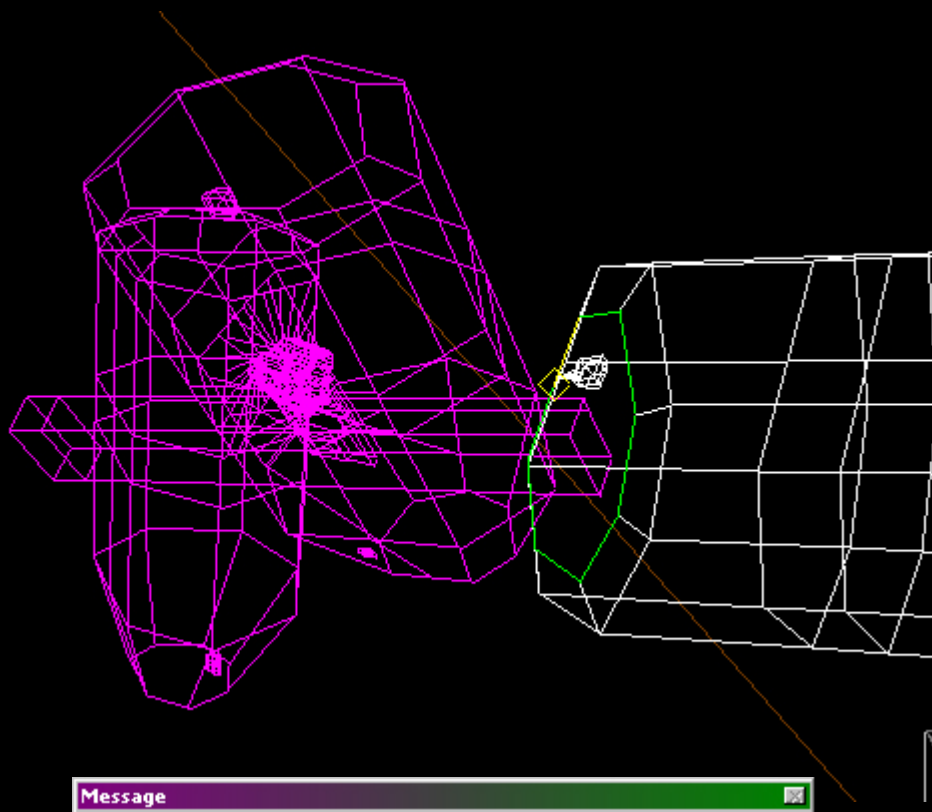


Attachen der Gruppe

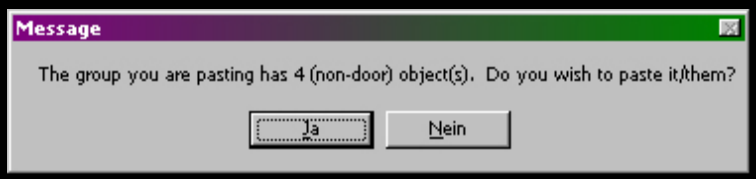
Hier geht ihr vor wie Ihr es bei Räumen gewohnt seid: zuerst macht ein Face current wo Ihr die Gruppe haben wollt. Danach aber geht Ihr auf Edit->Load Group..., und wählt Eure soeben verspeicherte Gruppe. Es passiert aber noch nichts!

Dann geht Ihr auf Edit->Place in Mine oder macht **Strg-V**, und die Gruppe wird in die Mine gesetzt:

Ihr könnt die Gruppe jetzt noch positionieren.



Danach macht Ihr Edit-> Attach, und bekommt - wenn Ihr Objekte in der Gruppe habt - diesen Dialog:



Ihr könnt also noch entscheiden, ob Ihr die Objekte miteinfügen wollt. Et voila...

...ne tolle Sache, oder? Türen, Objekte, alles da.

Und noch ein Trick

Da das Gruppen-Feature es erlaubt mehrere Räume auszuschneiden und diese dann aus der Mine entfernt werden, kann man über diesen Weg mehrere Räume löschen - Ihr müßt die Gruppe ja nicht speichern. Sie bleibt aber solange in der Zwischenablage, bis Ihr etwas anderes kopiert bzw. ausschneidet.

Und noch einer

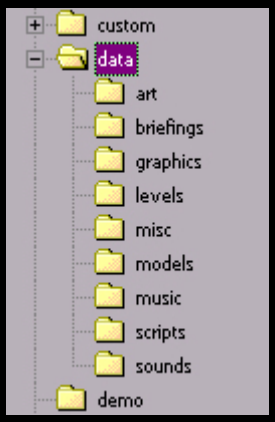
Wer genau hingeguckt hat, dem wird Edit->Place on Terrain aufgefallen sein. Man kann Gruppen auch am Terrain platzieren, dazu muß eine Terrainzelle selektiert sein; das markierte Face der Gruppe wird dann antiplanparallel zur Terrainzelle ausgerichtet, samt der Gruppe. Achtet aber darauf nur eine Terrainzelle zu markieren, sonst:

124 - Customs, nochmal

Ragil Ral

In den Abschnitten F und H wurde viel über Custom-Texturen und -Objekte, ihren `.gam's` und deren Verwendung schwadroniert. Es sind mehrere Möglichkeiten aufgezeigt worden, wie man Customs handhaben kann, hier soll nun eine weitere - bei weitem die bequemste - behandelt werden.

Atan hat herausgefunden, daß die Entwickler von Descent mit einer Ordnerstruktur gearbeitet haben, die wir Levelbastler uns zunutze machen können.



Erstellt Euch folgende Ordnerstruktur im Ordner von D3:

- ?
...
- hier stellt Ihr Eure `.ogf`'s und `.oaf`'s rein

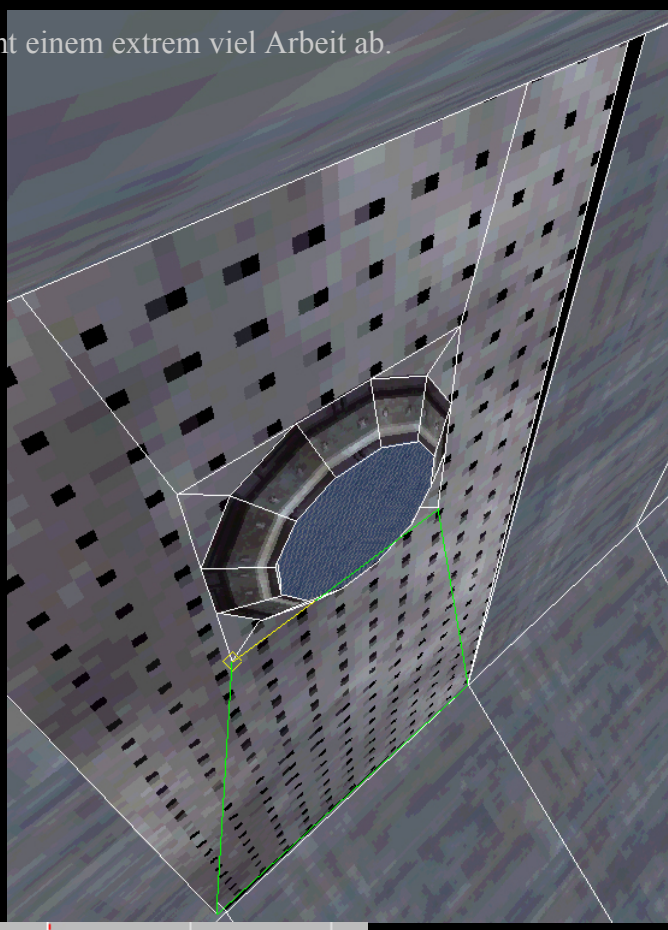
- ?
- hier kommen die `.oof`'s rein
- Music files
- Dallas-Skripten
- hier kommen `.wav`'s für Soundeffekte rein

Wenn Ihr dort eure Customs reinstellt, habt Ihr sie im Editor immer sichtbar und spart Euch das Gepacke von `.hog`-Files. Außerdem habt Ihr dann eine 'zentrale Sammelstelle' und müßt euch zum Packen des Levels nicht immer alles zusammensuchen. Weiters, wenn Ihr hier etwas ablegt das den gleichen Namen wie ein Original-Bauteil von D3 hat, wird beim Bauen dieses verwendet anstatt jenem, daß sich in der `d3.hog` befindet. Auf diese Weise könnt Ihr zum Beispiel die Original-Türen reparieren; Ihr packt die Tür aus der `d3.hog` aus, repariert sie und legt sie unter `models` ab - ab sofort wird diese verwendet. Ihr müßt dann nurmehr die reparierte Tür eurem Level beipacken und braucht keinen extra `.gam`-Eintrag dafür.

125 - Bauen mit Objekten

Ragil Ral

Die hier beschriebene Methode stammt von Atan, und nimmt einem extrem viel Arbeit ab.



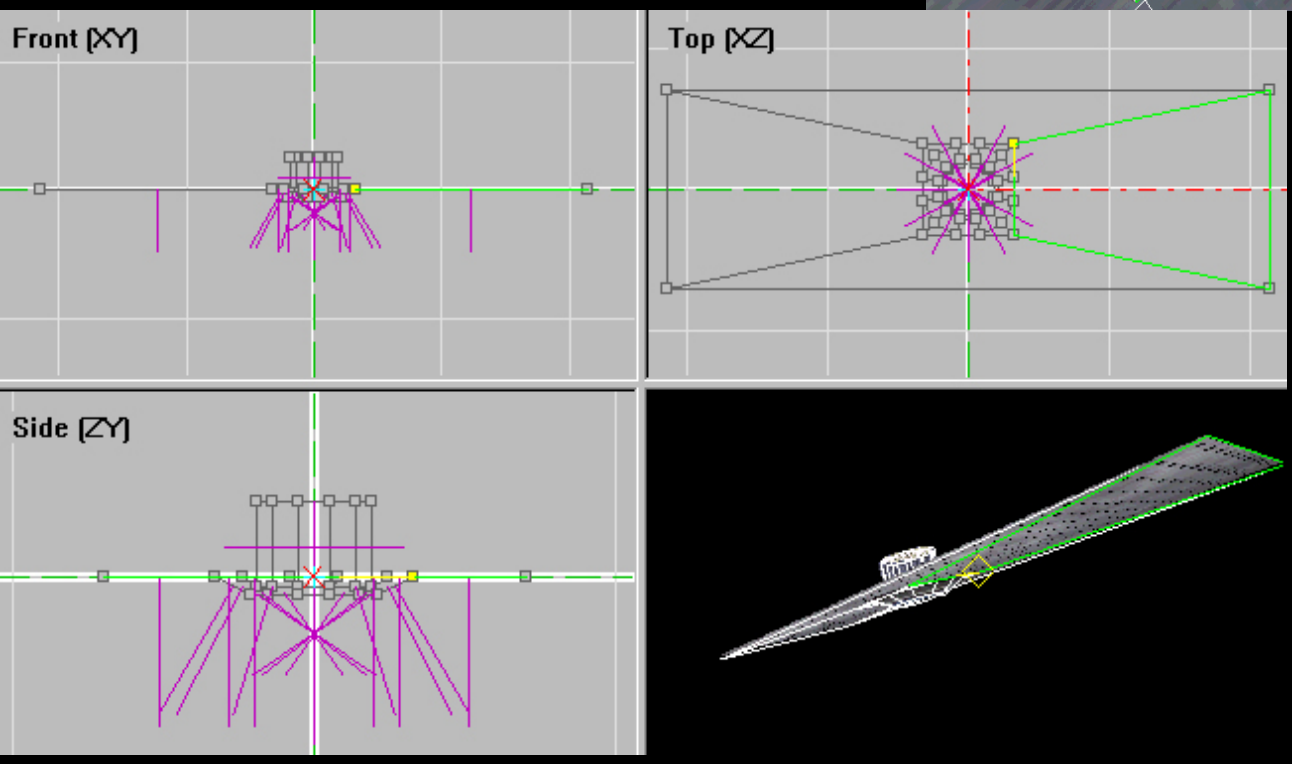
Die Problematik

Ihr werdet sicher bereits öfter das Problem gehabt haben, ein Facekonstrukt an irgendetwas auszurichten, aber da D3Edit Winkeloperationen in diskreten Schritten - minimal 1,4063° - ausführt, ist die Ausrichtung manchesmal einfach nicht genau durchzuführen. Rechts seht Ihr so ein Beispiel, die Lampe läßt sich nicht genau einpassen, es bleiben Lücken.

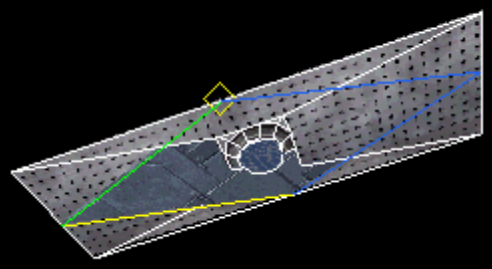
Es gibt Abhilfe!
In ganz kurz : Das Konstrukt, daß man öfter verwenden und immer wieder ausrichten muß wird in ein Objekt mit Groundpoint verwandelt und kann so über die `.gam` in ein Level eingebunden werden. Wird es dann eingefügt, liegt es planparallel. Danach kann man es ins Level mergen.

Das Objekt der Begierde

Zunächst erstellt Ihr Euch einen Raum, wo Ihr das Objekt reinstellt:



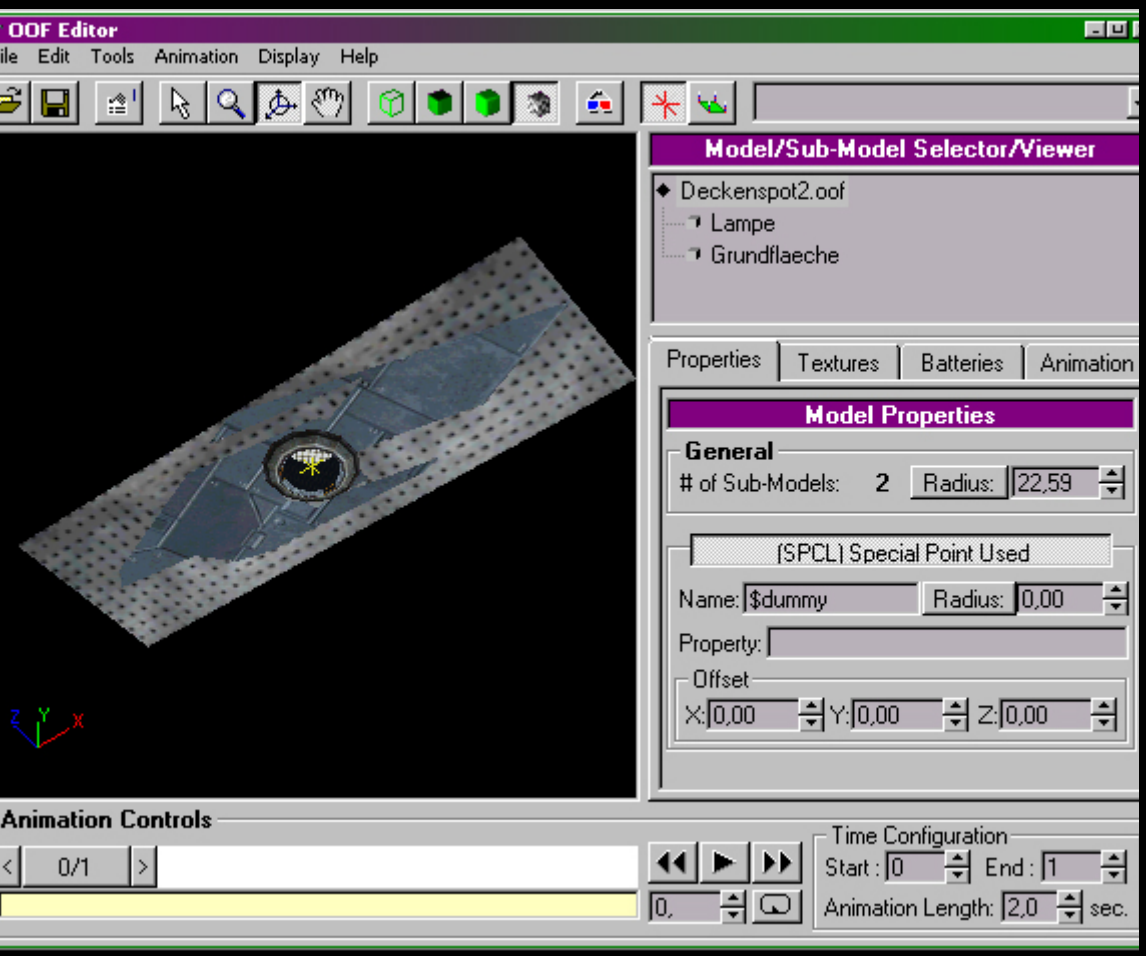
Nun braucht Ihr noch ein Face, daß dann zum Groundpoint konvertiert wird; dieses wird dann zur Ausrichtung des Objektes verwendet. Bei der Lampe oben sollen die vier umgebenden Faces plan zum Face liegen wo man sie einbaut; Fügt also ein Face derart ein, daß es Euren Vorstellungen entspricht.



Welche Textur das Face hat ist egal. Die Normale bestimmt die Richtung in die das Objekt nach dem Einbau blickt, muß daher in die gleiche Richtung gucken, wie das Face an welches das Objekt an- oder eingebaut werden soll.

Es ist von Vorteil das Konstrukt möglichst mittig zum Koordinatenursprung zu machen. Speichert den Raum ab, und schmeißt das OOFedit an; ladet dort Euren Raum mit dem Objekt.

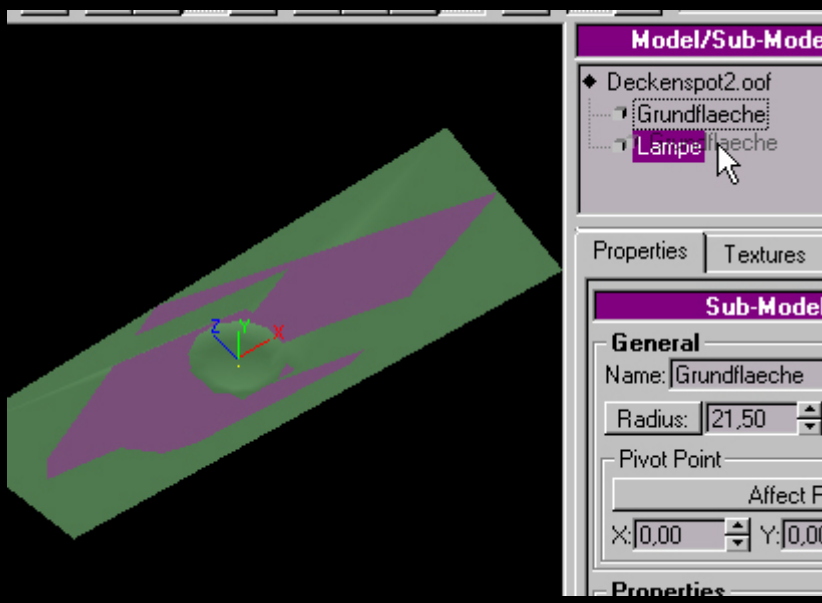
Vom Raum zum Objekt



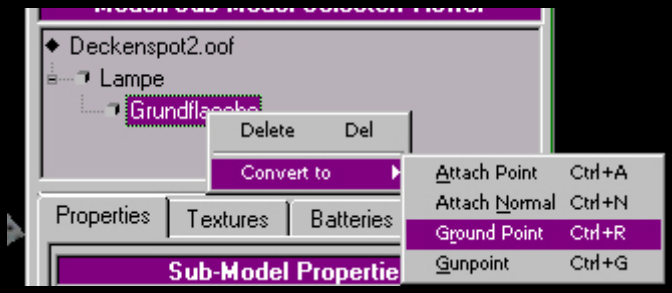
Im OOFedit vergebe ich immer das Special Point Used mit dem Namen `$dummy`; habs noch nie ohne probiert ☺
Die Benennungen sind nicht zwingend nötig, dienen nur zur Klarheit hier.
Die Pivotpunkte müssen auf 0 gesetzt werden, es sei denn Ihr wollt es anders; nach dem mergen wird das Konstrukt nämlich mit seinem Pivot-Punkt an das Face ausgerichtet.

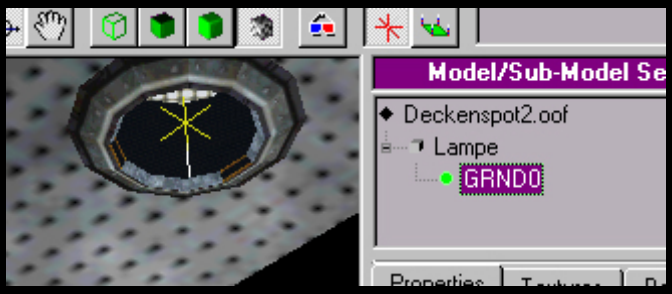
Als nächstes macht das Face welches zur Grundfläche werden wird zu einem Submodell von eurem Konstrukt, indem Ihr es Links anklickt und auf das Objekt zieht:

Hier kann man wohl nix falsch machen 😬



Danach rechtsklickt auf die zukünftige Grundfläche und wählt Convert to->Ground Point; das Face verschwindet und wird durch ein gelbes Raumkreuz ersetzt, das einen längeren weißen Strich hat (unten): der Groundpoint.





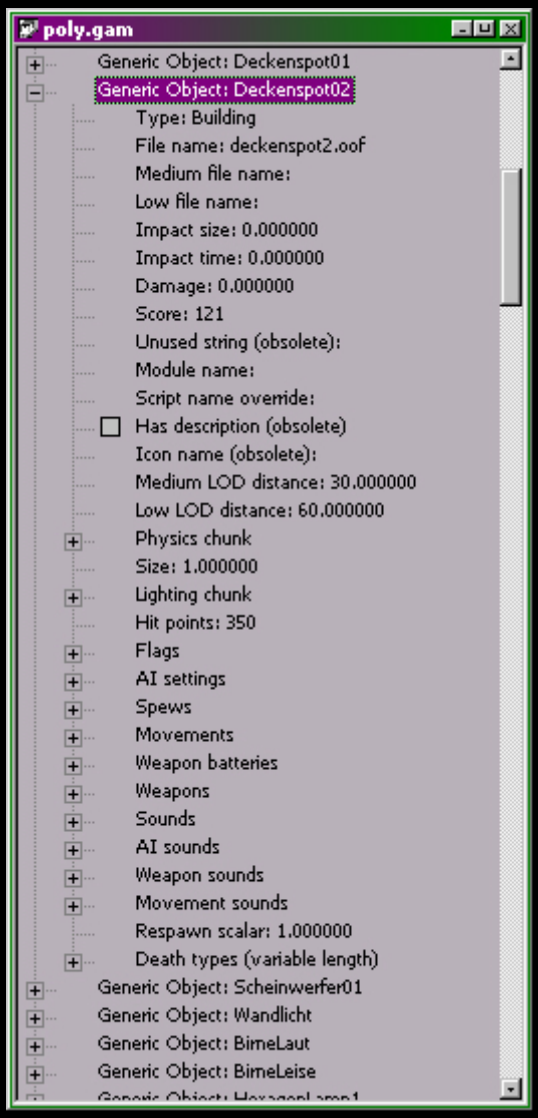
Wie ihr sehen könnt, wurde auch die Benennung geändert. Speichert nun das Objekt als `.oof` ab.

Dieses war der eine Teil.

Um das Objekt auch zu verwenden, muß es einen `.gam`-Eintrag haben.

Das Objekt einbinden

Werft also das Gamtool an, öffnet oder erstellt eine `.gam`-Datei, erstellt bzw. kopiert Euch also einen `.gam`-Eintrag für ein Generic Object. Benennt es, tragt die `.oof` ein, den Rest braucht Ihr nur setzen wenn Ihr das Objekt als solches verwenden wollt; wenn Ihr es mergen werdet, könnt Ihr das so lassen wie es ist.

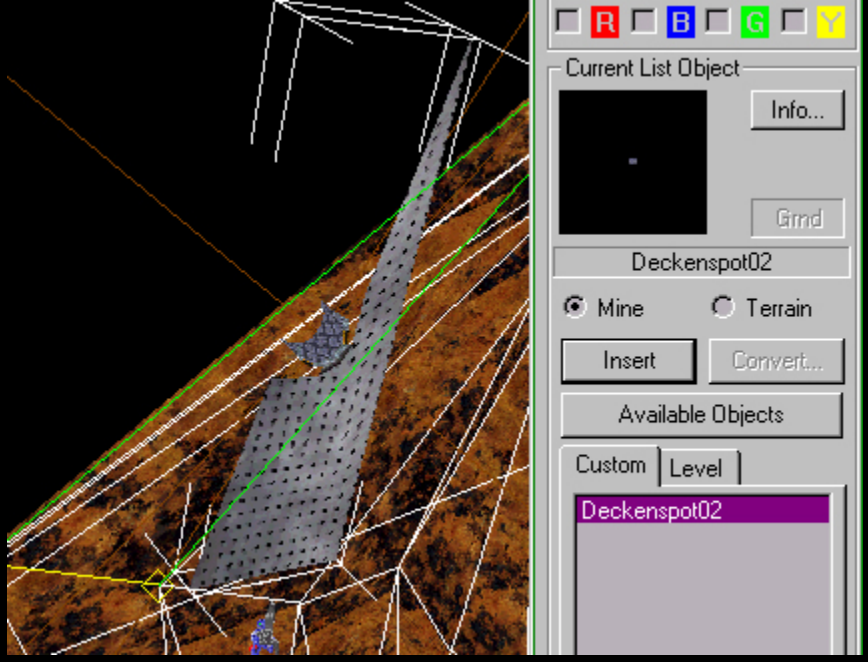


Ordnet es entweder dem Type Clutter oder Building zu, ich bevorzuge letzteres (es soll ja ein Bauelement werden, außerdem gibt es da weniger in der Liste, wodurch sie schneller lädt).

Um das Objekt ordentlich zu verwenden, vollführt noch die Schritte die nötig sind um es in D3Edit zu sehen, ich empfehle die Methode Nr117 - Nochmal Customs; Hier brauche ich das Objekt bloß in `<descent3dir>\data\models` abzulegen, fertig.

Ladet dann vor dem Öffnen Eures Levels die `.gam`.

Öffnet den Level, öffnet die Objekt Bar (O), zieht Euch euer Objekt in die Custom-Liste, wählt ein Face aus wo es hinsoll und klickt **Insert**:

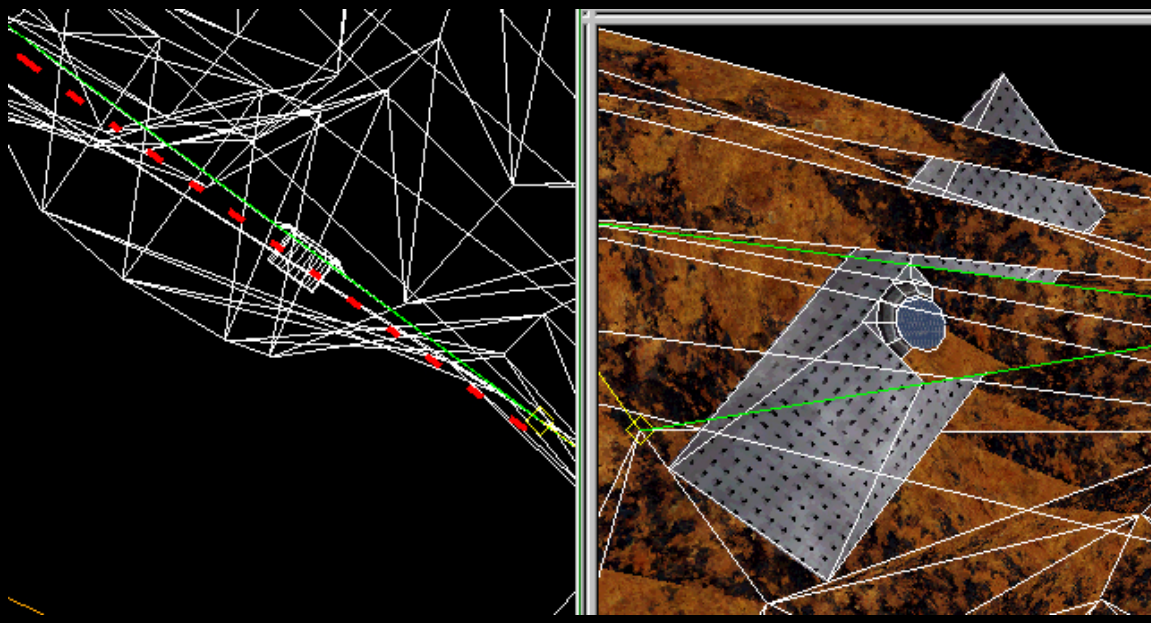


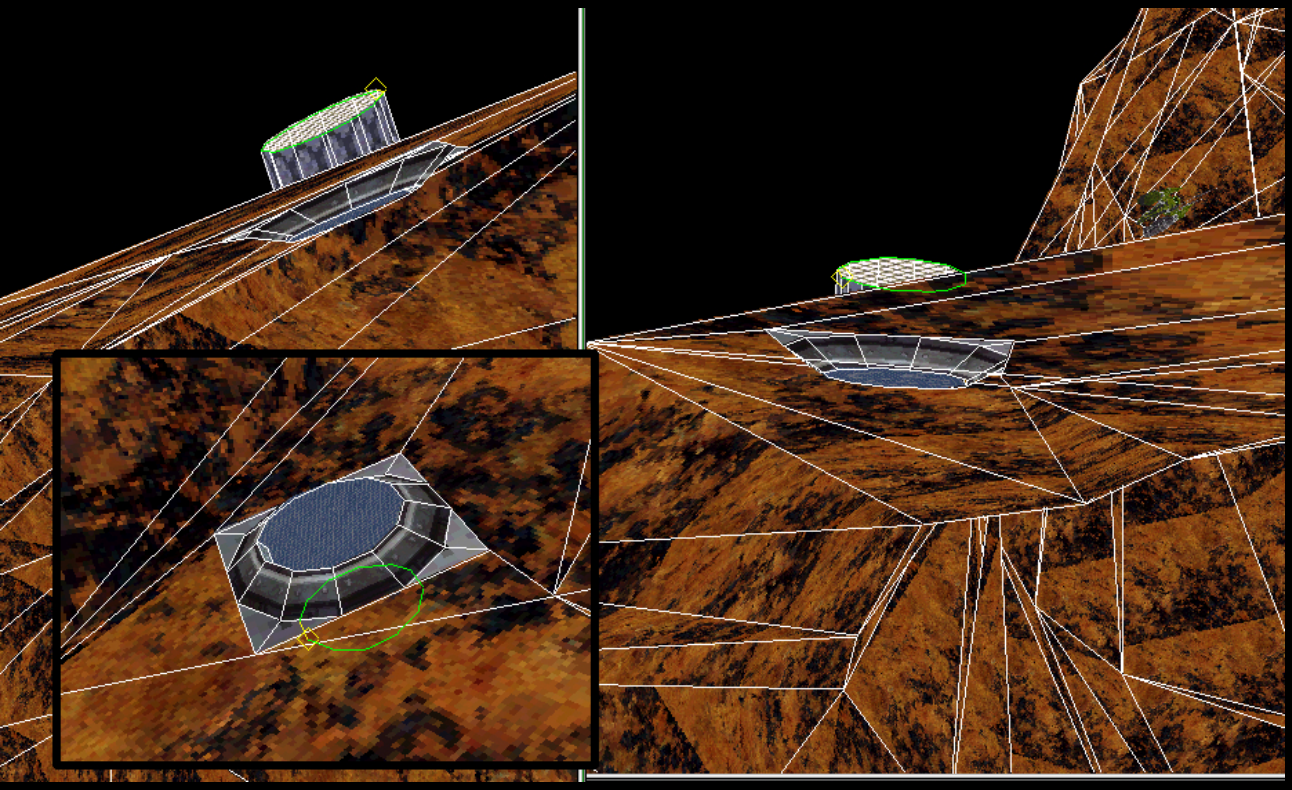
Das Objekt wird mit der Ebene des Faces das jetzt ein Groundpoint ist planparall zum current Face eingefügt.

Danach geht in die World View, klickt Room->Merge Object into Room, und das Objekt wird in ein Facekonstrukt verwandelt, daß Ihr dann in eure Shell integrieren könnt (unten.).

Die rote Linie soll unterstreichen, daß das Objekt planparallel liegt.

Ein wenig drehen, und/oder verschieben, dann noch die Faces einpassen...

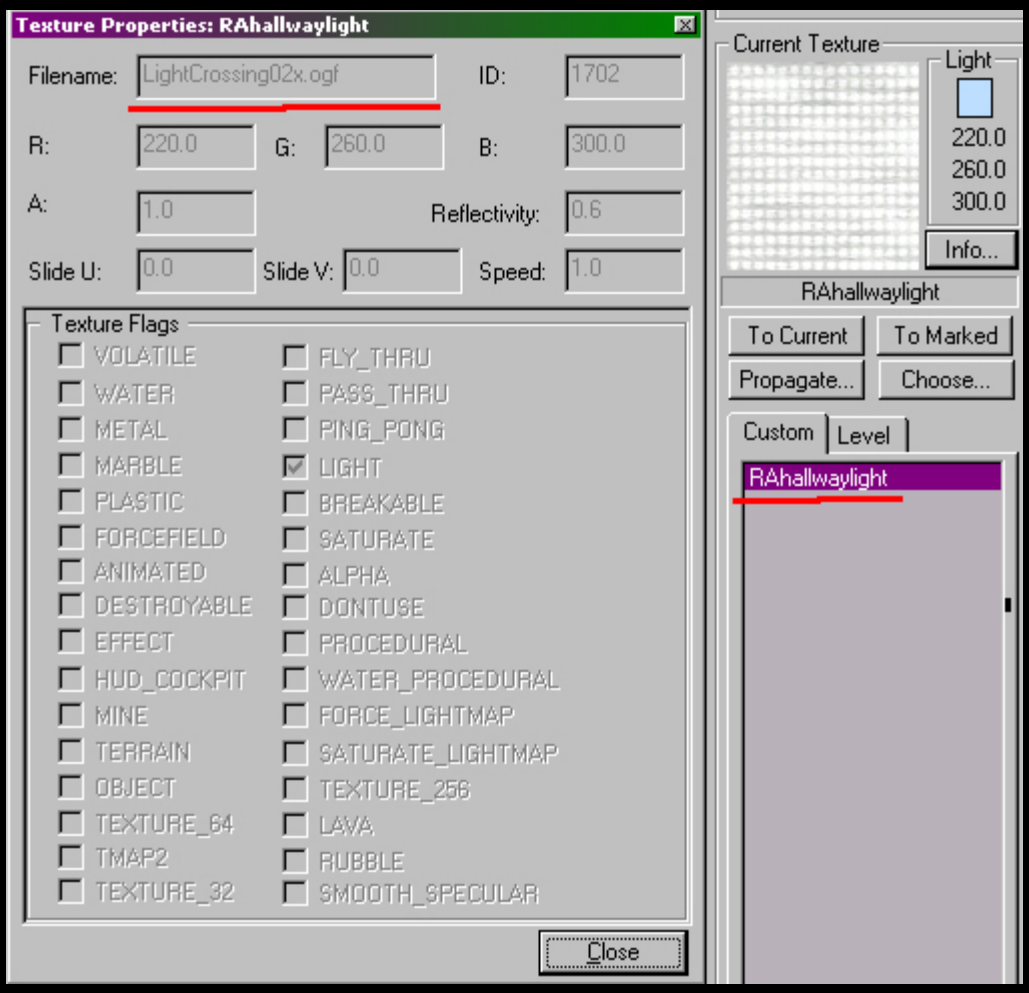




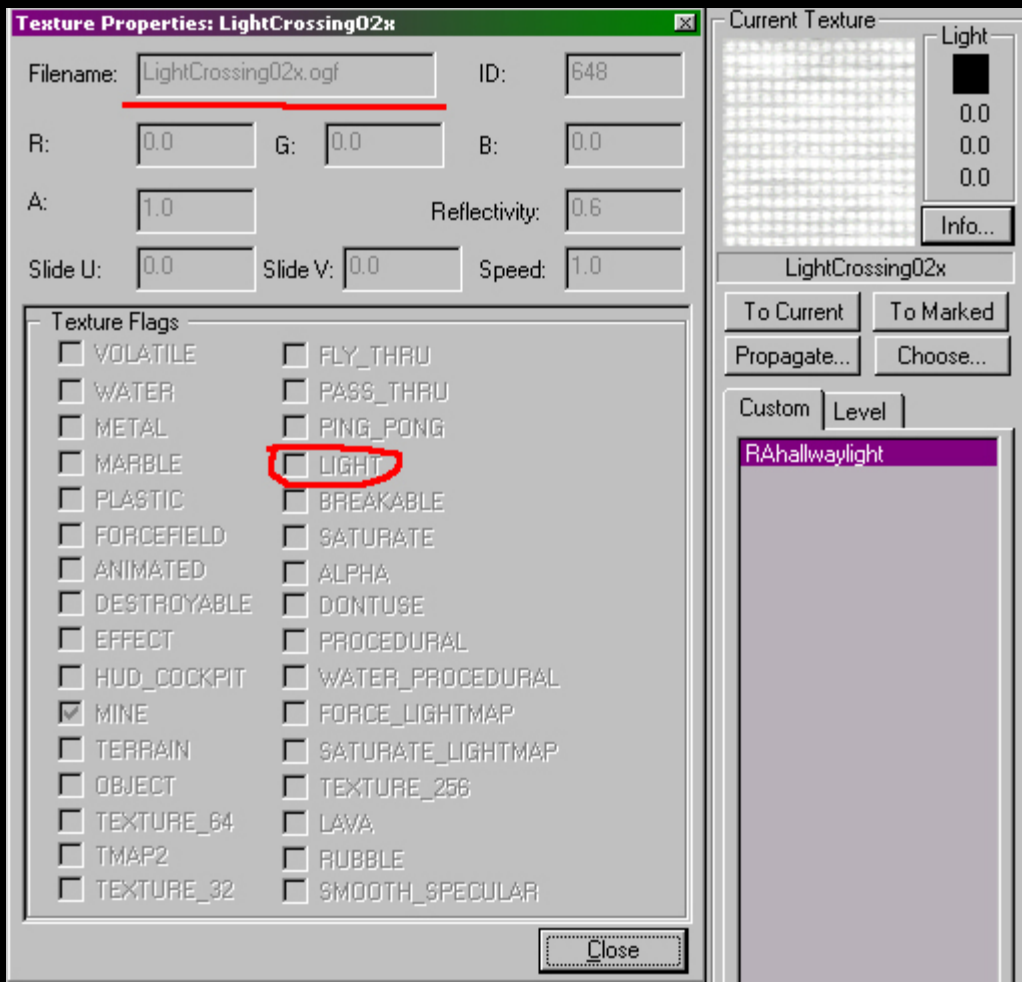
...und das Ganze sitzt perfekt!

Verify wird hier keine Fehler melden.

Eine Sache ist allerdings:
Achtet bei Lichttexturen, daß Ihr dann im eingemergten Objekt auch die richtige habt; es wird nämlich öfters mal ein und dieselbe `.ogf`-Datei für mehrere Lichter verwendet. Habt Ihr etwa im Objekt die 'RAhallwaylight' als Lichttextur, wird sie nach dem Mergen zur zugrundeliegenden Textur geändert, das ist in diesem Fall die 'LightCrossing02x'; und die gibt garkein Licht.



Links: die RAhallwaylight-Eigenschaften, achtet auf Filename und die Lichtwerte.



Rechts: Die zugrundeliegende `.ogf` hat das Lichtflag nicht

Nochmals Dank an Atan für diesen Schmee!

126 - Mercenary-Bots verwenden

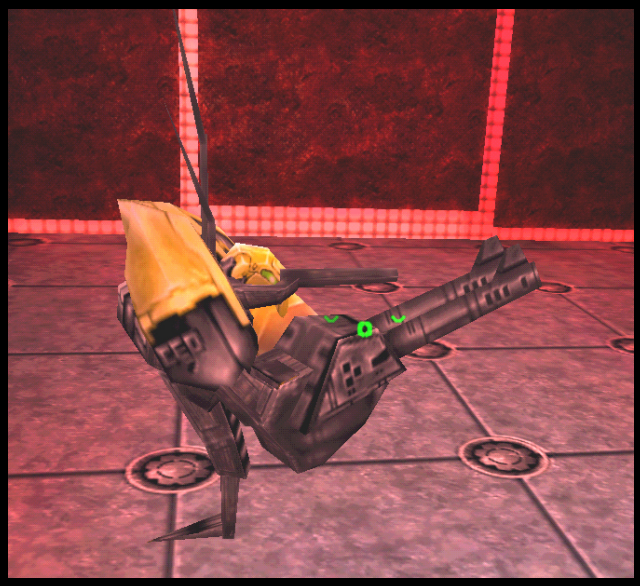
Ragil Ral

Es ist zwar möglich die Mercenary-Bots als Objekt ins Level zu holen, die tun dann aber entweder garnix oder wenig. Das liegt daran, daß für die Bots im Addon-Pack einiges programmiert wurde weswegen sie mit der Standard-`dll` keine Aktionen vollführen.

Um Merc-Bots zu verwenden, muß man also alles Relevante ins eigene Level hinzufügen. Das ist im einzelnen:

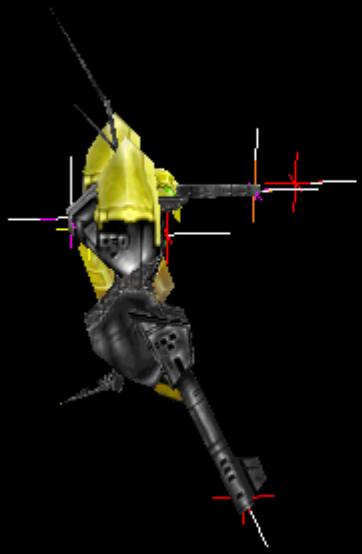
1. Der Bot (na nona),
2. alle Objekte die zu diesem gehören,
3. die Texturen die zu diesen Objekten gehören,
4. natürlich die GAM-Seiten für die Objekte und Texturen,
5. die entsprechende `.dll` für das Verhalten des Bots.

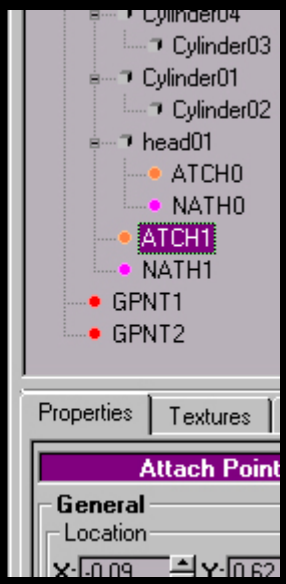
Der Black Stormtrooper aus dem Mercenaries-Level 7 hat es mir schon immer angetan, die Geschichte mit dem roten Laservisier ist einfach endgeil. Dieses Tut wird ihn verwenden. Also extrahiert mit dem Quickedit erst mal die `merc.hog` und die `merc.mn3`, die Dateien die wir benötigen sind zwischen diesen beiden Hog-Files aufgeteilt.



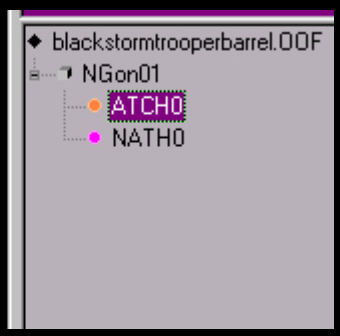
Der Bot selbst ist schnell eingebaut; behandelt ihn wie ein normales Custom-Objekt. Ihr nehmt also die `.oof` des Bots und seinen GAM-Eintrag und baut beides in Euer Level ein. Wenn Ihr Euch das dann ingame anguckt, werdet Ihr bemerken daß sich der Bot nichtmal dreht - man muß ihm schon direkt in die Schußlinie fliegen, bevor der feuert. Also nicht so ganz das Wahre.

Ein anderes Feature dieses Bots ist die Explosion wenn man ihn killt, diese wird über ein weiteres Objekt realisiert, welches an den Bot angehängt ist. Herauszufinden welches Objekt das ist, ist leider eine Sisyphusarbeit; zuerst müßt Ihr Euch den Bot im funktionsfähigen Original betrachten, dann im OOFEdit nacheinander alle Objekte öffnen, bis Ihr das Richtige gefunden habt. In diesem Beispiel war es (glücklicherweise) relativ leicht, da die Namensgebung günstig war. Die Objektdateien sind `blackstormtrooper.OOF` für den Bot und `blackstormtrooperbarrel.OOF` für den 'Rucksack'.



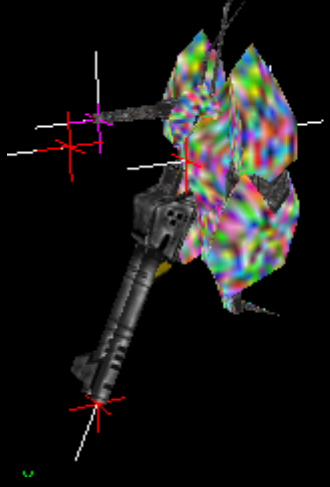


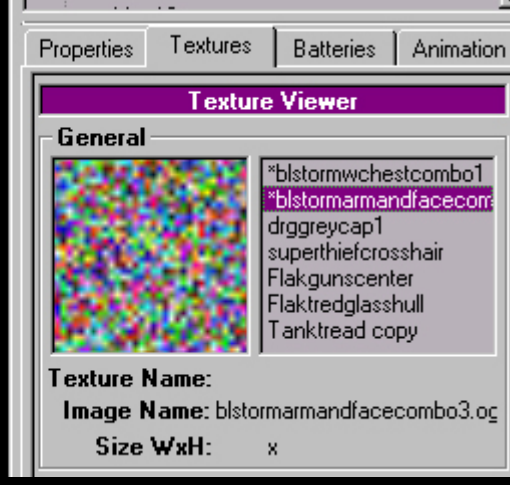
der Black Stormtrooper...

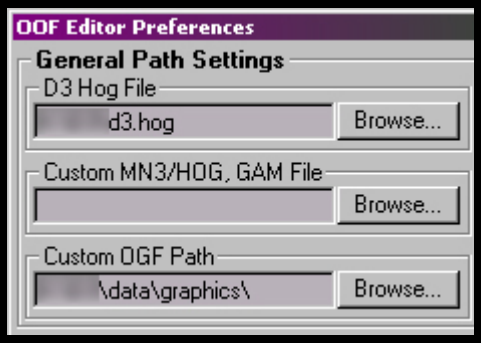


...und sein 'Rucksack'

Falls Ihr im OOFEdit am Objekt 'Farbrausch'-Texturen vorfindet, müßt Ihr noch die zugehörigen Texturen lokalisieren. Das ist einfach, geht ins Register Textures; diejenigen welche fehlen sind mit einem Asterisk gekennzeichnet. Sucht Euch diese zusammen, stellt sie nach `<D3dir>/data/graphics` und teilt dem OOFEdit mit wo diese liegen: Öffnet den Einstellungs-Dialog von OOFEdit (File->Preferences...),





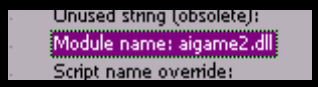


und unter Custom OGF Path zeigt Ihr einfach auf das entsprechende Verzeichnis. Am bequemsten ist es wenn Ihr es so macht wie in 'Customs, nochmal' beschrieben. Natürlich könnt Ihr hier ein beliebiges Verzeichnis nehmen, nur sollten die Texturen sich auch dort befinden.

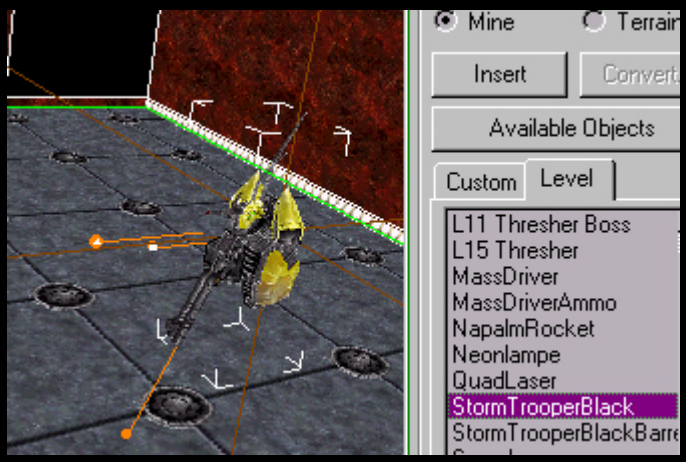
Sobald OOFEdit über diese Info verfügt, schlägt es Euch vor die Texturen neu zu laden und zeigt diese dann auch gleich an.

Die GAM-Seiten holt Ihr Euch aus der table.gam, welche in der `merc.hog` abgelegt ist; vergeßt nicht die GAM-Einträge für die benötigten Texturen.

Wenn Ihr den GAM-Eintrag für den Bot öffnet, seht Ihr daß unter Module name etwas eingetragen ist - dies ist die `dll`, welche den Bot 'belebt'. Diese müßt Ihr finden; sie befindet sich in der `merc.mn3`. Auch die GAM für den 'Rucksack' verweist auf diese Datei.

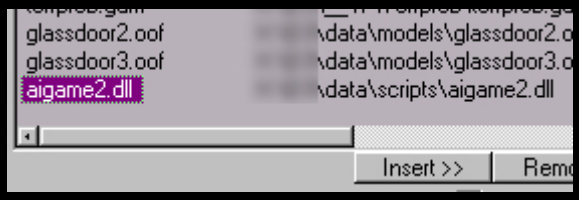


Legt diese in `<D3dir>/data/scripts` ab.



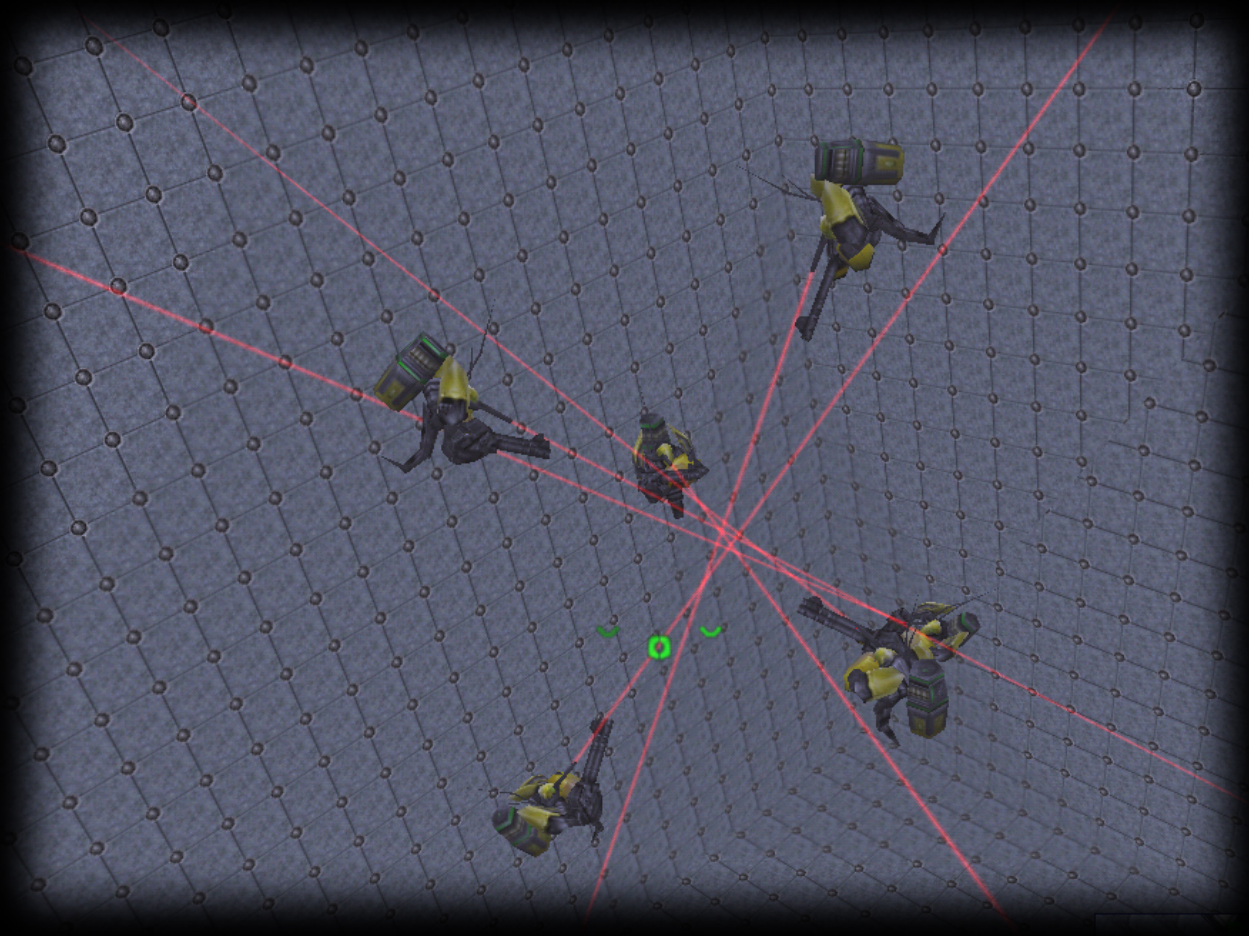
Nun baut Euch den Bot ins Level ein.

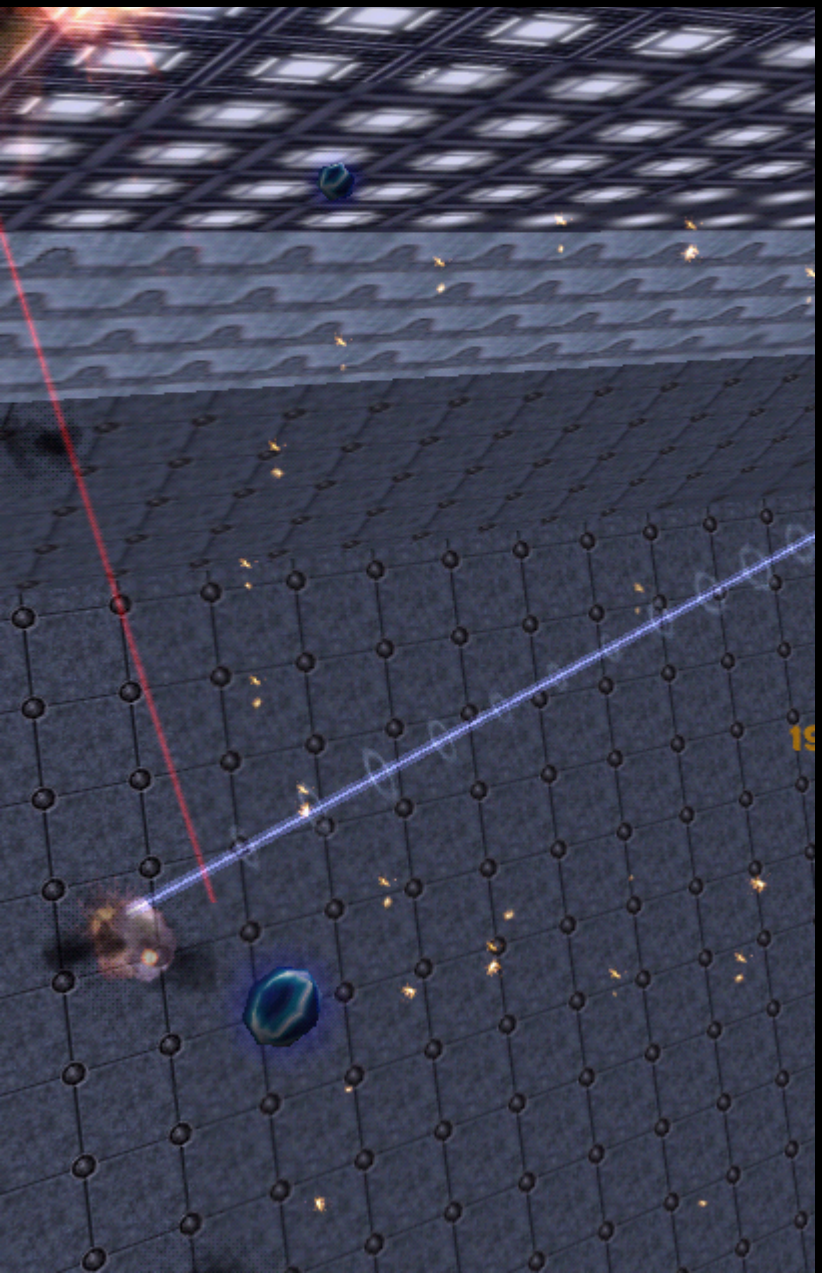
Damit der auch macht was man sich erwartet, muß noch die `dll` mit ins Level, ihr fügt sie im Quickedit ein:



Packt den Level dann zusammen und steigt in Euren Pyro.

Der Robot ist nun scharf!





Wie Ihr oben sehen könnt, haben die Bots nun ihre 'Rucksäcke.'

Links seht Ihr den Schrapnell, welcher von den explodierenden Bots ausgeht... Besser Abstand halten! 😬

127 - Schiffswahl in Singleplayer - toprog

Ragil Ral

Thread: /www.descentforum.de/forum/viewtopic.php?t=4261&sid=eacfd4b2f87dc1b7ac2451d907033ba4

LEON:
Hey.
How do I make the ship selection screen b4 the level start, where you can choose between pyro, pheenix or the tank? Or even BP if that is posible.

Atan:
Not tried this up to now:

DALLAS
Levelstart ->
ACTION -> Mission
Enable the given ship [Ship], so it can be chosen in single player
And I believe you can enable the ship if you end the first level and start the second inside level briefing too.
That's why I inserted a Startlevel inside FiringRange if I remember this right.

LEON:
Thanks Atan.

Now I only need to install and get dallas to work.

LEON:
Ok, i've been playin around with dallas, but seems i can't get this to work.

I have 2 levels, a intro level and the main level. First level have the script: owner>level, event>level end, then always performe enable given ship to use in singleplyer.

Must i make a briefing screen, so the ship can show up for selection on the next level? or will the selection screen show up by it self?
And what is the name of the ships? will D3 [dallas] understand 'phoenix' or 'magnum' or 'magnumAHT'?

I tried to open up your Firingrange, but the cpp file wasen't there so i couldn't study the scripts you made.

Atan:

I used the briefing for ship selection in FR.

And for Dallas you must use the names out of your (table) gam.
Look at the little pic below to see how to write the names the right way. (Without 'Ship:' for sure)
But I can't say if this will really work in DALLAS because I didn't used that way up to now.
If you can't get it to work please tell us, I will have a deeper look then.

ships_194.jpg
Beschreibung:
Dateigröße: 6.19 KB

Angeschaut: 105 mal

LEON:
Thanks for your help Atan.

Still no go.

Excuse my newbieness Sehr glücklich .. but acording to my logic.....the 'enable given ship script' used in the first level, doesn't say anything about what the second level should do. So when second level loads, it just loads like nothing have happen. I dont see how the 'enable given ship script' in first level would make second level start with an ship selection screen. But then again....Im really new to this Sehr glücklich

Is it possible that I need to adjust some of my settings in D3edit as well, so it reads all the files right? I'm thinking of the tablefile manager.

Thanks again

Atan:

No, no need to change something if the rest of your script is working in your level well.
Never change a running system! Tag den med ro! Smilie
Please wait, I'll have a look if I can made ship selection working by using a DALLAS script.
May take some time to check this out..

Atan
Moderator

Zitat:
Enables a ship (given it's name) so it can be chosen in single player ship selection dialog.

This means that you can't start with this ship as I thought, it's just to allow the player to choose that ship the way stated above at (next)second level start.
To enter that ship selection dialog you'll need a briefing so I believe.
Maybe there is an other way to realize the ship selection, to see if this would be possible needs some more time now...

Atan
Moderator

I found a function called 'PlayerChangeShip' which IS the one needed, but I wasn't able to use it inside DALLAS.
There is no given interface for us to handle it inside Singleplayer, at least I found none Traurig
So I'm sorry, but I can't find any other way than the briefing to use the ships.
Remember, you still need to enable the wanted ship(s) in the Start level script.
With the end of the level two briefing there will come the ship selection dialog automatically then, no need to search this inside briefing.
I'm not pretty sure about, it's just what I remember, which may be to wrong.., sorry.
I would have to test this out, to make two test levels and a briefing, but I have not the time doing this in the moment.
If you can't figure it out please tell me, then I will take the time and build such a test mission.

LEON
Forum-Nutzer

Atan hat folgendes geschrieben:
Tag den med ro! Smilie

Ok, jeg vil ta det med ro, Smilie Hehe

Atan
Moderator

Ok LEON, now it was me who couldn't sit still Winken

I did a quick build of two level, made a script for level1 which enables the Magnum and starts a timer which will end the level one in 10 secs.
Then I made a mini briefing, just a picture without any other stuff.
Collected both levels, (cpp), dll, msg, brief+picture inside a mission and told level 2 to own the briefing.
This minimal mission works. I attached the mission here so you could try it out.
I added the cpp so you could test the phoenix yourself.
Let me know if the problem is solved.

ShipSelect_Test.zip
Beschreibung:

Download
Dateiname: ShipSelect_Test.zip
Dateigröße: 316.16 KB
Heruntergeladen: 4 mal
Nach oben
Benutzer-Profile anzeigen Private Nachricht senden
LEON
Forum-Nutzer

Anmeldungsdatum: 11.01.2005
Beiträge: 14
Wohnort: Norway

BeitragVerfasst am: 20.10.2008, 23:12 Titel: Antworten mit Zitat
Atan!! That worked!!

Thanks big time for fast service Sehr glücklich

C ya.
Leon
_____

Darkside Heartless
DBB Captain
DBB Captain

Posted Wed Feb 04, 2004 2:55 pm View user's profile Reply with quote Send private message
how do you select ships at the beginning of a level?
[EDIT] I should say, how do you put a ship selection menu at the beginning of a level[EDIT]

Interceptor6
DBB Ace

DBB Ace

Posted Thu Feb 05, 2004 3:52 pm View user's profile Reply with quote Send private message
It involves Dallas. You must select 'enable the given ship, so it can be chosen in single player', which should be under mission under actions in Dallas. You'll either need a setup level or have the other ships selectable after the level is completed for the first time.

..Or, you could try checking the flag 'allow by default' for the Phoenix and Magnum in gamtool. I've never tried it, but it might be worth a shot. It would eliminate the need for a setup level, and simplify the entire process.

good luck.

-Int

Krags
DBB Ace
DBB Ace

Posted Thu Feb 05, 2004 4:39 pm View user's profile Reply with quote Send private message
Also, I think the level needs to have a briefing.

AceCombat
Owned by Timex
Owned by Timex

Posted Sat Feb 07, 2004 1:23 pm View user's profile Reply with quote Send private message
using the "Allowbydefault" does work for D3, i use it all the time

128 - Scripting & Matcen - toprog

Ragil Ral

Thread: http://www.descentbb.net/viewtopic.php?t=14392&sid=9941f91fdb6a0aeca8bcb8f3317c9e60

Author Message
LEON
DBB Ace
DBB Ace

Posted Sat Sep 13, 2008 1:06 am View user's profile Reply with quote Send private message
Sup folks

Im new to scripting. Have just made a few that works. Get a key to open a door, end level on either room enter or bot killed, and stuff.
What I can't figure though, is how to....like...when all bots from an matcen is spawned and killed---> open a door. Or, all matcen bots killed ---> end level.

Anyone?
Thanks

Leon

Kyouryuu
DBB Alumni
DBB Alumni

Posted Sat Sep 13, 2008 10:38 pm View user's profile Reply with quote Send private message
It's been a really long time since I touched DALLAS.

I'm not sure if what you are asking is possible. In order to track the death of a given robot or robots, you have to give them individual names if I remember correctly. The problem with the Matcen is that it is spawning robots on-the-fly and therefore you can't name them.

LEON
DBB Ace
DBB Ace

Posted Mon Sep 15, 2008 5:32 am View user's profile Reply with quote Send private message
Ah, ok, thats some bad news.
Hmmm, I wanted to make a SP lvl where the main fokus was to fight bots. like, you enter a room, the door locks behind you, a matcen start to spawn bots, and when you have killed them all you get access to what ever you was looking for in that room, and the door opens again.
Maybe theres some other way around this, I dunno, if its possible to script the matcen (not the bots) so when its finish, then you get what ever effect you want.

Kyouryuu
DBB Alumni
DBB Alumni

Posted Mon Sep 15, 2008 9:30 am View user's profile Reply with quote Send private message
Maybe don't rely on a Matcen to do it?

Maybe look at how Proving Grounds set this up in Descent 3. I think in those situations, the boss robot was just there already, waiting for your arrival. Another potential option is that you can hide robots until you are ready to use them, IIRC.

LEON
DBB Ace
DBB Ace

Posted Tue Sep 16, 2008 10:12 pm View user's profile Reply with quote Send private message
Matcens is kind a useless if you cant script them. They are just there.....but not part of the story!?

Yes, the bot can be there from the start, but I was thinking of a lot more than just one or even 2-3. More like 20, but fighting 20 bots at the same time is just over the top.
I can hide them you say? Does that mean I must script every single bots? Sound like an option, must be a lot of scripting, though.

If I can't figure a way to do this, I'll come up with another gameplay or story. Is just that I wanted a kind of "anarchy" feel on the mission. You advance from one area to the next one by killing all the bots. What I like about SP levels is the fighting. I don't mind pusles much, and the replay value is low as well.

Ok, I'll try somting
Thanks for your help Kyouryuu

Leon

Foil
DBB Admiral
DBB Admiral

Posted Wed Sep 17, 2008 6:23 am View user's profile Reply with quote Send private message
I'm not sure it can be done through DALLAS (I'd have to go back and look), but it's certainly possible. After all, there's the scripting in the SP level where you have to kill all but a few of the troops.

Duper
DBB Master
DBB Master

Posted Wed Sep 17, 2008 10:49 am View user's profile Reply with quote Send private message
I think the problem would be in what you 're trying to do Leon, is that in order complete your script, you would have to ID each bot spawned so the script would know when to go to opening the door.

I'm not sure that the D3 engine allows for that kinda process. You can script metcens and there is also a tool within the editor that allows you to assign certain properties outside of scripting. (triggers and such) But assigning an ID or name to each bot as it is spawned? I'm not certain that, that is possible.

LEON
DBB Ace
DBB Ace

Posted Wed Sep 17, 2008 12:11 pm View user's profile Reply with quote Send private message
Hey.

I'm really new to this, so my sugestions might sound stupid.
When I make a matcen I set maximum bots to produce, so d3 at least know how many bots the matcen will generate. Then it should be possible to script ' when matcen1 has produced 20 objects, open door1'
This script wouldn't be optimal though, cus you get and complete message before you have killed the last bot, kind a odd.
I haven't find out how to script matcen at all, so I dunno about this.

Edit, Or maybe its possible to set a timer on the matcen. Matcen will be active for.....lets say 5 min. and door opens after 5 min, as well.

The Lion
DBB Ace
DBB Ace

Posted Wed Sep 17, 2008 12:33 pm View user's profile Reply with quote Send private message
There's an event generated when a matcen spawns a bot, which can give you
its object handle.

Finding out when such newly generated bots are destroyed is another matter.
I'm pretty sure it can be done, but maybe not using just DALLAS since it only
lets you tie events to pre-existing objects.

You'd have to trap all "Object destroyed" events and compare the handles to
a list of spawned bots which you are maintaining.

Kyouryuu
DBB Alumni
DBB Alumni

Posted Wed Sep 17, 2008 4:38 pm View user's profile Reply with quote Send private message
I don't have DALLAS installed, but maybe an if-then can do this? If you use an if-then, you may be able to capture the robot as the owner in the "if" clause, and then check against it in the "then" clause?

Foil
DBB Admiral
DBB Admiral

Posted Thu Sep 18, 2008 12:06 pm View user's profile Reply with quote Send private message
The Lion wrote:
I'm pretty sure it can be done, but maybe not using just DALLAS since it only
lets you tie events to pre-existing objects.

Agreed, it wouldn't be too hard to do using the D3 SDK C++ code.

But I think Leon is strictly using the DALLAS script editor... correct?

LEON
DBB Ace
DBB Ace

Posted Thu Sep 18, 2008 12:17 pm View user's profile Reply with quote Send private message

Foil wrote:
The Lion wrote:
I'm pretty sure it can be done, but maybe not using just DALLAS since it only
lets you tie events to pre-existing objects.

Agreed, it wouldn't be too hard to do using the D3 SDK C++ code.

But I think Leon is strictly using the DALLAS script editor... correct?

Correct!

WillyP
DBB Ace
DBB Ace

Posted Sun Oct 26, 2008 6:15 am View user's profile Reply with quote Send private message
I think this can be done with DALLAS: There is an event 'Matcen Spawned' where the matcen id is specified and the spawned object becomes 'IT'.

Two ways I see it could done, either count the number of spawns and trigger when the last bot spawns... would open the door with one bot alive...

or: Make the last spawn of the matcen be a non robot object, like a power up, then the script could check for a non-robot spawn. (IT is (object type robot) = false) Then the player would be left with a power up and an unlocked door.

oops... sorry for the bump... and I see LEON has already released his level. Embarassed

LEON
DBB Ace
DBB Ace

Posted Sun Oct 26, 2008 7:53 am View user's profile Reply with quote Send private message
Thanks WillyP, I'll take a look at it.

As for my level, I droped the story and made a fight only level Smile

Leon

LEON
DBB Ace
DBB Ace

Posted Sun Oct 26, 2008 1:20 pm View user's profile Reply with quote Send private message
Ok, did some testing.

when matcen [matcen ID] has spawned an [IT]
IT is an powerup
Then open door 1

I made a matcen that spawned 4 robots first then last spew was an shield powerup.

And the door open up Smile

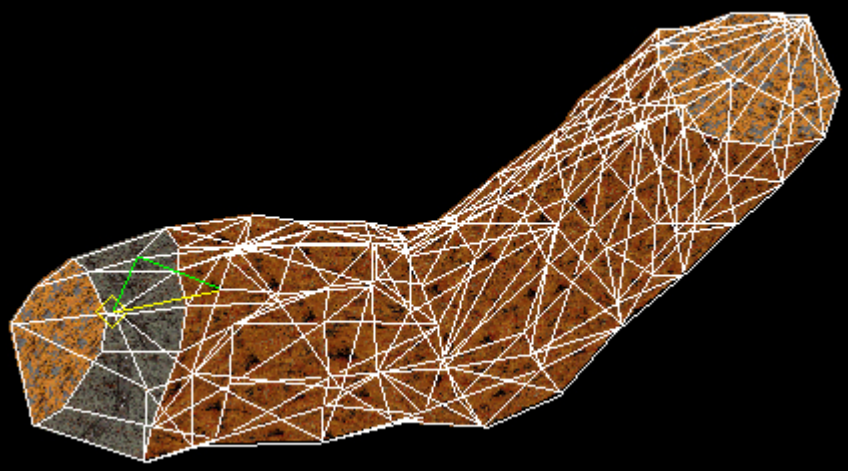
Only thing i couldnt figure was to ID the matcen, since thers no matcen identification, as far as i know.
So if i have several matcen in a level, it would be
an problem, no?

Edit: Checked some more, matcen are named 'matcen1' 'matcen2' and so forth. I might use those ID's.
But i didn't find a way to set matcen ID in dallas.

129 - Räume teilweise skalieren

Ragil Ral

Vielleicht hattet Ihr schon mal die Situation, daß der Raum im Großen und Ganzen gepaßt hat, aber bestimmte Teile davon zu groß oder zu klein geraten sind. In diesem Beispiel habe ich einen höhlenartigen Gang, dessen Ende ich ausweiten möchte:

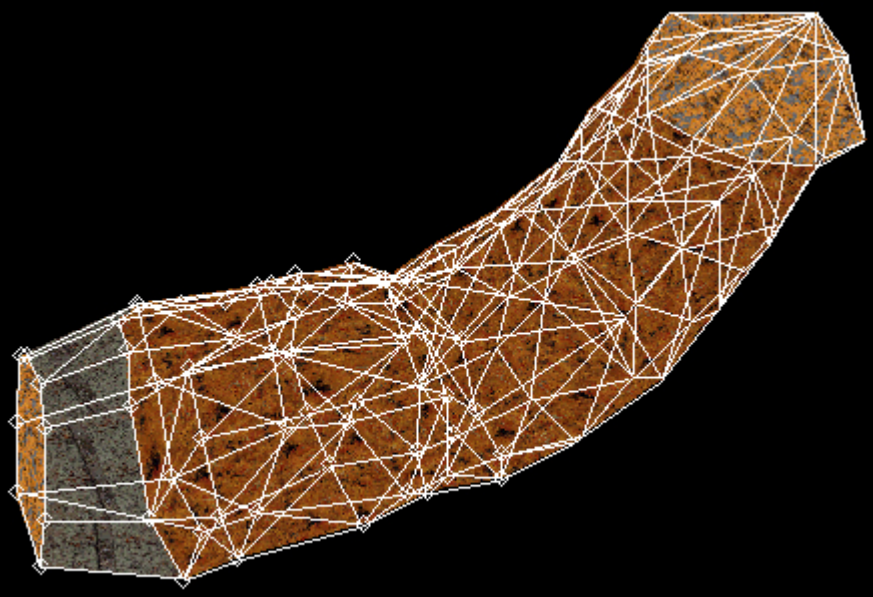


Hier kommt nun die Funktion für Face expand und contract zum Einsatz - vielleicht ein wenig überraschend... 😁

Ablauf...

Hier wird ausgenutzt, daß Face Expand und Contract JEDES Face bearbeiten, egal ob valide oder nicht. Also markiert man die Verts des Bereiches den man skalieren möchte, fügt ein Face ein und skaliert dieses. Danach gegebenenfalls zurechtrücken und die Texturen ausrichten.

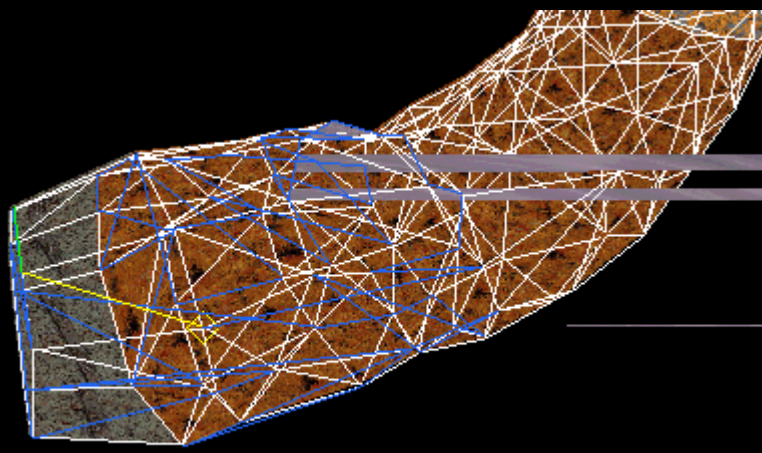
...und die Praxis



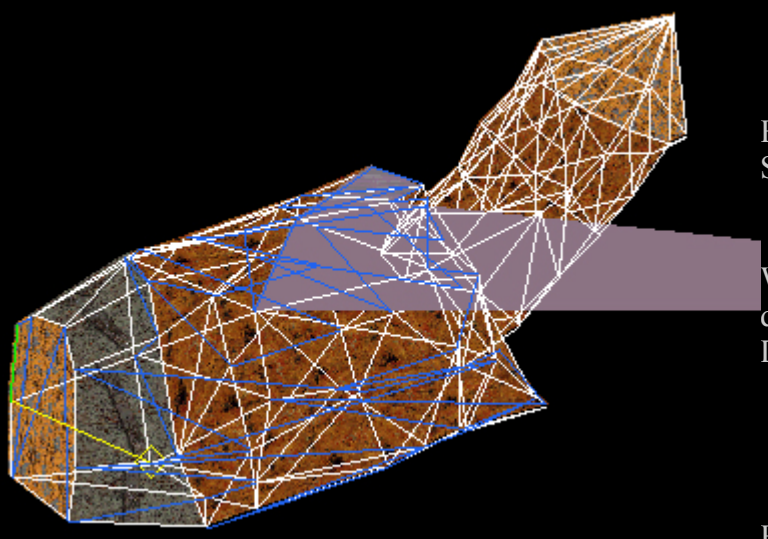
Als erstes markiert die Verts des zu skalierenden Bereiches, seit v40 kann man sich diese auch in der 3d-View anzeigen lassen:

Hier geht es sich gerade aus, es sind 52 Verts.

Danach fügt Ihr dort ein Face ein (**Shift-Ins**, unten)



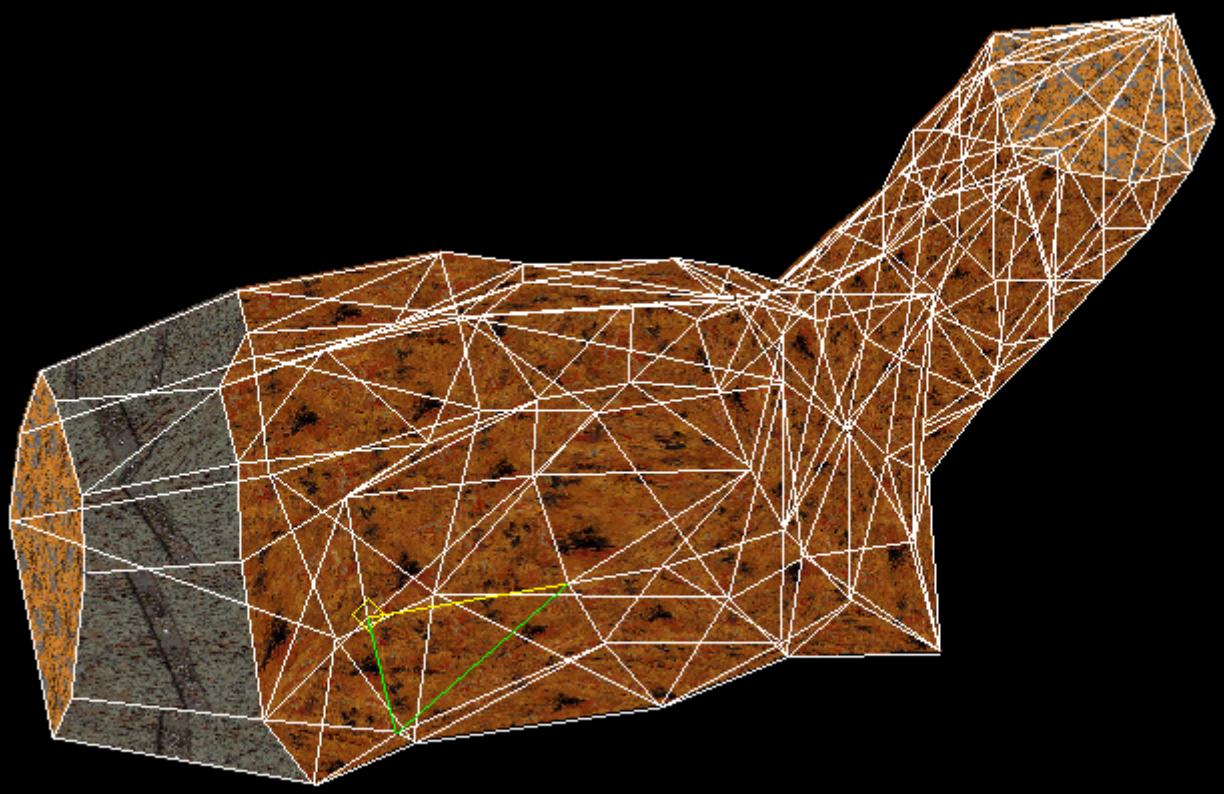
Wie Ihr an den Darstellungsfehlern sehen könnt, ist das Face alles andere als valid. Das eingefügte Face ist current und markiert.



Einige Klicks auf Face Expand später sieht die Sache dann so wie links aus.

Wechselt nun in den Face Mode. Jetzt könnt ihr den veränderten Bereich noch zurechtschieben. Danach löscht einfach das markierte Face.

Et voila...



Nun noch die Texturen ausrichten, fertig.

Die Methode hat den Vorteil daß der Face-Verband nicht zerrissen wird, was bei Einsatz der dialoggesteuerten Skalierfunktion ( ) leider passiert.

Update v40:

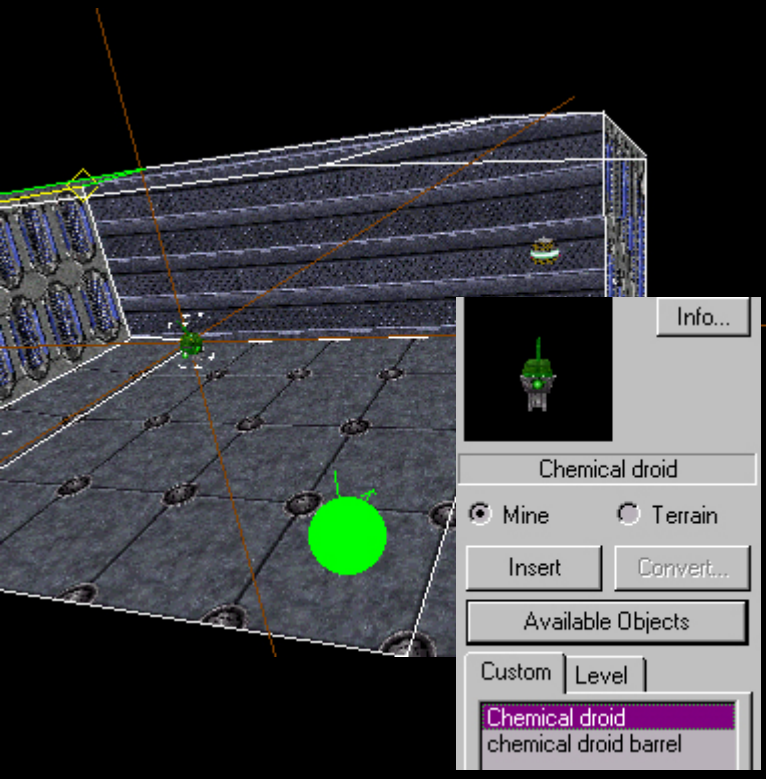
Die Skalierfunktion hatte einen Positionierungsfehler, der nun behoben ist. Das 'Auftrennen' des Face-Verbandes geschieht aber immer noch, also nicht vergessen Remove Extra Verts auszuführen wenn Ihr die dialoggesteuerte Skalierung verwendet.

130 - Zusammengesetzte Objekte inprog

Ragil Ral

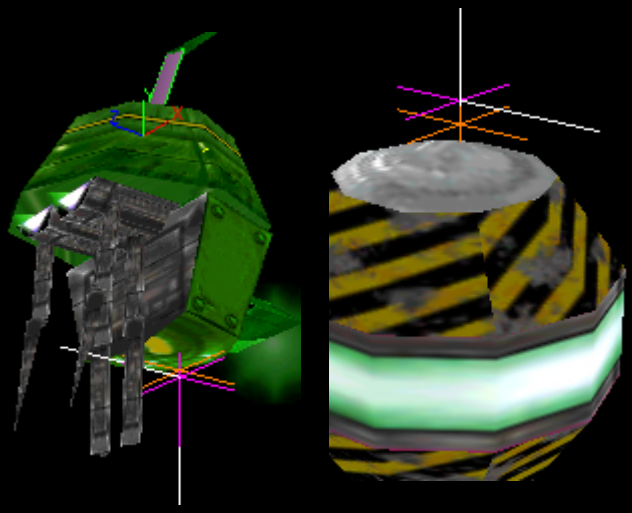
Wie Ihr spätestens aus der Sektion 089 - Ship Tutorial wißt, ist es möglich Objekte an andere anzuhängen. Die Mindestvoraussetzung dafür ist, daß beide Objekte mindestens einen ATCH haben; optional sind NATH, sie dienen zur Ausrichtung der Objekte untereinander.

Grundsätzlich ist die Sache nicht so kompliziert, doch sollte die Verwendung von DALLAS keine Schwierigkeiten mehr machen; auch OOFEdit sollte Euch vertraut sein.



In diesem Beispiel werden wir den Chemical Droid und seine Kapsel aus dem Retri-Level 2 verwenden.
Beginnt also mit einem Level, holt Euch die beiden Objekte rein und benennt diese. Hier heißt der Chemical Droid Carrier, die Kapsel Payload.

Beide Objekte besitzen jeweils einen ATCH und einen NATH:



Wenn Ihr die Objekte ins Level setzt und diesen startet, liegen sie natürlich getrennt vor; hier kommt jetzt DALLAS ins Spiel.

Baut Euch dieses Skript ein:

Script 000: Aufklauben
Owner: Level
Event: Level Start
If the following condition is met:
ALWAYS
Then perform the following actions:
DO NOTHING
Clipboard

Insert a New Action
Insert a Nested IF-THEN Clause
Insert a Nested IF-THEN-ELSE Clause
Replace with a Different Action
Delete
Cancel

DO NOTHING
Objects
Players
Doors
Rooms
Triggers
Spew
Weather
AI
Sound & Music
Timer
Level Goals
Mission
Math
User Vars
Misc
Scripts

Set object [Object] shields to [Shields]
Set object [Object] energy to [Energy]
Play object [Object] animation from frame [StartFrame] to [EndFrame], cycle time = [CycleTime],
Damage object [Object] by [DamageAmount]
Kill object [Object]; delay type = [DeathDelayType], explosion size = [DeathExplosionSize], death
Kill object [Object] with its default death
Delete object [Object]
Set object [Object] lighting distance to [Distance]
Set object [Object] lighting color = [Red],[Green],[Blue]
[Enable/Disable] gravity for object [Object]
Set movement type for [Object] to [MovementType]
Set object [Object] movement direction = <[X],[Y],[Z]>, speed = [Speed]
Attach new object of type [ChildTypeName] attachpoint [ChildPoint] to object [Parent] at attachpoin
UnAttach [AttachedObject] from an object
Object [ParentObject] drop all attached objects
Attach existing object [Child] attachpoint [ChildPoint] to object [Parent] at attachpoint [ParentPoint]

Message List
Search in Script for :
Flags: AIFlags, AIGoalFlags, CinematicFlags, DeathFlags
Enums: Activate/Deactivat, AIModeType, Axis, Can/Cannot
Objects
Doors

Parent ist der Träger (hier der Chemical Droid mit dem Namen Carrier) und Child ist das angehängte Objekt (hier die Kapsel namens Payload).

Da beide Objekte jeweils nur einen ATCH haben, paßt in diesem Beispiel der Eintrag 0 für ChildPoint und ParentPoint.

Script 000: Aufklauben
Owner: Level
Event: Level Start
If the following condition is met:
Then perform the following actions:
Attach existing object Payload attachpoint 0 to object Carrier at attachpoint 0
Child: Payload
ChildPoint: 0
Parent: Carrier
ParentPoint: 0
Clipboard

Speichert/kompiliert das Skript und baut es in Euer Level ein.

Und siehe da, der Bot trägt die Kapsel!

Wenn Ihr den Bot zerstört und Gravitation im Level habt, fällt die Kapsel dann nach unten.

Dies ist die einfachste Variante. Es gibt da aber noch mehr...

-Supertrooper: haxn und oberteil, beschreibung

-zusammenhängen mit attach new object to..., ablauf&screenies

d3jake:(zwei zusammengesetzte objekte, eins zerstörbar)
Though I'm not here often (damn school...) I haven't gotten a chance to ask about compound objects, like the Jugg or dragon thingie. I'm trying to create a light that is a half of a cylnder that is kinda framed and I was wonder if I could make that into two objects so I can have the light portion blow leaving the "holder" behind. Or should I mess with it as one object?

Jon the Great:(ATCH und DALLAS zum zusammensetzen)
You use attach points and then Dallas scripting to put them together.

Kyouryuu (s.o.)
Yeah, as far as I know, you can construct them as individual objects with individual behaviors, but then use DALLAS scripting to "dock" them together at defined mount points.

Darkside Heartless: (Objekte nach Zerstörung ersetzen, DALLAS)
Isn't it also possible to script an object to be replaced by another after it gets destroyed?
Kyouryuu:(s.o., Ausführung)
Yes.
First, somewhere else in the level, you need to have the object you want to replace it with.
In DALLAS, what you'd do is on the death of that object, copy the object's location to the clipboard. Then, move the replacement object to the position in the clipboard.

Anhang – Nützliche Infos

Liste aller D3 Tools		=Legion=	456
House of Mirrors	Bilder aus dem Abyss		461
Ingame-Hilfen fürs Testen	Einfache Mittel um den Level ‚in action' zu überprüfen	Ragil Ral	462
D3-Edit Limits	Grenzen beim Bauen	(LL)Atan	464
Anwendung von D3 Levelinfo	Andwendung des Spezialwerkzeuges von Atan	Atan	465
Dokumentation der Türen	Alle Türen von D3	Ragil Ral	468
Texturenkatalog		RagilRal	516
D3Edit Tastenkombis	Hotkeys für den D3Edit - alle auf einer A4-Seite		518

Toc

Liste aller D3 Tools

=Legion=

Ich habe mich bemüht, möglichst alle vorhandenen Tools hier aufzulisten, bei Verfügbarkeit mehrerer Versionen habe ich die Neuere genommen. Ältere Tools habe ich weggelassen, sofern für diese ein (besserer) Ersatz vorhanden ist. Das Ganze ist thematisch gruppiert.

Die Liste ist ein Kompilat der Tools, die auf folgenden Seiten verfügbar sind:

http://archives.descent.cx/d3/d3tools.html
http://descent3fischlein.de/tools/tools.php4
http://www.planetdescent.com/site/files/d3/utilities/development.asp

Der Stand der Liste ist März 2008.

Tabellenschlüssel...

Verwendungszweck	`pfad\zum\tool\`	
Name	Beschreibung	Autor
`Dateiname`		Programm-Version
Art		Größe in kB

wobei:

Art	Beschreibung
packed	Einfach irgenwohin entpacken, kann meist sofort verwendet werden; readme's lesen!
installer	erfordert eine Installation zur ordnungsgemäßen Funktion.
weder noch	spezial, hier den Anweisungen der readme's unbedingt folgen

Zunächst mal unter `tools\`

D3 Edit	Der echte, wahre, einzige D3-Editor!	Outrage / (LL)Atan
`D3EditAV40.zip`		1.1 [AV40]
packed		1465 kB

Skript/Codebezogen	tools\coding\	
Ming C-Compiler	damit kompiliert man die DALLAS-Skripte.	
egcs-1.1.2-mingw32.zip		
packed, siehe tuts		6200,0 kB
SDK-TOOL v0.01 beta	Das Tool ermöglicht es unter LINUX Gamemodes zu übersetzen. Es installiert das SDK, einen kompatiblen Compiler und wird in den nächsten Versionen noch verbessert, mit mehr Funktionen	D.Cent
sdk-tool_v0.01_beta.zip		0.01
packed, siehe tuts		30,7 kB
SDK 1.4 (Linux)	SDK braucht man um Skripten in seine Levels einzubauen.	Jeff Slutter
d3-sdk_v14.tar.gz		1.4
packed, siehe tuts		573,9 kB
SDK 1.4 (Windows)	SDK braucht man um Skripten in seine Levels einzubauen.	Jeff Slutter
d3-sdk14.zip		1.4
packed, siehe tuts		634,1 kB
HOG-OGF	Eine Textdatei mit den Daten aller .ogf-Dateien.	?
hog-ogf.zip		?
packed		18,5 kB

Konverter	tools\converter\	
2 ORF	Konvertiert .stl, .dxf und .ase nach .orf	Magus
2orf_v091.zip		0.91
packed		119,8 kB
3DSGen	Konvertiert OOF-Dateien ins 3DS-Format. Grundsätzlich wird alles was in einer .oof gespeichert ist konvertiert (Texturen, Texture-Mapping, Hierarchie, Special Points, Keyframe-Daten...) Enthält auch das MAXOOFUtility. Cmd-Line-Tool	SuperSheep
3dsgen.zip		1.0
packed		88,6 kB
DXF2ORF	DXF2ORF is a utility for converting various Drawing eXchange Files to Outrage Room Files used by the Descent3 Level Editor, D3Edit.	Cougar
dxf2orf_vb.zip		Beta
packed		11,4 kB
MS3D2ORF	Import rooms created in Milkshape to D3Edit's ORF format. Very handy for those of you who don't like D3Edit's interface. Cmd-Line-Tool	Descenterace
ORFTOOL.zip		0.5 beta
packed		57,3 kB
OOF to 3DS	With this program you can convert the geometry information stored in OOF-files (Outrage Object Files) to 3DS-files (3D-Studio). Cmd-Line-Tool, erfordert installiertes 3dmax	Tycoon
oofto3ds.zip		1.0
packed		44,7 kB
OOF2ORF	Wandelt OOF\'s (Outrage Object Format) zu ORF\'s (Outrage Room Format) um. Das schöne daran ist das man auch die Texturen mit Entpacken kann, Erhält außerdem auch alle Sub-Objekte, UV-Mappings und den Rest aus der .oof	Robert Piontek
oof2orf_vb123.zip		beta 1.23
installer		1607,8 kB
PD3 to OOFGen Converter for 3DS Max	3dMax Plugins um nach .oof zu wandeln. erfordert installiertes 3dmax. Plugins für neuere und ältere versionen	Outrage
p3d-oof-utilities-MaxAllVer.zip		3
plugins		534,7 kB
X2orf	Konvertiert x exports von blender 1.42 in .orf files. Cmd-Line-Tool	Ouch
x2orf.zip		0.9.0a :)
packed		27,2 kB

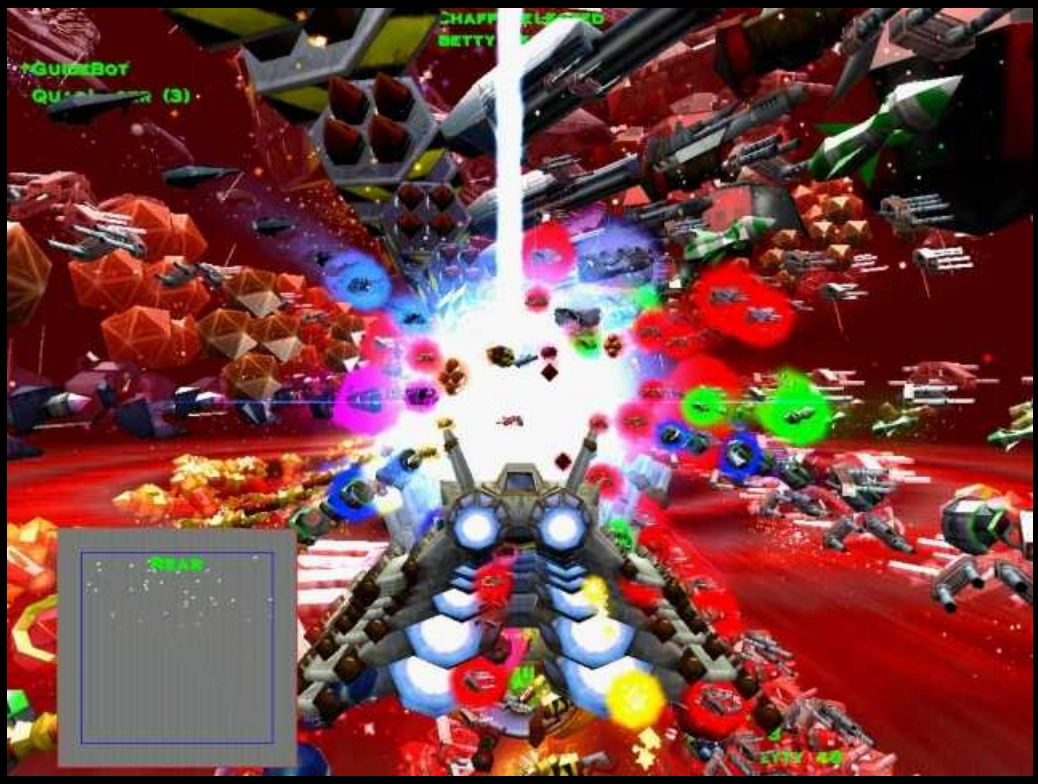
GAM-Datei	tools\gamfile\	
Descent 3 Control Panel	GAM-Dateien einsehen und editieren	Dark Knight
d3controlpanel_v102.zip		1.02
packed		350,3 kB
Gamtool	Editiert .gam Dateien. Das Tool benutzt man um Eigenschaften von Waffen, Schiffen und eigenen Texturen zu verändern oder zu erstellen. (und noch mehr)	Aldel
gamtool0611.zip		0.611
packed		163,3 kB
Hog Explorer	HOG2-Dateien einsehen und editieren	Gwar
hogexplorer_v05.zip		0.5
packed		28,0 kB
Hog View 32	Extrahiert Dateien aus .MN3s und .HOGs	Mike Menefee
hogview32_v3b.zip		1.0 Beta 3 build10
installer		488,7 kB
HOG2 Workshop	Ein sehr gutes Programm zum Editieren und Entpacken von Dateien aus D3 HOG2 Dateien.	Dark Knight
hog2workshop_v10.zip		1.0
packed		165,3 kB
Lex-Hogedit	Extrahiert und erstellt Descent3 HOG-Files	Lex
dh2e11.zip		0.0.0.0
packed		1807,9 kB
Packafied 0.5 (PR's)	Packafied ist ein Package-File-Editor. Beispiele sind Doom WAD-Dateien, Quake PAK-Dateien und eben Descent HOG-Dateien.	Peter Runge
packafied_v05.zip		0.5
packed java		295,8 kB
Table Editor 0.4 (PR's)	Mit diesem Programm editiert man GAM-Dateien; Waffen, Schiffe, Objekte, etc...	Peter Runge
tableedit_v04.zip		0.4
packed java		147,4 kB
Tycoons Hog 2 Viewer	Zeigt den Inhalt von D3's HOG2-Dateien an. Kein Editieren!	Tycoon
hogedit.zip		1.1
installer		210,5 kB

Objekte	tools\objects\	
alOOF 0.8	alOOF ist ein Editor für die Outrage Object Format-Dateien (OOF) von Descent 3.	Aldel
aloof08.zip		0.8
packed		110,8 kB
D3 View	Zeigt Descent 3 Polymodelle an.	Garry Knudson
d3viewer.zip		0.1
packed		40,2 kB
Descent 3 Tool	Dieses Programm kann die Objekte anzeigen, die mit dem Descent3 benutzt werden. Man kann damit auch eigene Objekte erstellen und diese manipulieren. Dieses Tool hat Import-/Exportfunktionen für OOF	Garry Knudson
d3tool_vb14.zip		beta 0.14
packed		73,9 kB
Model View 32	Alle Descent 3 Polymodelle anzeigen und editieren.	Heiko Herrmann
modv32b5.exe		1.0 Beta 05 Build 28
installer		616,6 kB
OOF View	Ansicht der Descent3 3d .oof-Modelle.	Gwar
oofview_v040.zip		0.4
packed		44,6 kB
Oofedit1.518	Ein Super Tool zum erstellen und Animieren von eigenen Objekten! Falls Ihr irgendeine ActiveX-Fehlermeldung bekommt, müßt ihr folgendes machen, geht ins Programmverzeichnis, öffnet dort eine Dos-Box und gebt dort ein: regsvr32 SSubTmr.dll	SuperSheep
oofedit1518.zip		1.518
installer		2304,1 kB
tExtractor 3.0	Maya-Plugin, das es erlaubt die UV-Koordinaten eines ausgewählten Polygons abzuwickeln und die Resultate als Grafikdatei auszugeben. Diese kann dann als perfekt passende Textur verwendet werden	Outrage
textractor30.zip		3.0
maya plugin		18,3 kB

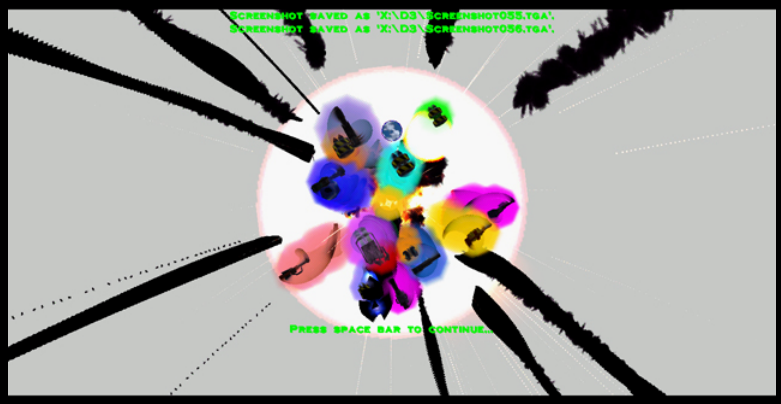
Sound	tools\sound\	
Descent3 Audio Taunt Manager	Konvertiert Wave-Dateien zum sofortigen Gebrauch in Descent3.	Lex
tauntmngr.zip		1
packed		260,6 kB
OSF Play	Tonnenweise OSF Taunts im "\custom\sounds" Verzeichnis? Hiermit kannst Du diese anhören. Benötigt wird "MusicTester", ist bei D3Edit dabei.	Lex
osfplay.zip		1
packed		210,7 kB
Sound Composer + Plugins	Ein neues Sound Tool. Der Composer wird, wenn er sich denn bewährt, den Taunt Manager ablösen. Mit neuem Plugin-Konzept für Erweiterungen. Inklusive Plugins: Sound Effects 1.1, Sound Recorder 1.1	Lex
d3sc+Plugins.zip		Alpha 1.0 Build 38
packed		647,6 kB

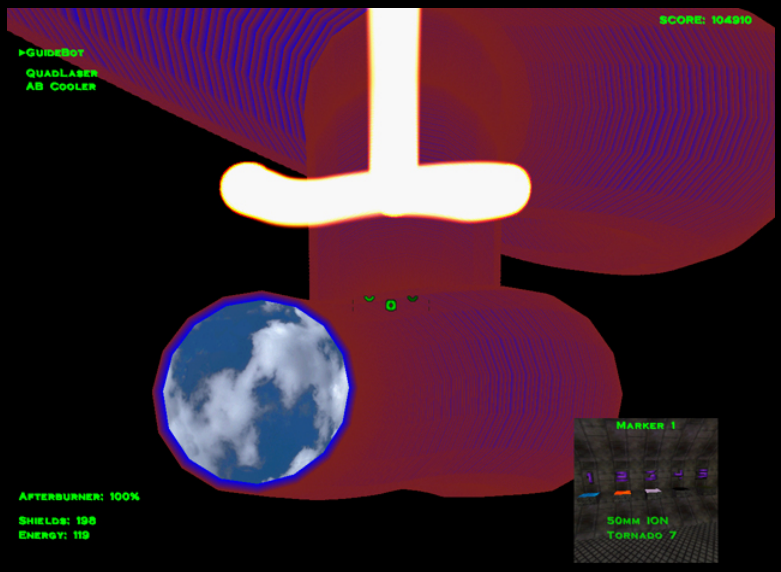
Texturen	tools\textures\	
Animake animake_vb0201.zip installer	Ein Tool mit dem man Animierte Texturen erstellen kann.	DCrazy 0.201 89,2 kB
D3 Image Tool d3it103.zip packed	Konvertiert TGA Dateien zu OGF Dateien. Ist allerdings besser zu handhaben als das OGFTOOL.	SuperSheep 1.03 122,7 kB
GIMP OGF/OAF Plug-in Gimpp-ogfoafPlugin.zip gimp plugin	Dieses Plugin erlaubt es, direkt .ogf und .oaf-Dateien zu lesen/schreiben, sodaß man nicht mit einem Konvertertool arbeiten muß. Den Inhalt der Datei einfach ins Gimp-Plugin-Verzeichnis entpacken	? 0.11 9,7 kB
OAF Editor (PR's) oafedit.zip packed java	OAF Editor ist ein Descent 3 OAF (Outrage Animated File) Editor.	Peter Runge 1 151,6 kB
Pic View 32 picview_vb01.zip packed	Konvertiert D1 & D2 Texturen in Descent 3 Texturen.	Chris Becker 1.0 beta 01 build 03 459,9 kB
Tycoons Logo Editor tyclogoed_v20-p.zip installer	Multiplayer-Schiffslogos erstellen und editieren.	Tycoon 2.0 2073,3 kB

Sonstige Tools	tools\the_rest\	
AutoEdit autoedit.zip installer	Ein kleines Tool das es erleichtert mn3 Dateien zu erstellen. Es ist nützlich wenn man Levels testet und sie immer wieder neu Packen muss.	[DCG]Roadrunner 1.0.4 1563,4 kB
BRIEFS32 briefs32.zip packed	Damit kann man Briefings erstellen. Mit Tutorial!	Topher 1.0 Beta 01(Build05) 856,3 kB
D3 Quick Test 2.1 D3Quicktest21.zip packed	Quicktest als Standalone-App. Praktisch, wenn man nicht am Level, aber in .gam's Veränderungen vornimmt, so muß nicht jedesmal D3Edit gestartet werden	Anthony Galica 2.1 40,6 kB
D3Cleaner d3cleanerfull.zip packed	Mit diesen Tool kann man D3 schneller starten. Man kann einen Piloten konfigurieren, dessen Sounds und Grafiken, und man kann noch Befehlzeilen-Schalter setzen	Roadrunner 0.0.78 97,9 kB
D3MInstaller 1.0 d3minst.zip packed	Ein All-In-One-Setup-Programm, das die Installation von D3-Mods unterstützt und erleichtert. Einfach eine Custom-Datendatei, Eula-Datei und die Mod-Hauptdatei (.d3m) angeben, und Du hast einen Installer!	DCrazy 1.0 13,2 kB
Descent3 Mission Browser d3msninfo.zip packed	Zeigt Informationen wie Autor, URL usw. der Missions-Dateien. Ist auch sehr nützlich für Server OP's, denn das Prog. hat auch eine Extract Funktion, mit der man die OOF's entpacken kann.	Lex 1.0 215,2 kB
Unstrung UnStrung03.zip installer	Damit editiert man die 'string'-files; HUD-Nachrichten oder Anderes.	Josh Allen 0.3 1718,0 kB


GUIDEBOT
REAR

NETWORK LOSS
NETWORK PING
QUADLASER
MATTHEW
KARX
REAR


PRESS SPACE BAR TO CONTINUE...


SCORE: 104910
GUIDEBOT
QUADLASER
AB COOLER
MARKER 1
AFTERBURNER: 100%
SHIELDS: 198
ENERGY: 119
50MM ION
TORNADO 7

REAR
MARKER 1
GUIDED

...zurück

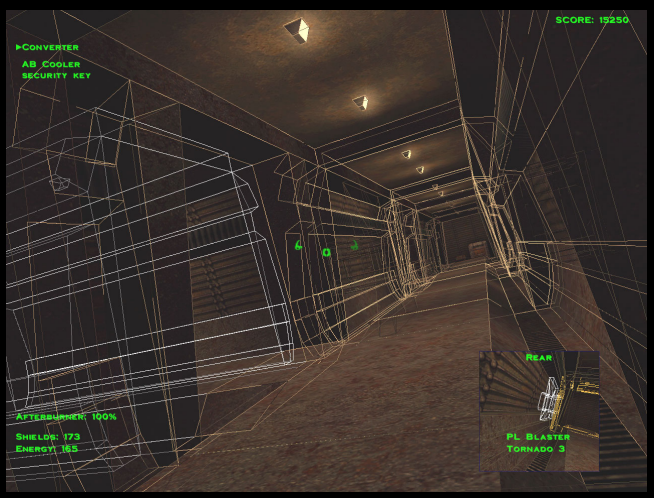
Ingame-Hilfen fürs Testen

Ragil Ral

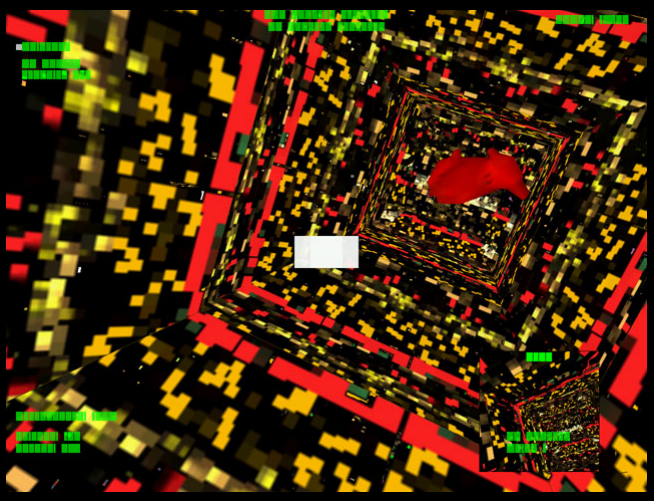
Es gibt ‚Cheats', jedenfalls werden sie so verwendet, welche das Entwickeln & Testen ein wenig erleichtern. Einfach wie einen Cheat ingame eingeben, hier eine Aufstellung:

`framelength || frametime || frametimer`
Zeigt Dir die aktuelle Bildwiederholrate. Wenn es beim Spiel mal bei der Anzeige hakt, kannst Du hiermit recht einfach die Stellen finden, die noch optimiert werden müssen.

`renderstat`
Zeigt Dir die zur Zeit am Bildschirm angezeigten Vertices und Polygone. Allerdings die, welche die Grafikkarte errechnet, und nicht diejenigen aus dem D3Edit!

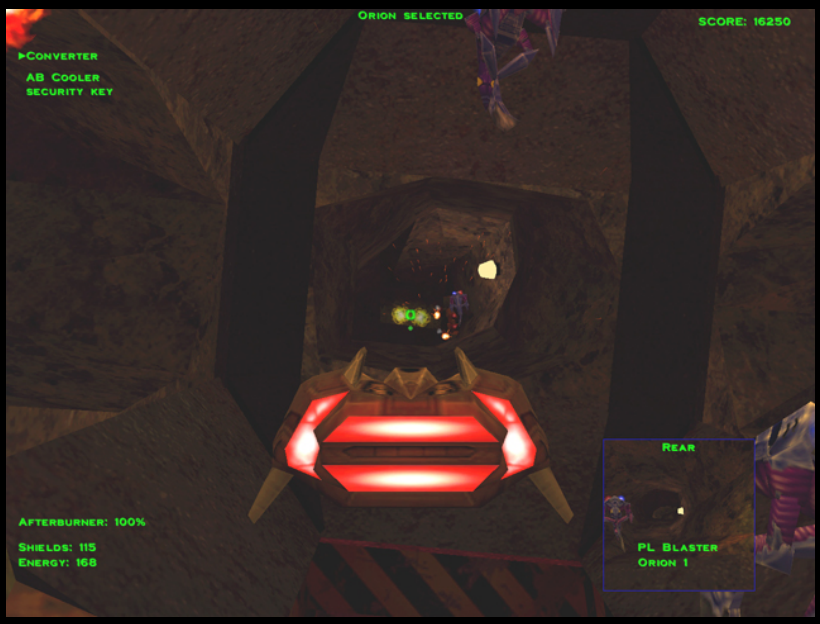


`outlinem`
Wenn Du dein Level testest, siehst du ihn im texturiertem Drahtmodel, etwa drei Räume weit. Terrain und Objekte werden allerdings nicht geoutlined



`weirdtexture || shananigans`
Würfelt die Texturen durcheinander... vielleicht ein Easteregg der Entwickler?

Halte die ^-Taste (Zirkumflex-Akzent) gedrückt und schieß eine Flare auf ein Face: Du bekommst den Raum und die Nummer angezeigt; hilfreich beim Ausrichten der Texturen.



byebyemonkey
Chase View – man sieht sich aus einer Position hinter- und leicht oberhalb seines Pyros.

Beim Singleplayer-Entwickeln können auch die hier hilfreich sein, obwohl's ‚echte' Cheats sind:

moreclang
zu einem Level springen; einfach dann die Nummer eingeben.

Dann wären da noch

treesquid	Komplette Karte
deadofnight	entfernt alle Bots aus dem Level
testicus	Cloak

... es gäbe noch einen für Unverwundbarkeit, aber den verrat' ich jetzt nicht :]

D3-Edit Limits

(LL)Atan

v1.0

Damit immer alles gut geht, hat Atan diese Liste herausgegeben. Diese Limits sollte man ernst nehmen und nicht überschreiten, wie ich aus eigener Erfahrung weiß ☺

Mine:

MAX_LEVELS_PER_MISSION	30	
MAX_ROOMS	400	// 0-399 (Mine rooms)
MAX_PALETTE_ROOMS	50	// max number of loaded rooms (400-449)
MAX_OBJECTS	1500	// total number of objects in world
MAX_POLY_MODELS	1000	// total number of POLY_MODELS in world
MAX_ACTIVE_DOORWAYS	30	// specific doors inside mine - noch unklar
MAX_DOORS	60	// predefined Doors useable by Editor
MAX_TRIGGERS	100	
MAX_PATHS	100	
MAX_PATH_PORTALS	40	
MAX_GAME_PATHS	300	
MAX_NODES_PER_PATH	100	
MAX_GOAL_ITEMS	12	
MAX_LEVEL_GOALS	32	
MAX_GOAL_LISTS	4	
MAX_WAYPOINTS	25	
MAX_SOUNDS	1000	

Rooms:

MAX_REGIONS_PER_ROOM	200	// Facestructures, not connected to room shell
MAX_FACES_PER_ROOM	3000	
MAX_VERTS_PER_ROOM	10000	
MAX_BNODES_PER_ROOM	127	

Faces:

MAX_VERTS_PER_FACE	64

Textures:

MAX_BITMAPS	5000	// Mine
MAX_TEXTURES	2600	// Room
MAX_LIGHTMAP_TEXTURES	60	// Mine
MAX_VCLIPS	200	// OAF
VCLIP_MAX_FRAMES	50	// oaf-Frames
MAX_BUMPMAPS	500	//
MAX_FORCE_FIELD_BOUNCE_TEXTURES	3	// TF_FORCEFIELD
MAX_SPECIAL_FACES	13000	// TF_METAL + TF_MARBLE | TF_PLASTIC

Objects:

MAX_MODEL_TEXTURES	35
MAX_POLYGON_VECS	2500
MAX_DETAIL_LEVELS	3
MAX_PROP_LEN	256
MAX_NAME_LEN	32
MAX_GROUND_PLANES_PER_MODEL	10
MAX_GUNS_PER_MODEL	64
MAX_SUBOBJECTS	30
MAX_POINTS_PER_SUBOBJECT	300
MAX_WB_GUNPOINTS	8
MAX_WB_FIRING_MASKS	8
MAX_WB_TURRETS	8

Anwendung von D3 Levelinfo

Atan

Version BETA 1.3

Neu seit 1.2:
Mit - auf dem Nummernblock wird zum vorher gehenden gültigen Raum gesprungen.
Mit + auf dem Nummernblock wird zum nächsten gültigen Raum gesprungen.
Sprung im Terrain entfernt
Sprung in Türen entfernt

Neu seit 1.0:
Mit 'Komma' auf dem Nummernblock wird zum vorher gehenden gültigen Raum gesprungen.

Neu seit 0.9:
Mit 'Enter' auf dem Nummernblock wird zum nächsten gültigen Raum gesprungen.

Neu seit 0.8:
Freeze Bug behoben D3 hängte sich auf wenn man von Internen in externe Bereiche flog.

Neue Checks auf:
Nonplanar Faces
Flipped faces
Missing faces
Unused verts
double verts
eingebaut
und eine Visuelle Unterstützung zum leichteren Auffinden der Fehler eingefügt.

Menu:
Nummernblock / aktiviert/deaktiviert den erweiterten Checkmodus. In Räumen mit vielen Faces verlangsamt dieser Check die D3 Engine.

Was ist D3 LevelInfo?

LevelInfo soll ein kleiner Helfer sein mit dem man seine D3 Level untersuchen kann, oder besser gesagt, mit dem man Informationen während des Fluges durch den Level erhalten kann. Es prüft die bekannten Limits, zeigt die Raum-Namen und was für ein Raum es ist den man grad durchfliegt, auch welche Flags gesetzt sind. Das kann für Entropy Level hilfreich sein.
Darum ist es auch mächtiger als Renderstat.

Dieser kleine Mod muss im multiplayer mode aktiviert werden, dafür muss man ein paar Kleinigkeiten vorbereiten:

A)

dieser Schritt muss einmal ausgeführt werden:
die Datei `check.d3m` in den Descent3/netgames Ordner kopieren.

B)

Für jeden Level der mit D3 LevelInfo gescheckt werden soll:

Seit den Versionen > 0.5 sollte es auch ohne den Step B funktionieren! Probiere es aus. Falls es nicht geht hier bitte weiter lesen:

Öffne den Level in D3Edit, starte Quicktest, und suche den Keywords - Bereich. Markiere anarchy, auch in Singleplayer-Levels die geprüft werden sollen!

Füge das keyword : `CHECK` ein und speichere den Level wieder.

C)

** !!! Kein Grund um Online zu gehen wenn man LevelInfo benutzen will !!! **

starte D3,
wähle Mehrspieler,
wähle DIRECT TCP/IP,
wähle Neues Spiel starten.

Rechts befindet sich : GAME TYPE,
CHECK auswählen,
Sollte CHECK dort nicht zu sehen sein hast Du vielleich Step A vergessen oder Du mußt erst die Mod-Typen durchscrollen um den Mod zu sehen.

Links nun die zu testende Version auswählen.

D)

Das Spiel starten

Wenn der Level nicht mit dem Mod Kompatible ist wird Descent3 eine Nachricht ausgeben. eventuell hast Du das Keyword `CHECK` nicht eingetragen, oder die Datei `check.d3m` nicht in den Descent3/netgames Ordner kopiert.

Wenn aber alles gut gegangen ist, dasnn solltest Du dich nun im Level befinden, ignoriere 'waiting for players' und starte das Spiel.

Auf der oberen rechten Bildschirmseite solltest Du das 'Key menu' sehen.

Einfg: Zeigt Raum Infos
Pos1: Zeigt Region Infos
BildHoch: Zeigt Special Room Infos
Entf: Font Grösse umschalten
Ende: Zeigt Versions Nummer
BildRunter: Zeigt/Vesteckt das 'key-menu'

Drücke '**Einfg**'

Nun sollten Informationen über den aktuellen Raum auf der linken Bildschirmseite sichtbar werden.

Fliege durch den Level,
die Informationen ändern sich sobald man in einen Raum fliegt.
Die Objekt Informationen ändern sich wenn man z.B. ein Objekt aufsammelt.

Wenn ein Problem erkannt wird, wird eine Meldung ausgegeben.

Weil der Region Check sehr langsam arbeitet, kann man ihn mit **Pos1** Ein- und Ausschalten.

Die Fontgrösse kann mit '**Entf**' umgeschaltet werden.

'**Ende**' zeigt die Versions Nummer des Mods

Wenn Du Informationen über einen Entropy-Level sehen möchtest, drücke '**BildHoch**' und es werden Rechts Unten dann die Infos ausgegeben.

Mit '**PageDown**' wird das 'Key-Menu' Ein- und Ausgeschaltet.

Mit '**/**' auf dem Nummernblock werden die erweiterten Überprüfungen und visuellen Effekte ein-/aus- geschaltet

Mit '**Enter**' auf dem Nummernblock wird zum nächsten gültigen Raum gesprungen
Mit '**,**' auf dem Nummernblock wird zum vorher gehenden gültigen Raum gesprungen

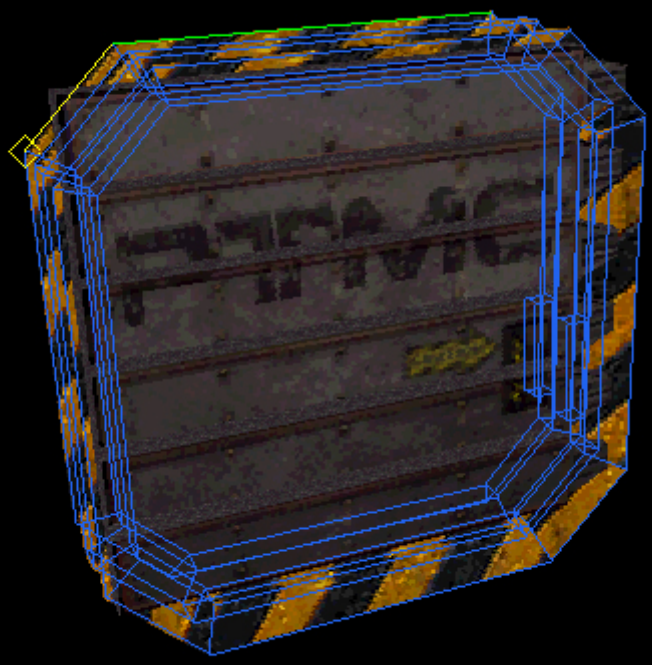
Dieser kleine Mod ist bei meinen Versuchen entstanden, benutze es auf eigenes Risiko.
Keine Garantie für korrekte Ergebnisse.
Ich übernehme keinen Support, evtl. erweitere aber ich das Tool von Zeit zu Zeit

Anmerkung, dieser Mod ersetzt nicht das sorgfältige Arbeiten in D3Edit.
Dort kann jeder Fehler bereits im Vorfeld gefunden werden.

Dokumentation der Türen in Descent 3

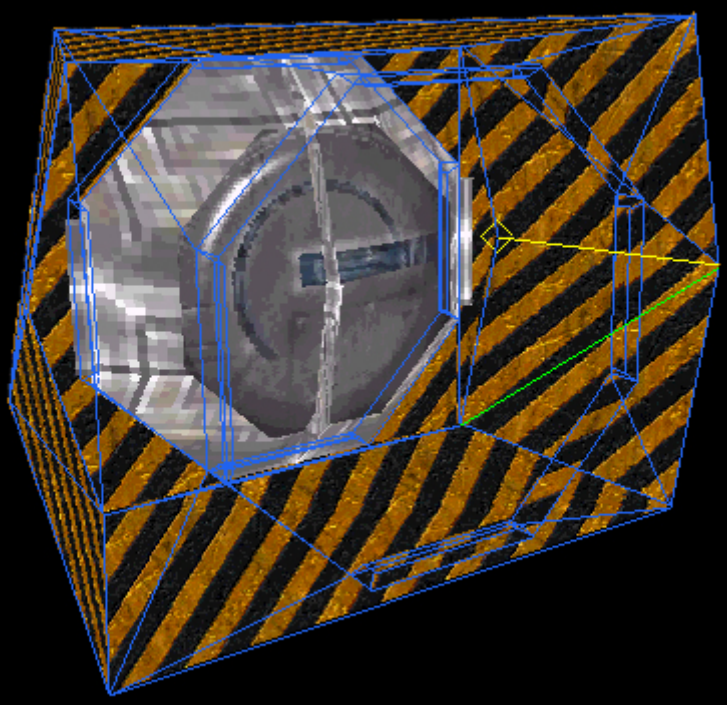
Ragil Ral

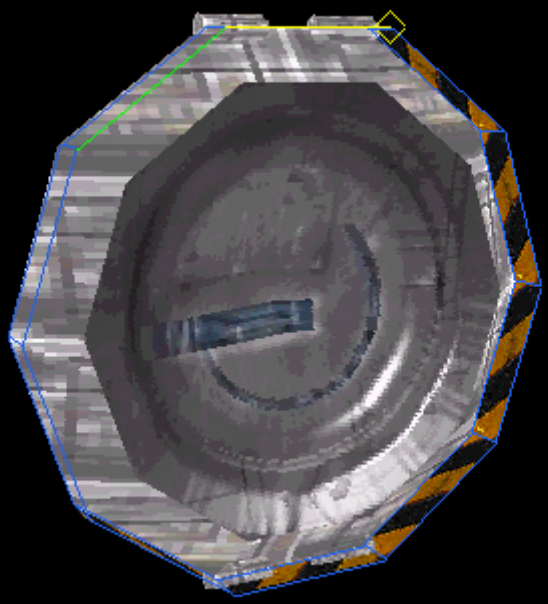
Türname:
PTMC Industrial 2
Größe ca:
20x18 Units
Verts des Frontfaces: 6
Hat Region-Einbauten:
Nein
Bugs:
7 duplicate Verts, 6 Unused Verts, 4 T-Joints
Sonstiges:
Die T-Joints befinden sich an den Faces, die beim Einbau der Tür zu Portalen werden.
Reparatur=Muß!
Viele Faces der Tür sind trianguliert, ein Vereinigen mit **Strg-Shift-LKlick** ist empfohlen.



Türname:
PTMC Industrial 4
Größe ca:
20x20 Units
Verts des Frontfaces: 8
Hat Region-Einbauten:
Nein
Bugs:
8 Duplicate Verts
Sonstiges:
-

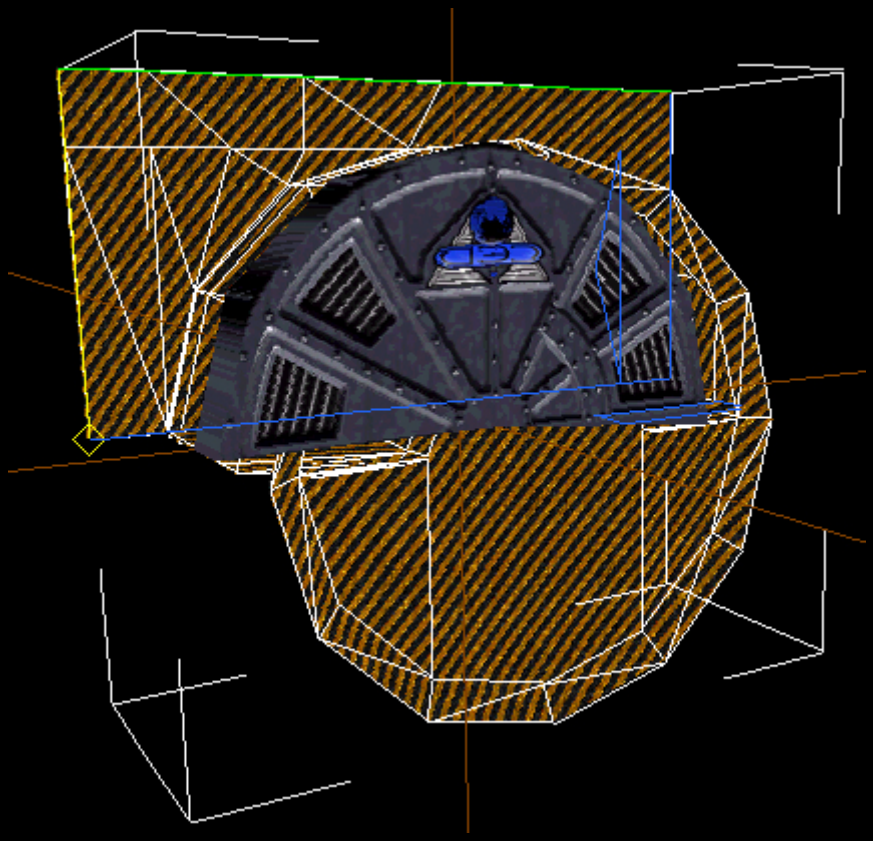
Türname:
SEANS STEAMVENT DOOR
Größe ca:
50x35 Units, Frontface 30x30
Verts des Frontfaces: 8
Hat Region-Einbauten:
Nein
Bugs:
8 duplicate Verts, 8 Unused Verts
Sonstiges:
Wurde die Tür einmal durchflogen, wird ein Sound abgespielt. Die Tür eignet sich gut als Verbindung zum Terrain.

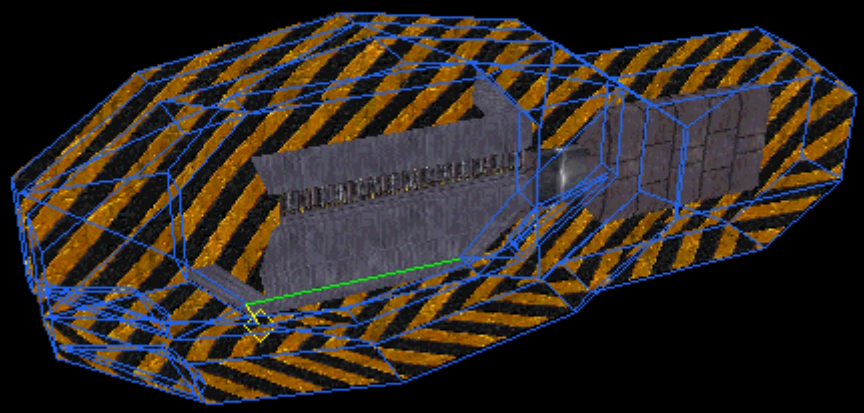




Türname:
SEANS HEATSINK DOOR
Größe ca:
Durcmesser 35 Units
Verts des Frontfaces: 10
Hat Region-Einbauten:
Nein
Bugs:
10 Duplicate/Unused Verts
Sonstiges:
Ist die SEANS STEAMVENT DOOR von der anderen Seite aus.

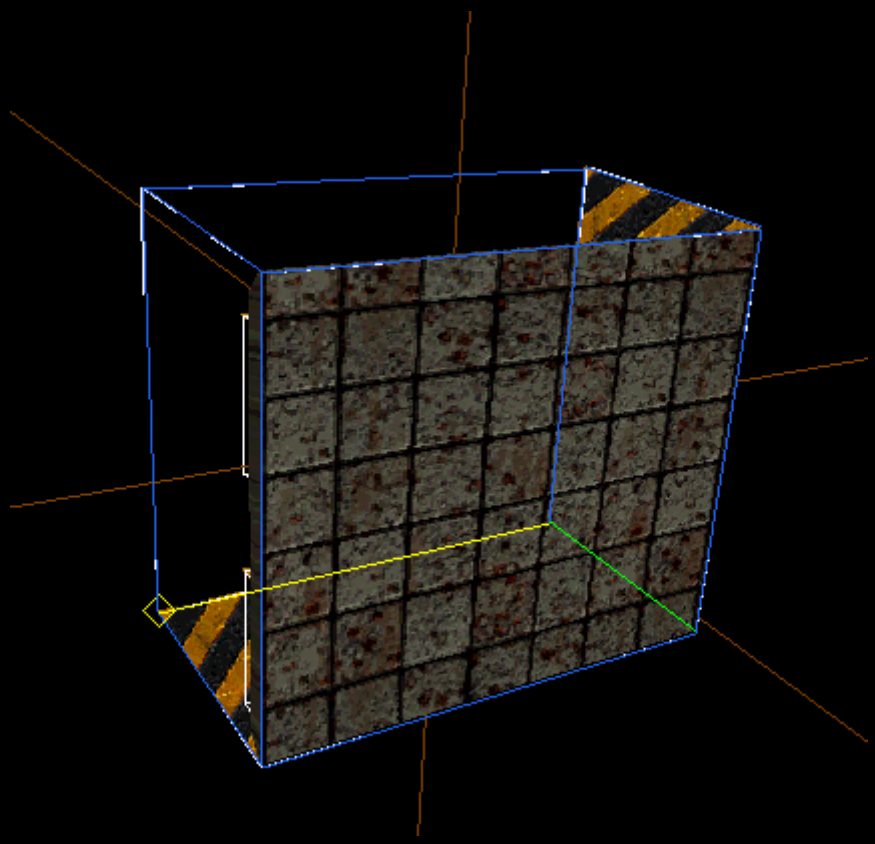
Türname:
DAN'S VAULT DOOR
Größe ca:
200x170 Units
Verts des Frontfaces: 12, Durchmesser 140
Hat Region-Einbauten:
Ja
Bugs:
13 duplicate Verts, 12 Unused Verts, 4 T-Joints
Sonstiges:
Die T-Joints befinden sich an den Faces, die beim Einbau der Tür zu Portalen werden. Reparatur=Muß! Viele Faces der Tür sind trianguliert, ein Vereinigen mit **Strg-Shift-LKlick** ist empfohlen. Die Tür ist ein wenig grob, der Reparatur- und Optimieraufwand etwas höher.

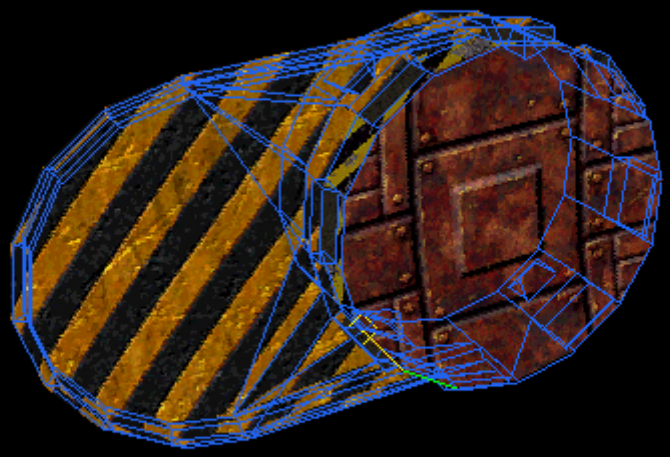




Türname:
CED 1
Größe ca:
80x30 Units, FF 60x30
Verts des Frontfaces: 10
Hat Region-Einbauten: Ja
Bugs:
10 Duplicate/Unused Verts, 2 T-Joints
Sonstiges:
-

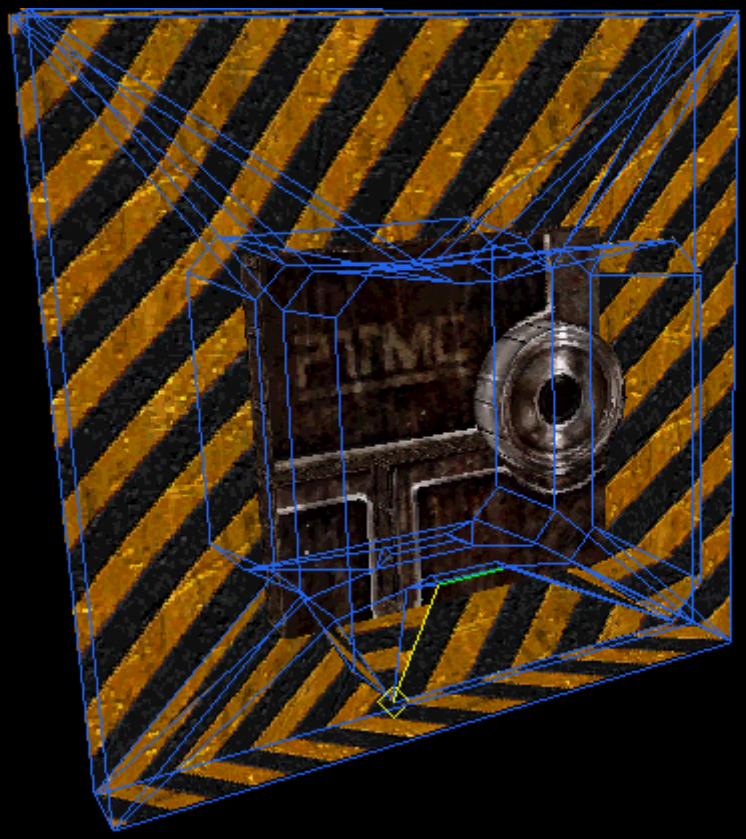
Türname:
SEAN'S TRAPDOOR
Größe ca:
28x26 Units
Verts des Frontfaces: 4
Hat Region-Einbauten:
Ja
Bugs:
4 duplicate / Unused Verts
Sonstiges:
Das Tür-Objekt hat 4 Concave Faces. Reparatur empfohlen.

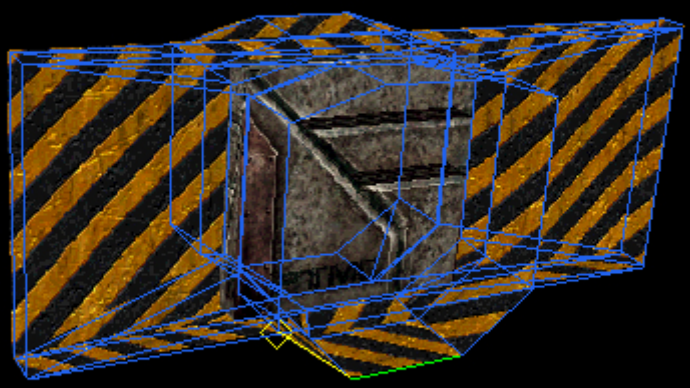




Türname:
LUKE'S SECRET DOOR
Größe ca:
35x20 Units, FF Durchmesser 19
Verts des Frontfaces: 16
Hat Region-Einbauten: Ja
Bugs: 16 Duplicate / Unused Verts, 3 T-Joints
Sonstiges:
Die Tür sollte optimiert werden.

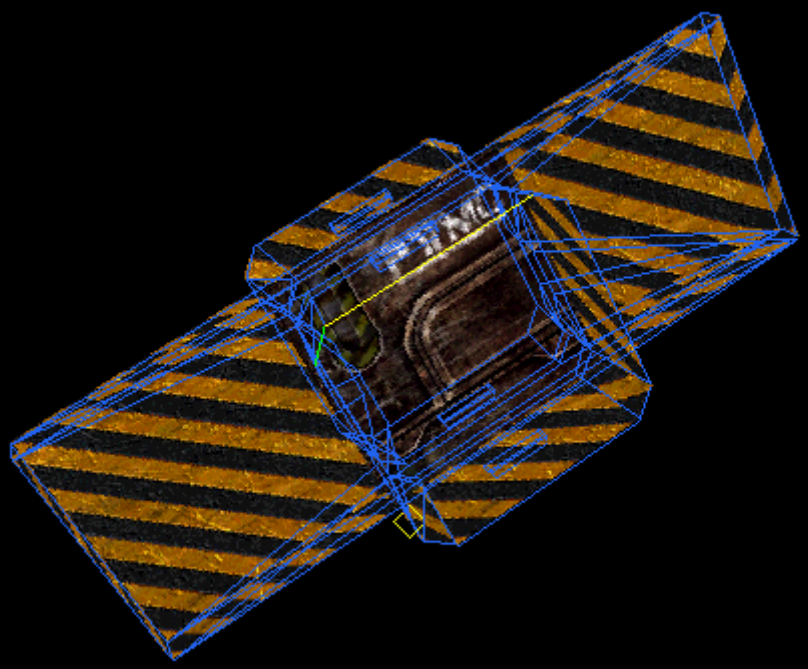
Türname:
PTMC Industrial 5
Größe ca:
38x38 Units, FF 20x20
Verts des Frontfaces: 8
Hat Region-Einbauten:
Nein
Bugs:
8 Unused Verts
Sonstiges:
-

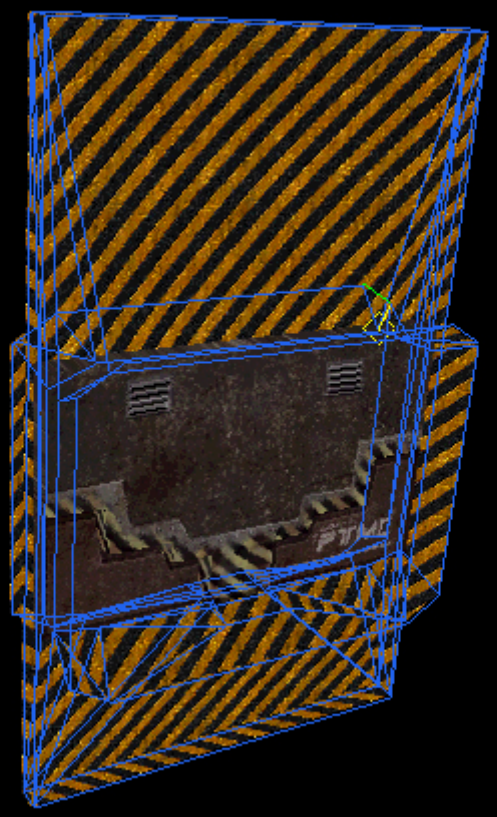




Türname:
PTMC Industrial 6
Größe ca:
55x22 Units, FF 20x20
Verts des Frontfaces: 8
Hat Region-Einbauten: Nein
Bugs: 8 Duplicate / UnusedVerts
Sonstiges:
Ein paar Faces wären zu optimieren.

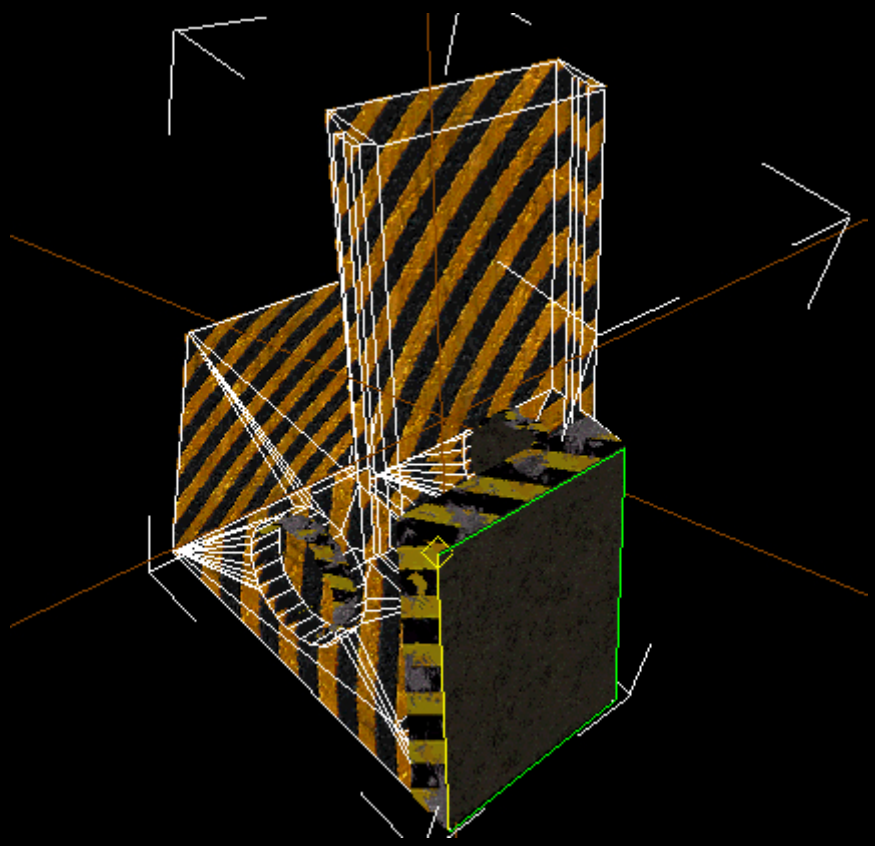
Türname:
PTMC Industrial 7
Größe ca:
21x63 Units, FF 20x20
Verts des Frontfaces: 10
Hat Region-Einbauten:
Ja
Bugs:
10 duplicate / Unused Verts, 1 T-Joint
Sonstiges:
Die Faces welche zu Portalen werden haben Extra-Verts an den Edges -> Optimierung empfohlen.

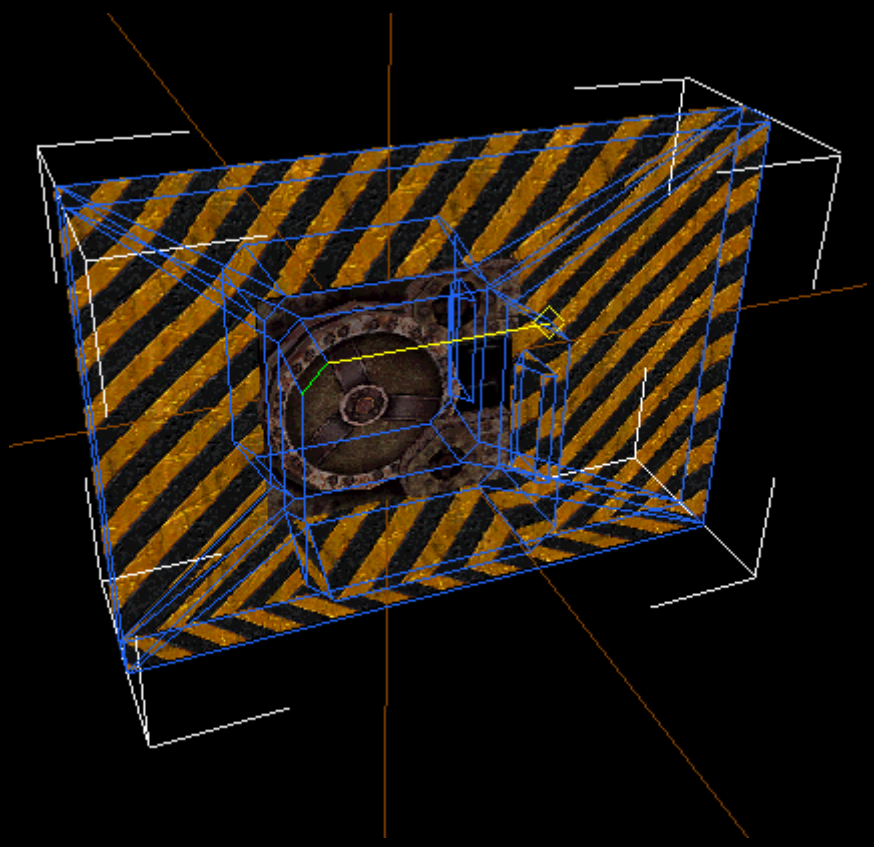




Türname:
PTMC Industrial 3
Größe ca:
62x105 Units, FF 60x46
Verts des Frontfaces: 8
Hat Region-Einbauten: Nein
Bugs: 8 Duplicate / Unused Verts
Sonstiges:
Die Tür wäre zwar aufwendig, aber doch zu optimieren.

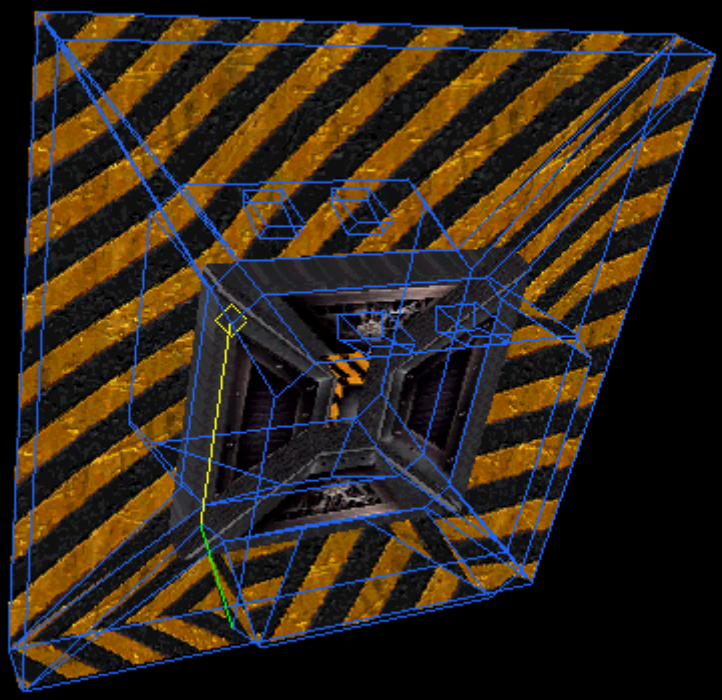
Türname:
SEAN'S ENERGY SECRET
Größe ca:
26x53 Units, FF 20x25
Verts des Frontfaces: 4
Hat Region-Einbauten:
Nein
Bugs:
4 duplicate / Unused Verts
Sonstiges:
Portalfaces nicht planparallel.

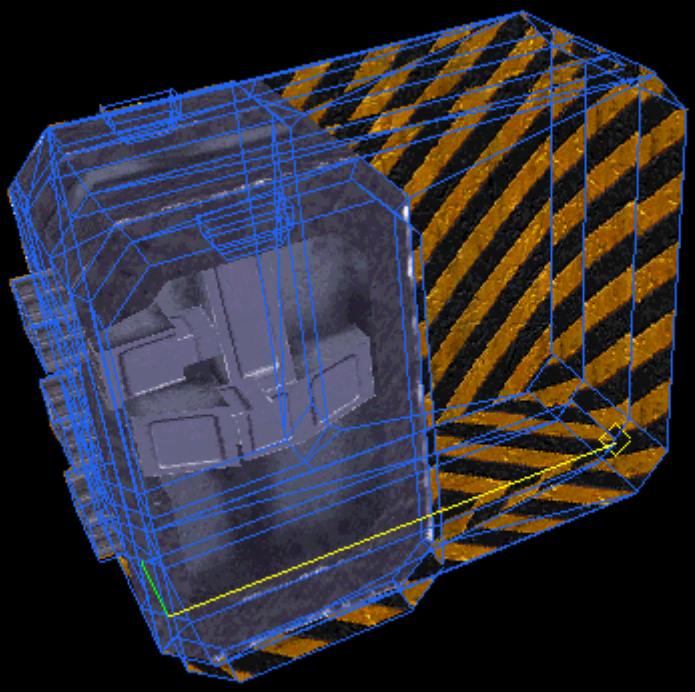




Türname:
PTMC Industrial 1
Größe ca:
52x39 Units, FF 20x20
Verts des Frontfaces: 8
Hat Region-Einbauten: Ja
Bugs: 8 Duplicate / UnusedVerts
Sonstiges:
-

Türname:
PTMC Covert 1
Größe ca:
39x41 Units, FF 20x20
Verts des Frontfaces: 8
Hat Region-Einbauten:
Ja
Bugs:
8 duplicate / Unused Verts
Sonstiges:
-

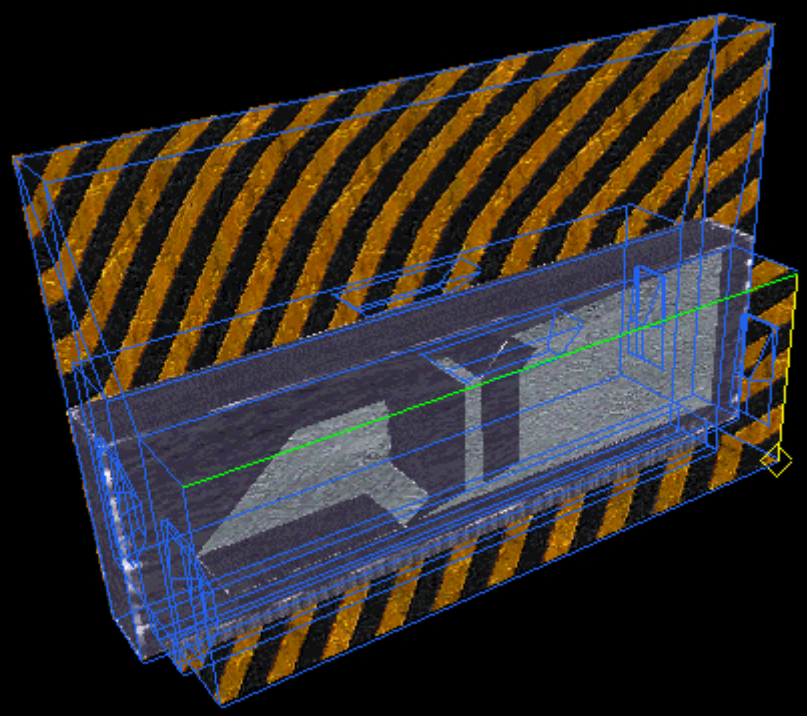


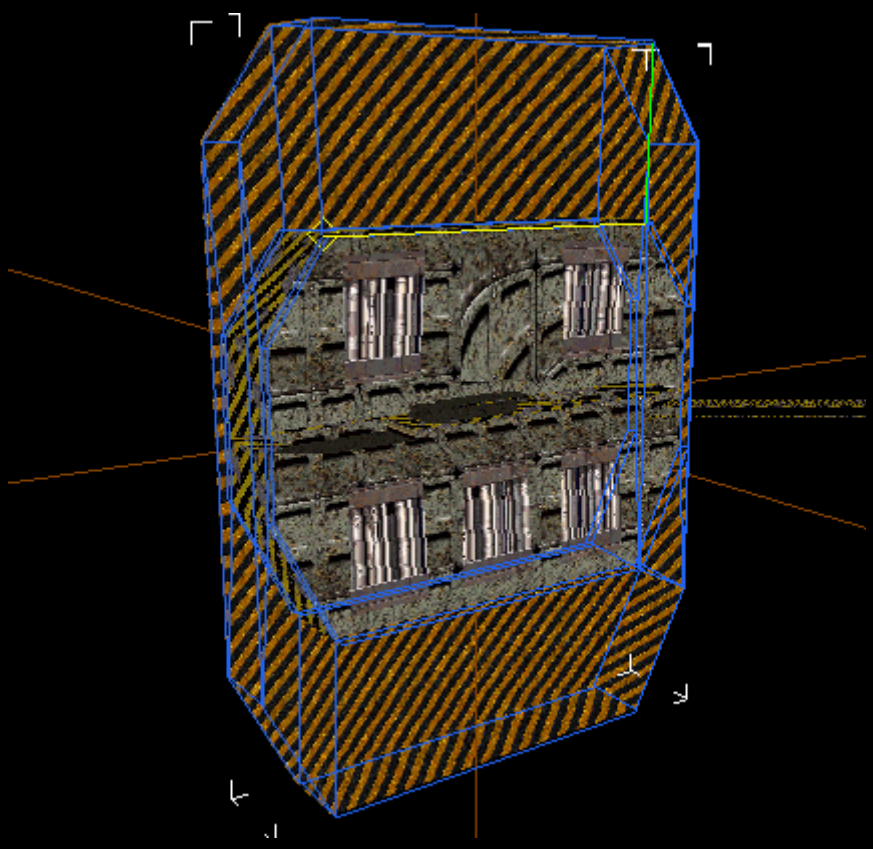


Türname:
ced_acc_door
Größe ca:
38x31 Units, FF 17x28
Verts des Frontfaces: 8
Hat Region-Einbauten: Ja
Bugs: 8 Duplicate / UnusedVerts, 8 T-Joints
Sonstiges:
Die Seite, wo man weiterbauen würde ist gesplittet -> Faces zusammenfassen. Beim Fehlerausbügeln öffnen sich Gaps.
Die Tür sollte optimiert werden.

Türname:
ced_wide_door
Größe ca:
56x36 Units, FF 50x17
Verts des Frontfaces: 4
Hat Region-Einbauten:
Ja
Bugs:
4 duplicate / Unused Verts, 16 Gaps
Sonstiges:
Der Unterbau muß optimiert werden.

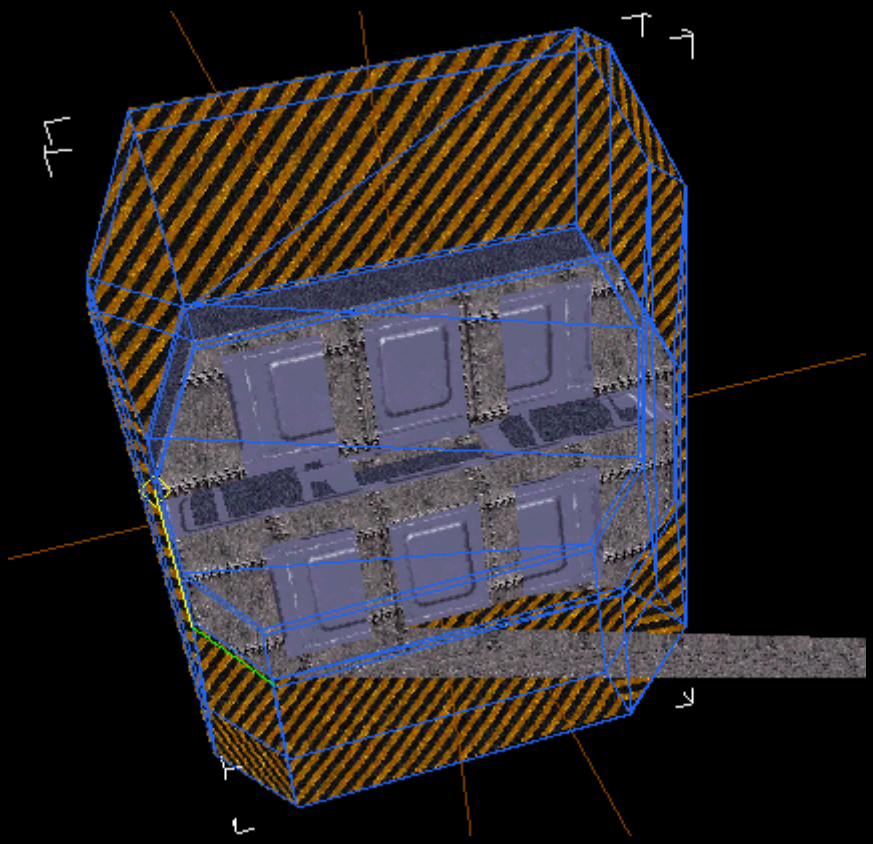
Die Tür ist nicht ganz unproblematisch; es ist mir zweimal passiert daß sie 'nen Level korrumpiert hat.

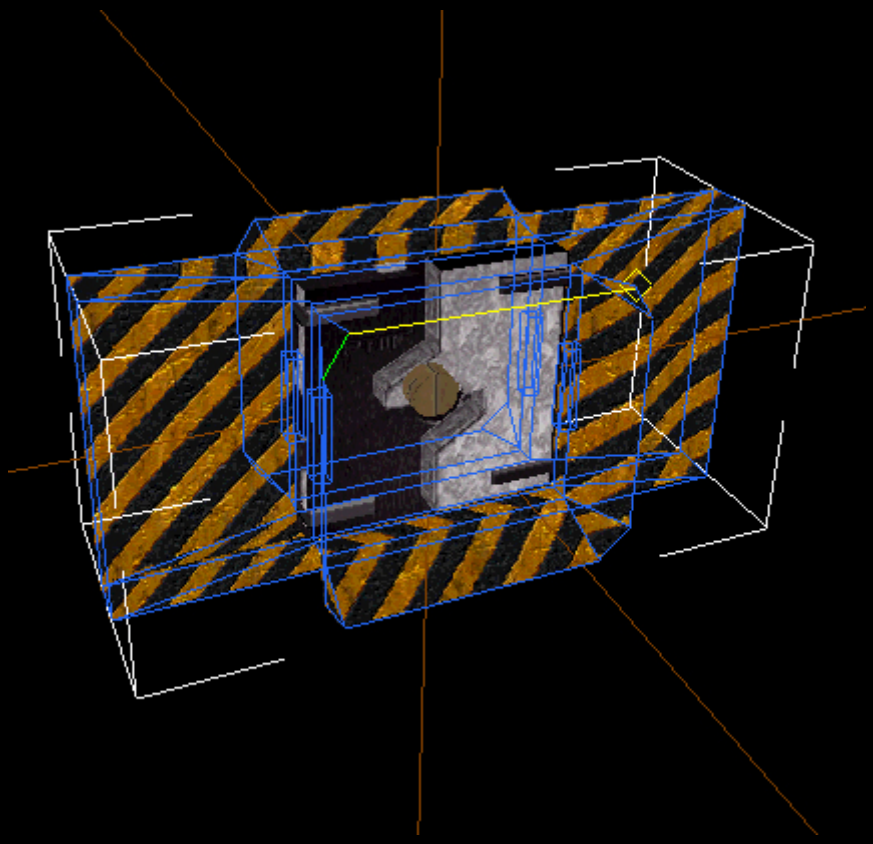




Türname:
PTMC Massive Door
Größe ca:
106x150 Units, FF 102x76
Verts des Frontfaces: 8
Hat Region-Einbauten: Nein
Bugs: 8 Duplicate / UnusedVerts, 8 Gaps
Sonstiges:
Das Türobjekt hat Concave Faces, die Shell ist nicht geschlossen.
Reparatur=Muß!
Optimierung empfohlen.

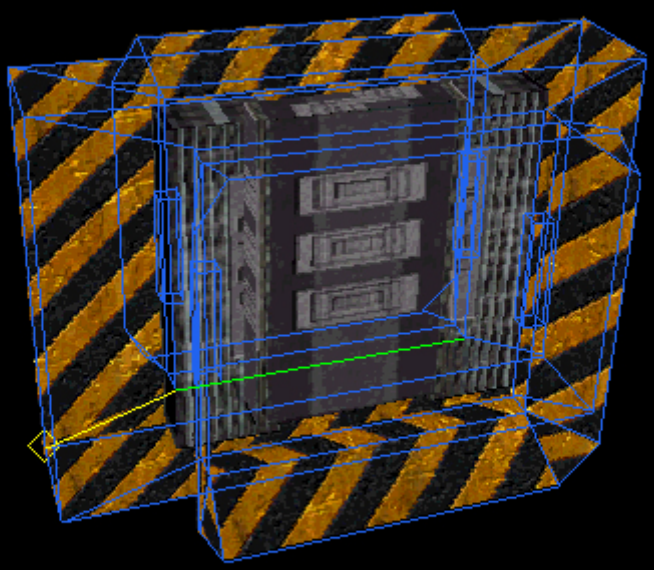
Türname:
exp_cargo_door
Größe ca:
106x150 Units, FF 102x76
Verts des Frontfaces: 8
Hat Region-Einbauten:
Nein
Bugs:
8 Duplicate / UnusedVerts, 8 Gaps
Sonstiges:
Das Türobjekt hat Concave Faces, die Shell ist nicht geschlossen. Reparatur=Muß!
Die Seite wo man weiterbauen würde, ist gesplittet - eine Faceunion mit **Strg-Shift-LKlick** ist durchzuführen.
Optimierung empfohlen.

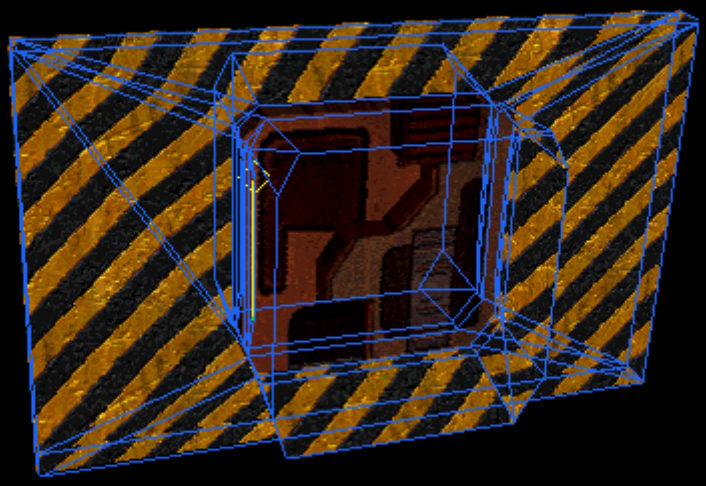




Türname:
PTMC Corporate A
Größe ca:
49x24 Units, FF 23x22
Verts des Frontfaces: 8
Hat Region-Einbauten: Nein
Bugs: 8 Duplicate / UnusedVerts
Sonstiges:
-

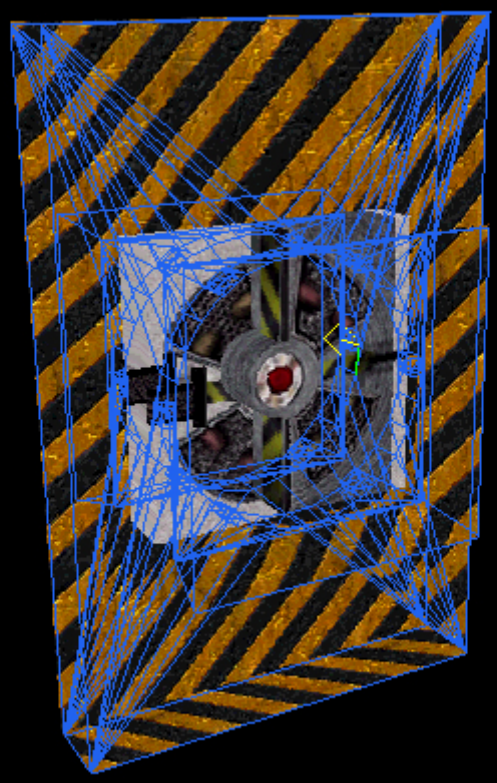
Türname:
PTMC Corporate B
Größe ca:
34x25 Units, FF 23x22
Verts des Frontfaces: 8
Hat Region-Einbauten:
Ja
Bugs:
8 duplicate / Unused Verts
Sonstiges:
Sehr kompakte Tür.

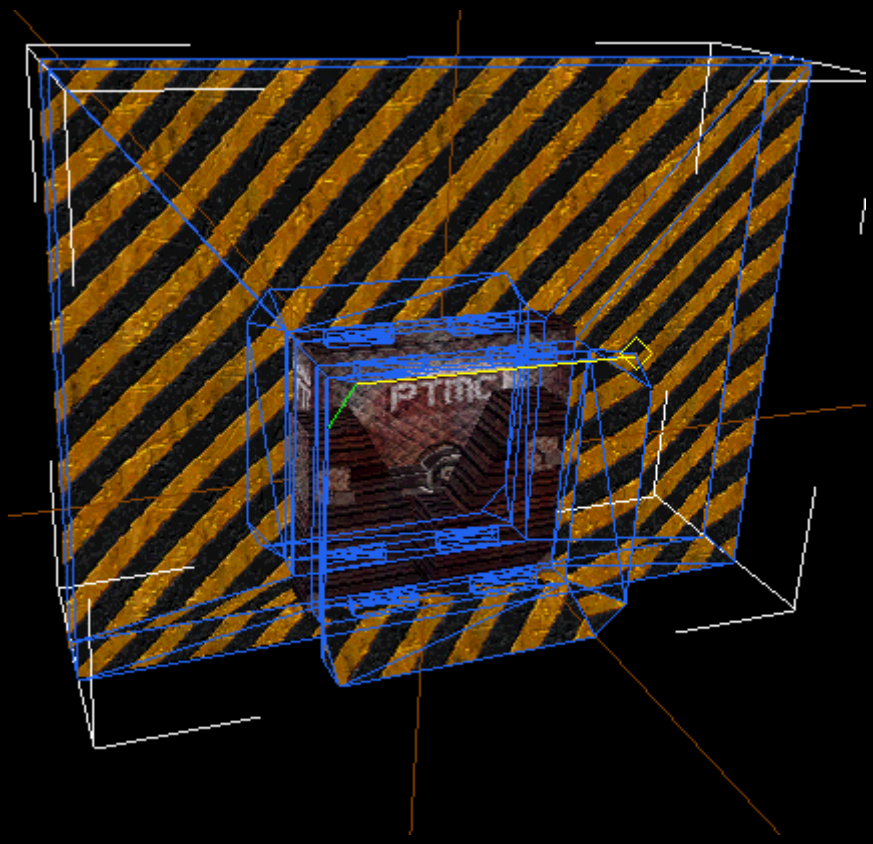




Türname:
PTMC Corporate E
Größe ca:
49x31 Units, FF 19x22
Verts des Frontfaces: 8
Hat Region-Einbauten: Nein
Bugs: 8 Duplicate /Unused Verts
Sonstiges:
Tür kann optimiert werden.

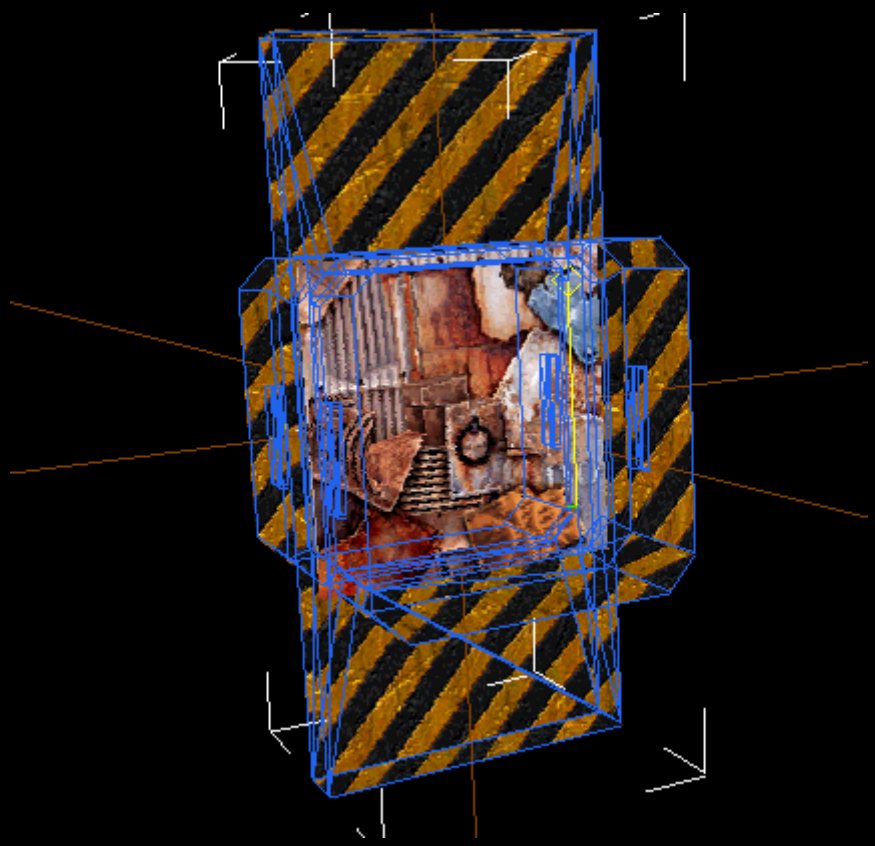
Türname:
PTMC Covert A
Größe ca:
31x48 Units, FF 21x21
Verts des Frontfaces: 4
Hat Region-Einbauten:
Ja
Bugs:
4 duplicate / Unused Verts
Sonstiges:
-

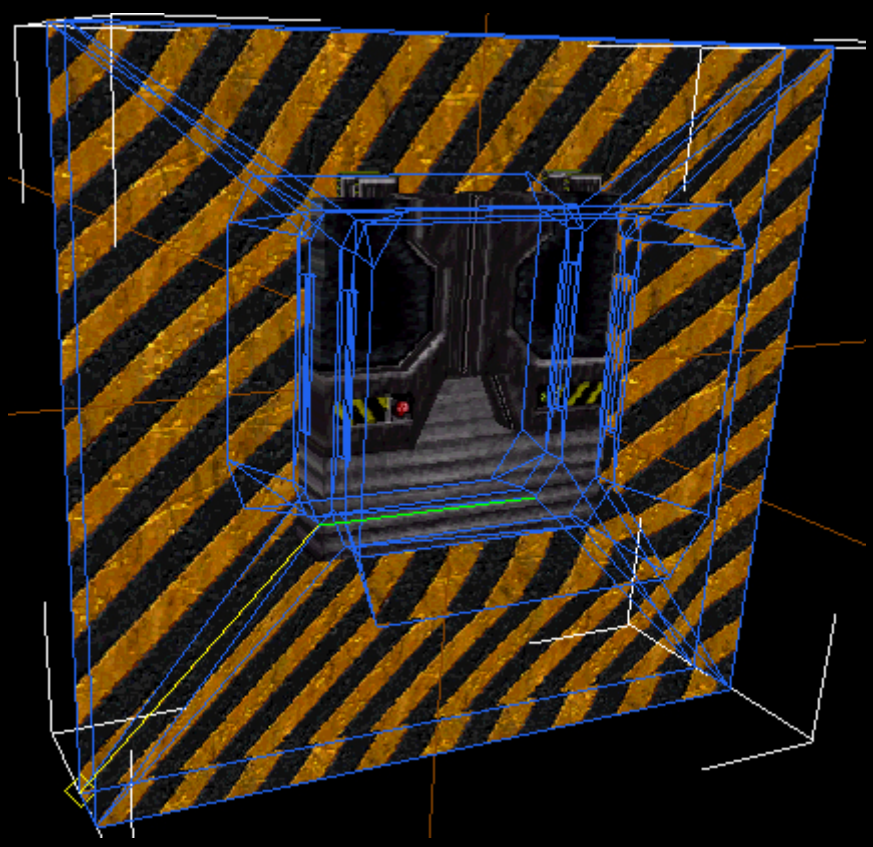




Türname:
PTMC Industrial A
Größe ca:
54x42 Units, FF 19x22
Verts des Frontfaces: 8
Hat Region-Einbauten: Ja
Bugs: 8 Duplicate / UnusedVerts
Sonstiges:
Optimierung der Shell empfohlen.

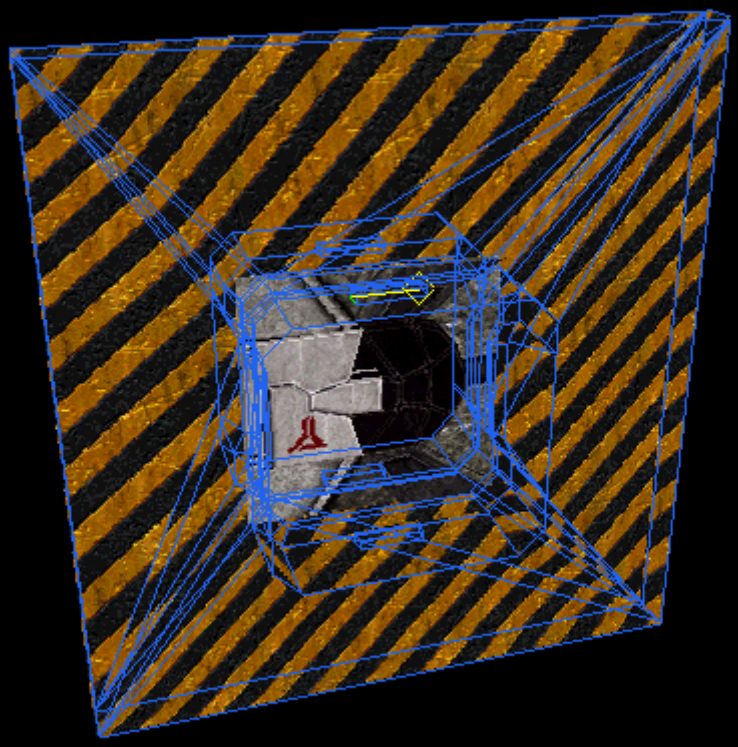
Türname:
Nomad 1
Größe ca:
21x46 Units, FF 21x21
Verts des Frontfaces: 8
Hat Region-Einbauten:
Ja
Bugs:
8 duplicate / Unused Verts
Sonstiges:
-

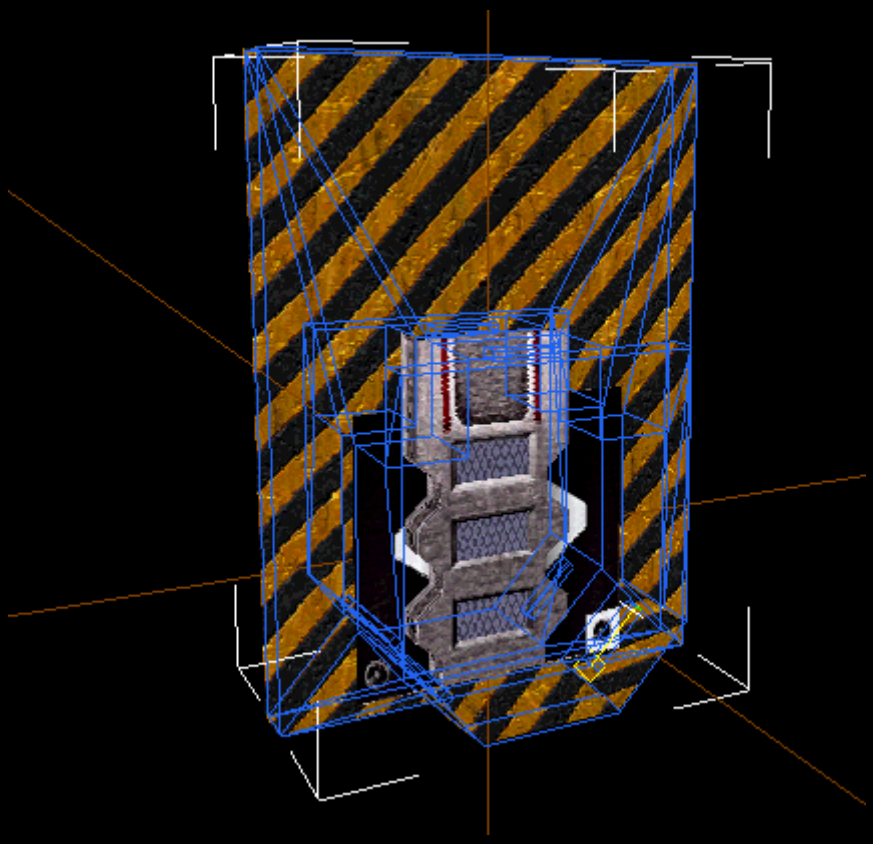




Türname:
PTMC Covert B
Größe ca:
45x45 Units, FF 23x22
Verts des Frontfaces: 8
Hat Region-Einbauten: Nein
Bugs: 8 Duplicate /Unused Verts
Sonstiges:
Shell könnte eine Optimierung vertragen.

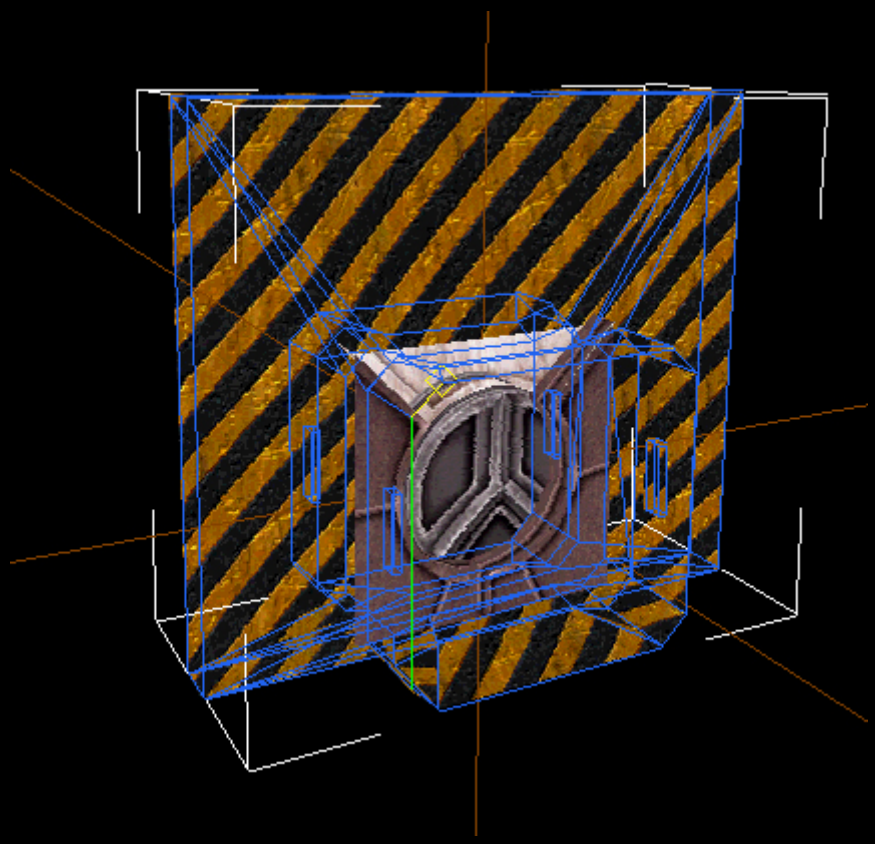
Türname:
Red Acropolis A
Größe ca:
51x53 Units, FF 21x21
Verts des Frontfaces: 12
Hat Region-Einbauten:
Ja
Bugs:
12 duplicate / Unused Verts
Sonstiges:
-

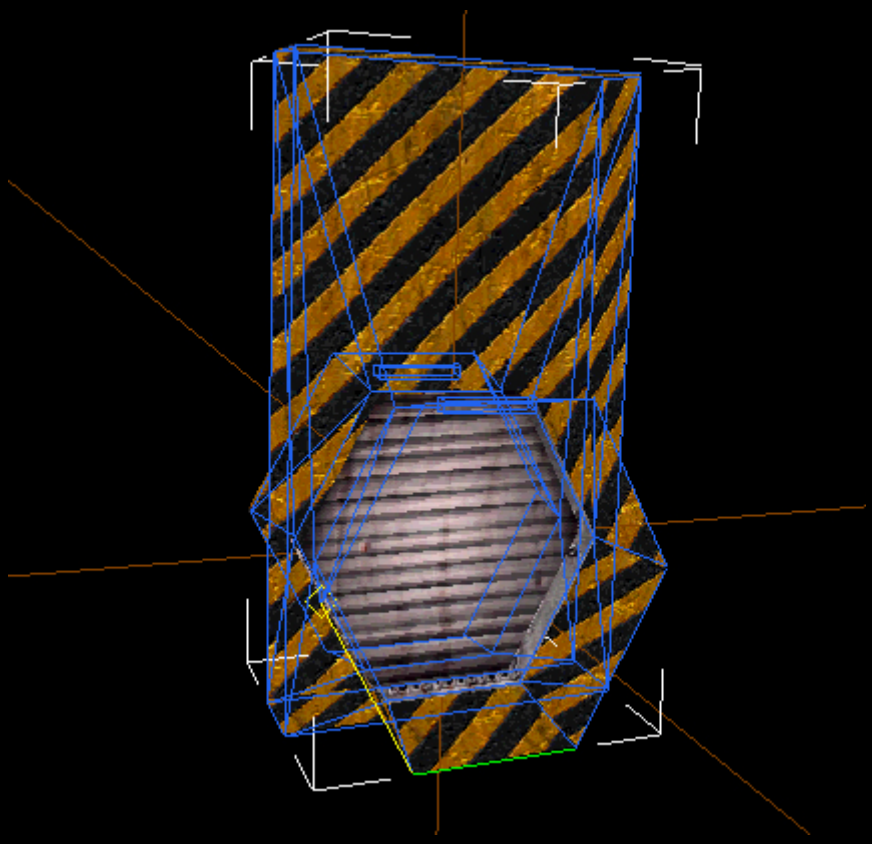




Türname:
CED Station
Größe ca:
31x43 Units, FF 20x25
Verts des Frontfaces: 6
Hat Region-Einbauten: Ja
Bugs: 6 Duplicate / unusedVerts
Sonstiges:
Hat das See-Through-Flag

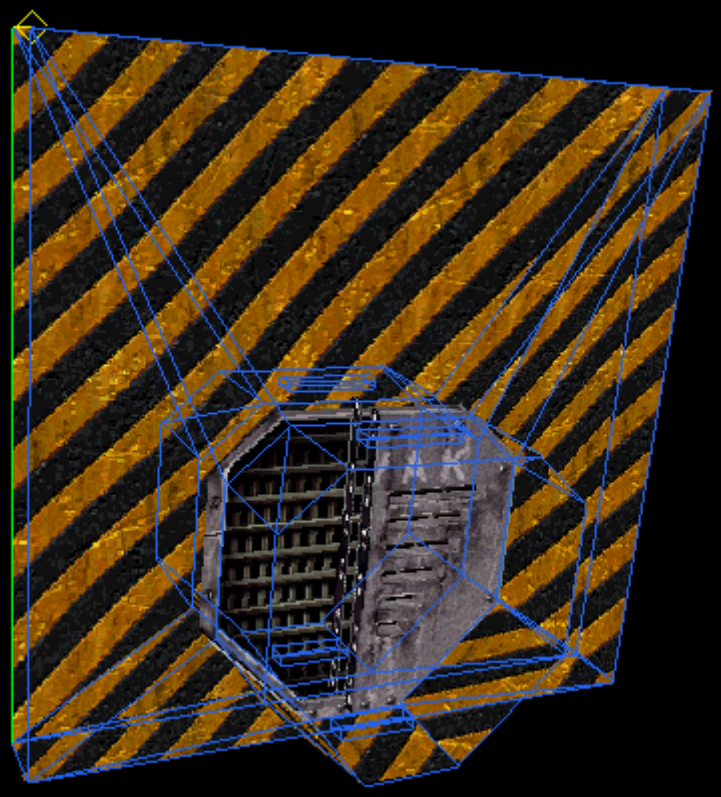
Türname:
Nomad 2
Größe ca:
41x37 Units, FF 21x21
Verts des Frontfaces: 8
Hat Region-Einbauten:
Ja
Bugs:
8 duplicate / Unused Verts
Sonstiges:
-

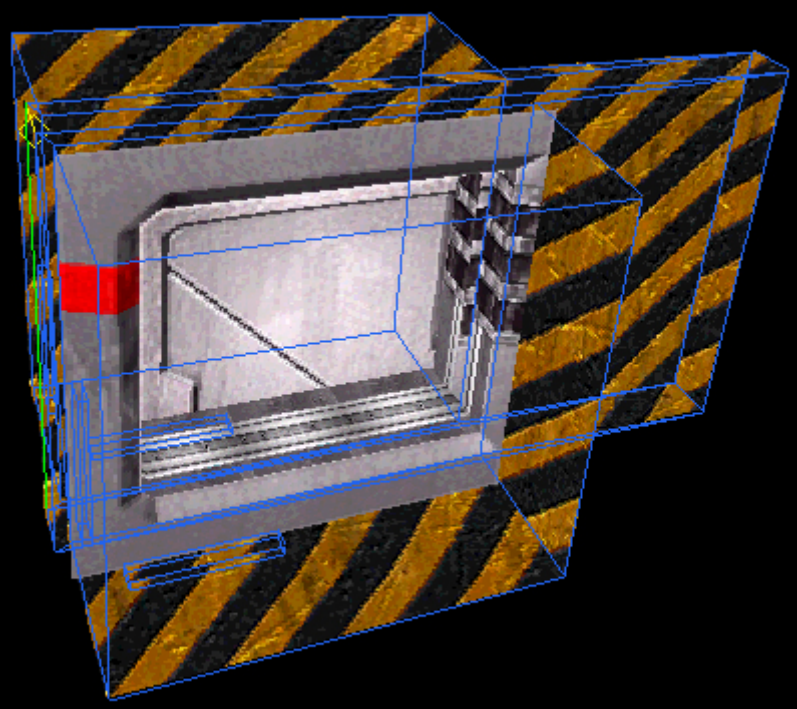




Türname:
Novak A
Größe ca:
23x42 Units, FF etwa 22 Durchmesser
Verts des Frontfaces: 6
Hat Region-Einbauten: Nein
Bugs: 6 Duplicate / unused Verts
Sonstiges:
Mit der Magnum kann man in der Tür steckenbleiben

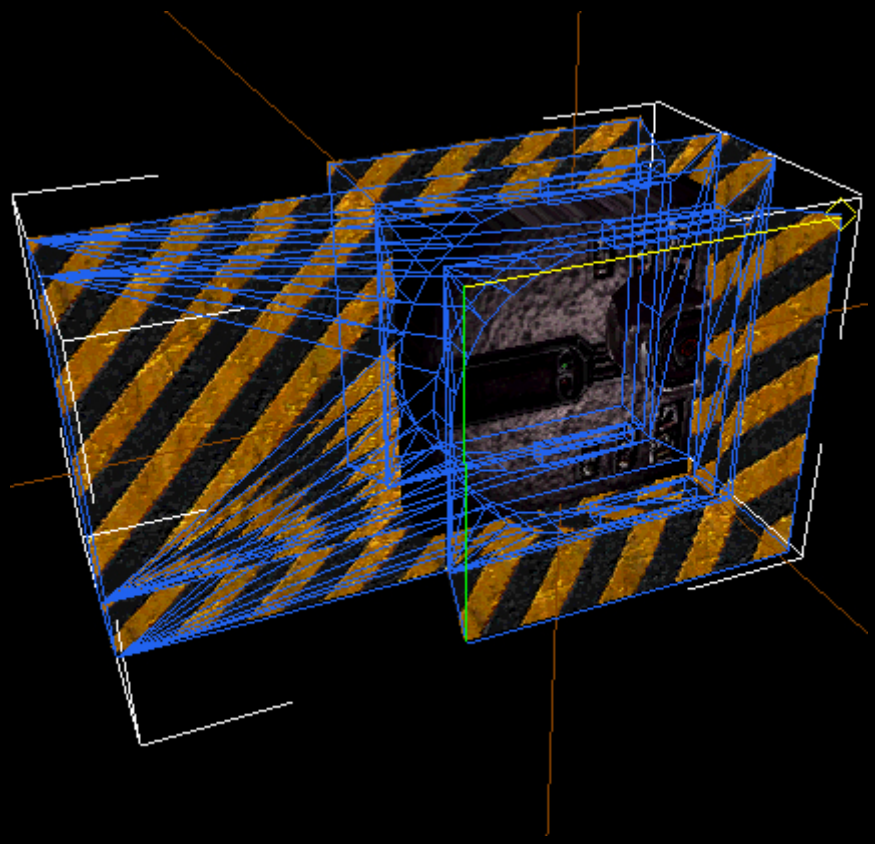
Türname:
Novak B
Größe ca:
41x40 Units, FF 19x19
Verts des Frontfaces: 8
Hat Region-Einbauten:
Ja
Bugs:
8 duplicate / Unused Verts
Sonstiges:
Das Door-Objekt hat Concave Faces.

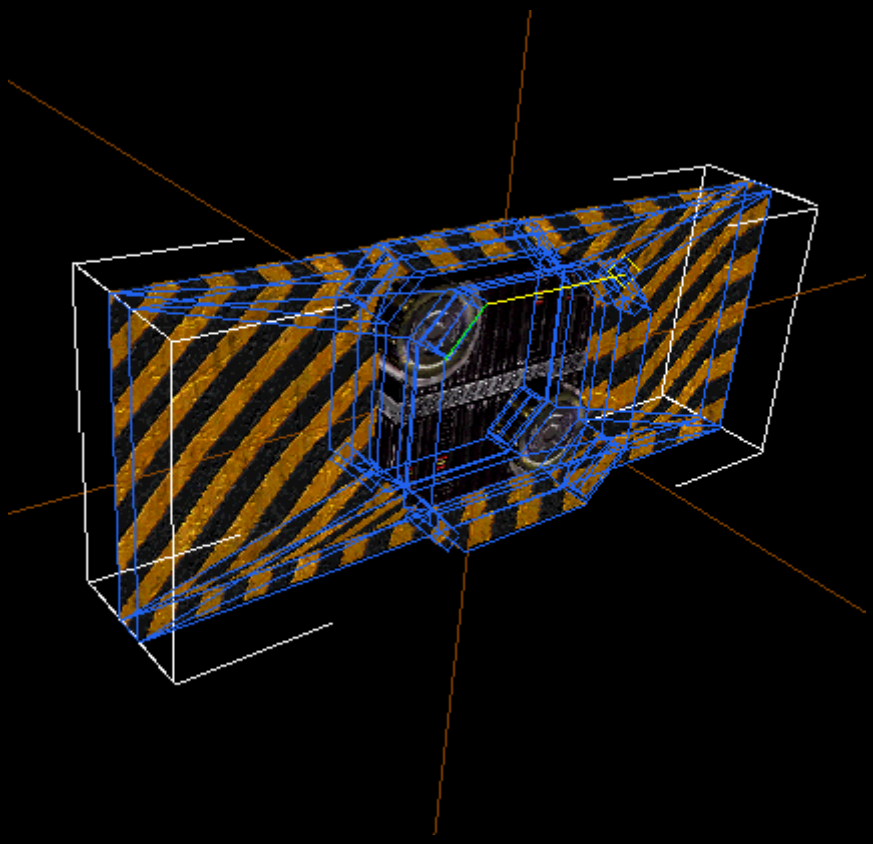




Türname:
PTMC Corporate C
Größe ca:
34x20 Units, FF 20x19
Verts des Frontfaces: 4
Hat Region-Einbauten: Ja
Bugs: 4 Duplicate / UnusedVerts
Sonstiges:
Man neigt dazu, in dieser Tür stecken zu bleiben

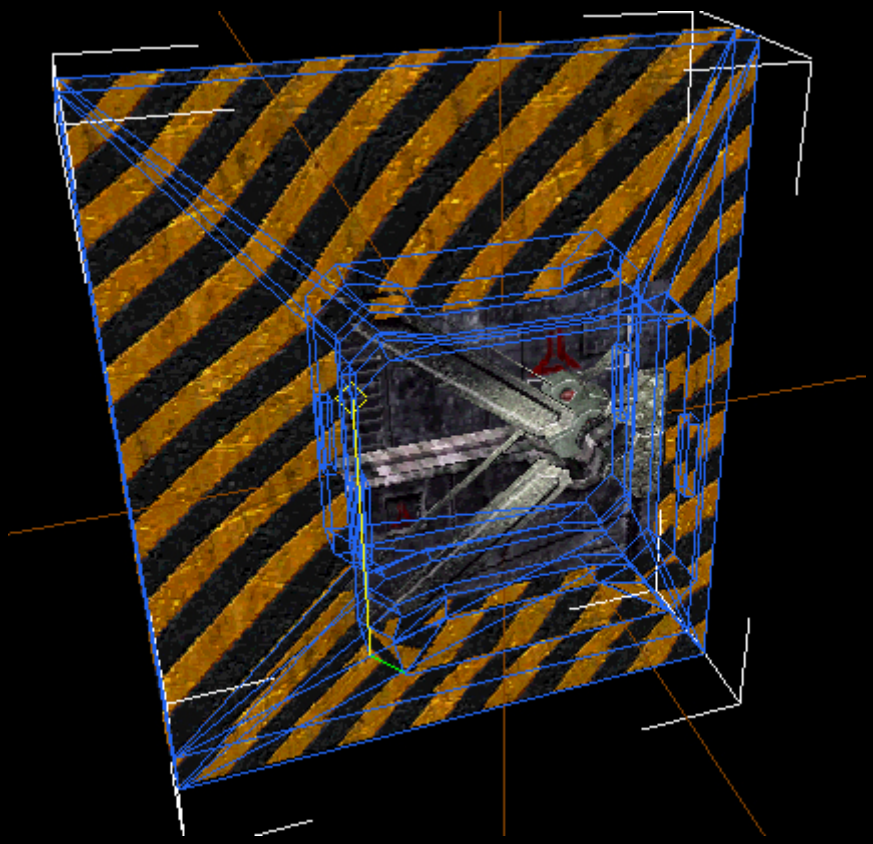
Türname:
PTMC Covert D
Größe ca:
40x23 Units, FF 21x21
Verts des Frontfaces: 4
Hat Region-Einbauten:
Ja
Bugs:
4 duplicate / Unused Verts
Sonstiges:
-

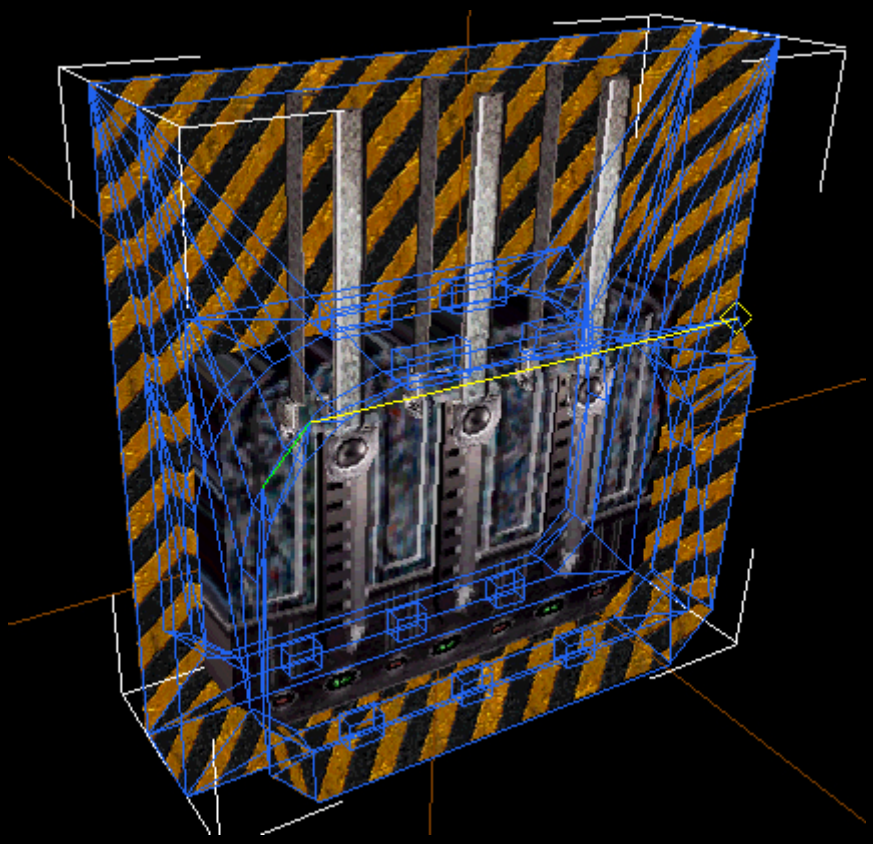




Türname:
PTMC Covert E
Größe ca:
60x24 Units, FF 21x21
Verts des Frontfaces: 8
Hat Region-Einbauten: Nein
Bugs: 8 Duplicate / unusedVerts
Sonstiges:
-

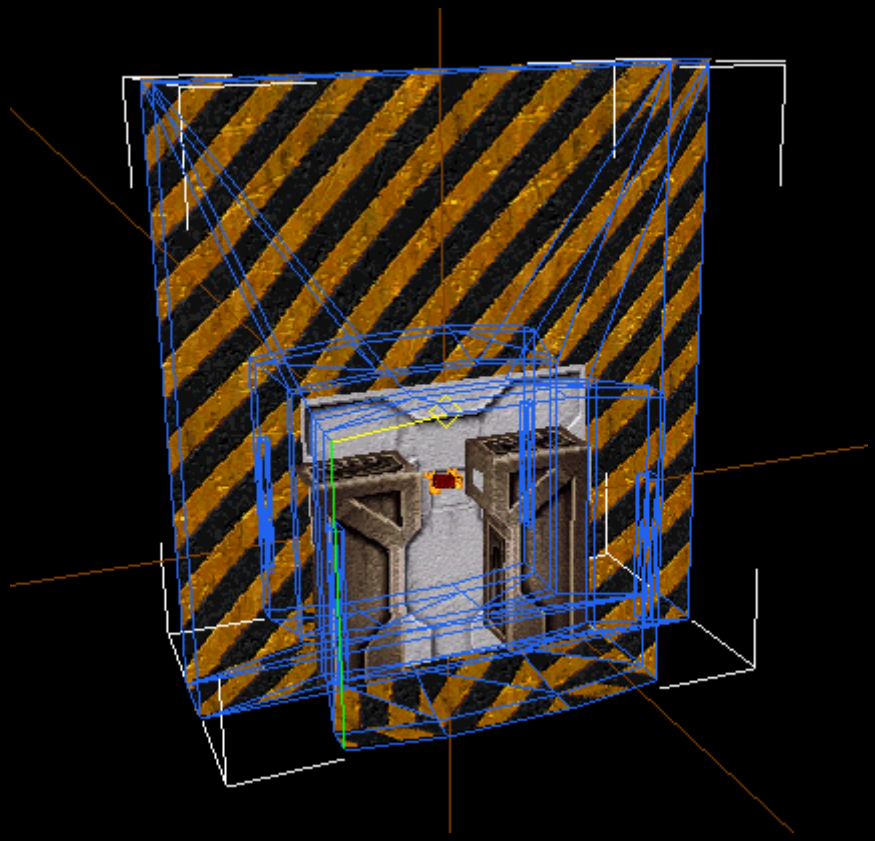
Türname:
Red Acropolis B
Größe ca:
36x43 Units, FF 21x21
Verts des Frontfaces: 8
Hat Region-Einbauten:
Ja
Bugs:
8 duplicate / Unused Verts
Sonstiges:
Teile des Türobjektes bilden kein geschlossenes Polygonnetz.

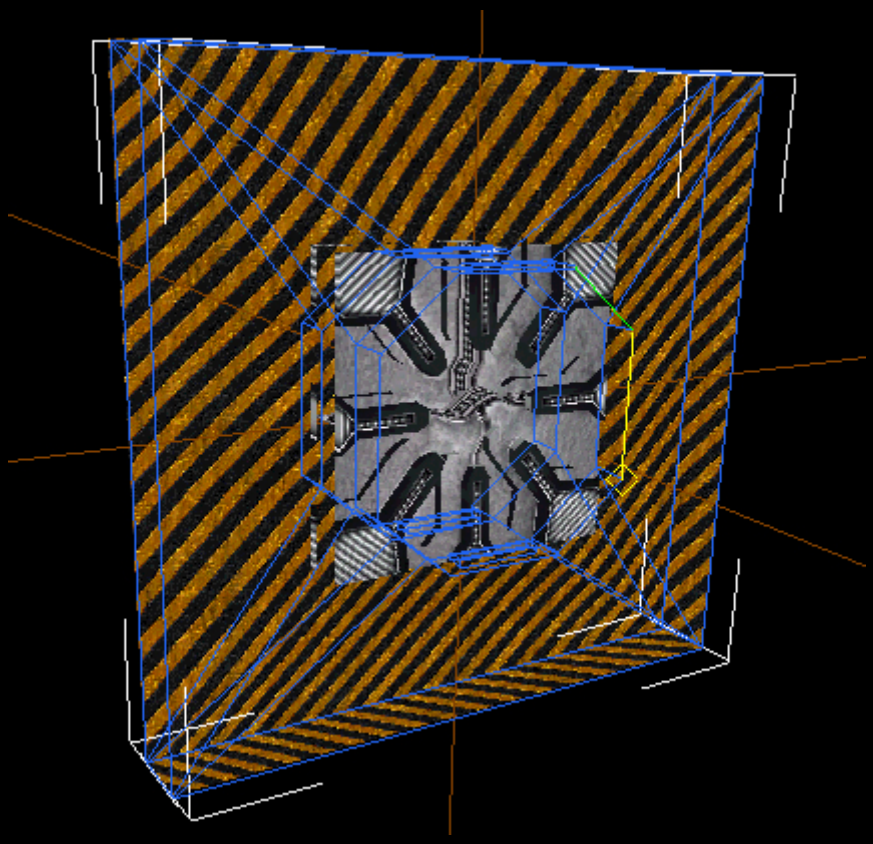




Türname:
PTMC Covert F
Größe ca:
50x51 Units, FF 42x31
Verts des Frontfaces: 8
Hat Region-Einbauten: Ja
Bugs: 8 Duplicate / unused Verts
Sonstiges:
-

Türname:
PTMC Covert C
Größe ca:
36x40 Units, FF 21x21
Verts des Frontfaces: 8
Hat Region-Einbauten:
Ja
Bugs:
8 duplicate / Unused Verts
Sonstiges:
Shelloptimierung empfohlen.
Hat die See through-Flag.

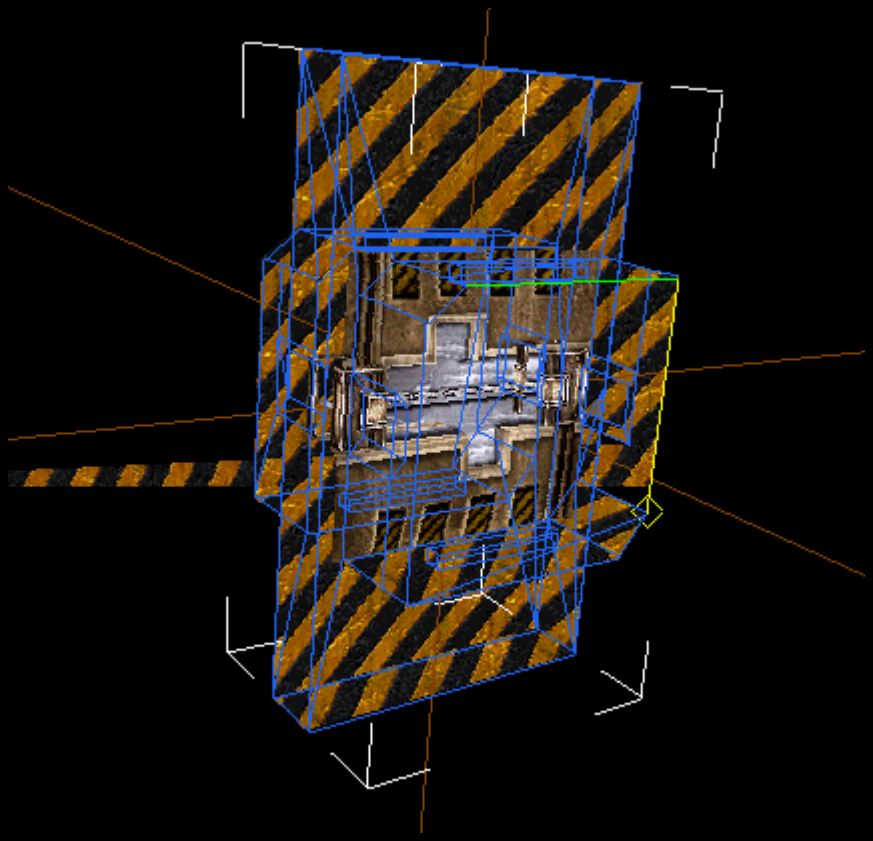


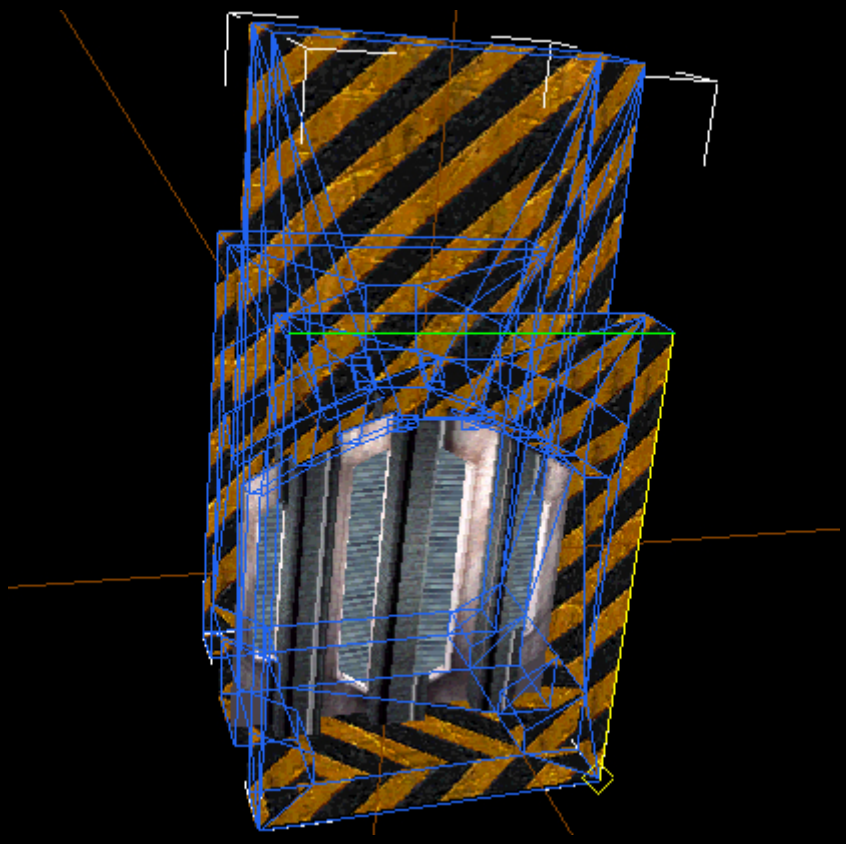


Türname:
PTMC Covert H
Größe ca:
84x83 Units, FF ist 37 im Durchmesser
Verts des Frontfaces: 8
Hat Region-Einbauten: Ja
Bugs: 8 Duplicate / unused Verts
Sonstiges:
Die Shell ist groß genug, daß kleinere Bots einfliegen können, das kann sehr lästig werden
Die Komponenten des Door-Objektes bilden kein geschlossenes Polygonnetz.

Türname:
PTMC Covert G
Größe ca:
23x43, FF 20x19 Units
Verts des Frontfaces: 6
Hat Region-Einbauten: Ja
Bugs:
2 Concave Faces, 1 Duplicate Face,
6 duplicate / Unused Verts, 2 T-Joints
Sonstiges:
Das Duplicate Face liegt über dem Anbau-Portal. Das FF ist gesplittet und erzeugt Gaps, wenn die Tür verbaut wird ->Portale auf beiden Seiten reparieren.
Die Seite wo man weiterbaut, ist ebenfalls gesplittet und muß zuvor vereinigt werden.

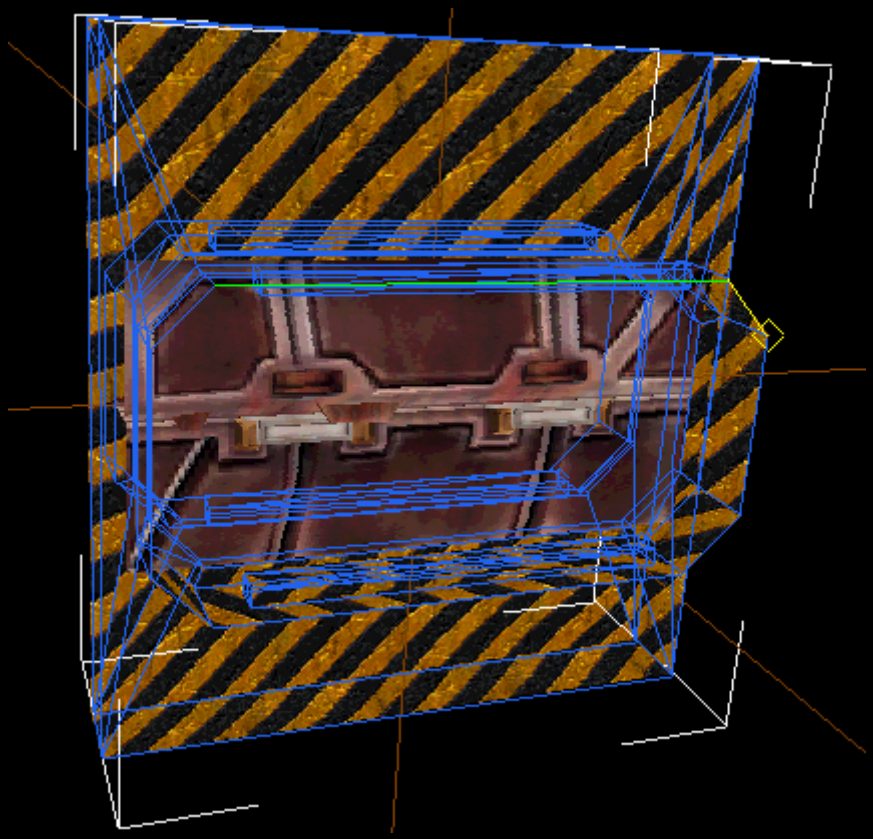
Reparatur=Muß!
Optimierung der Shell ist wärmstens empfohlen.

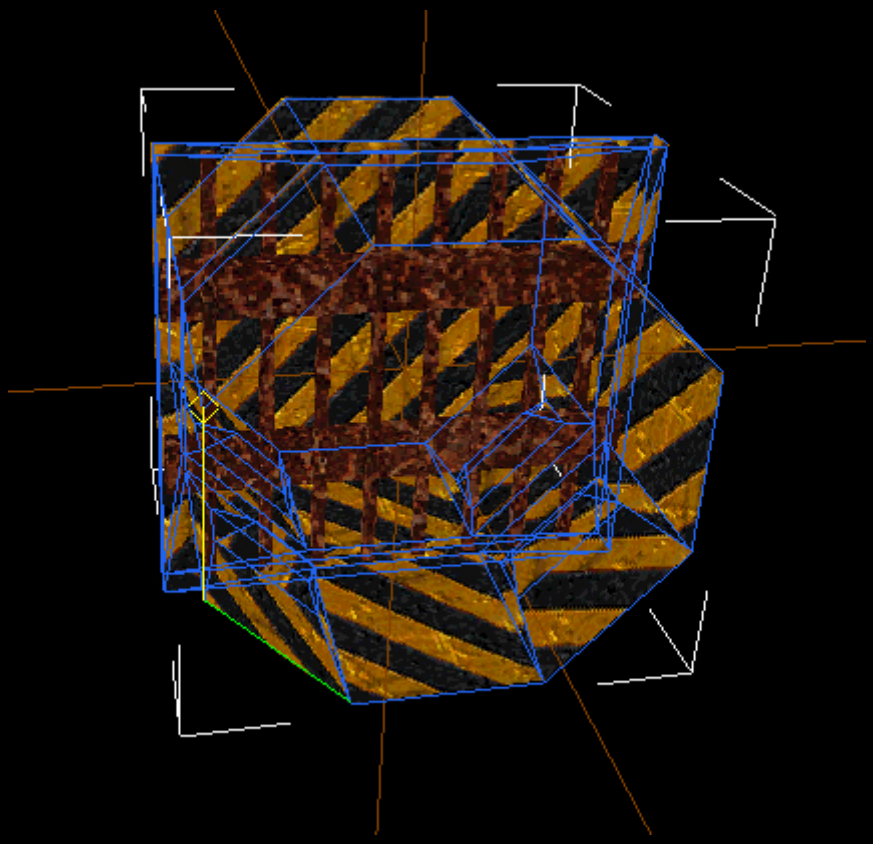




Türname:
PTMC Corporate D
Größe ca:
20x40 Units, FF 20x27
Verts des Frontfaces: 4
Hat Region-Einbauten: Ja
Bugs: 4 Duplicate / unused Verts, 2 Gaps
Sonstiges:
Es sind zwei Faces im seitlichen Bereich, welche sich überlagern, dort sind die Gaps. Das Door-Objekt bildet kein geschlossenes Polygonnetz.

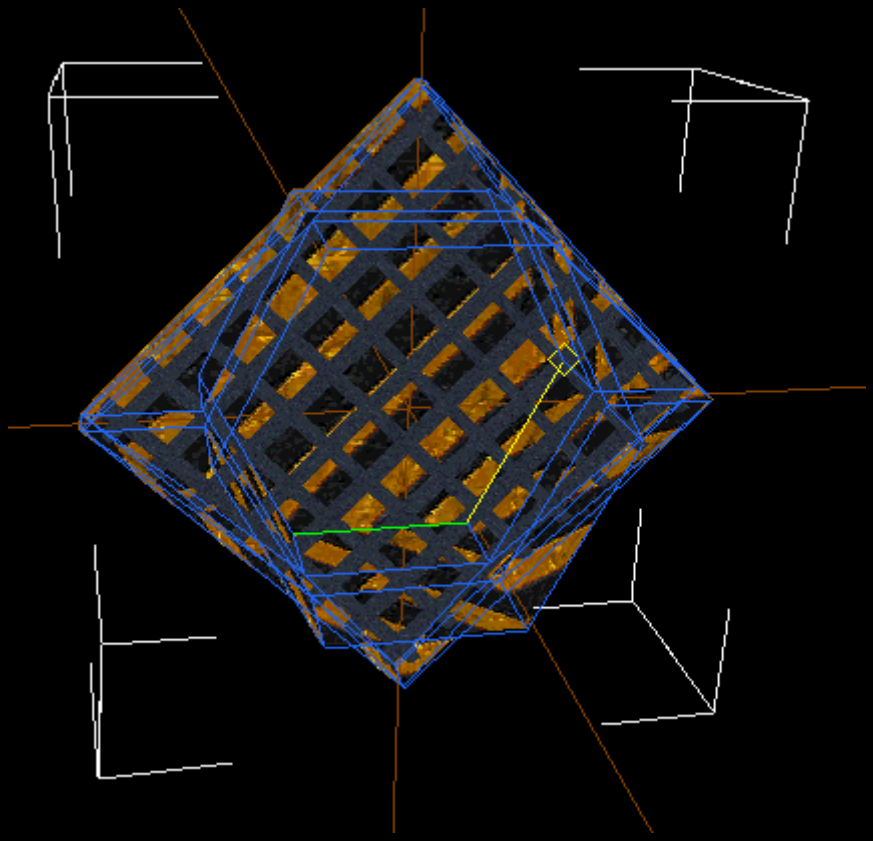
Türname:
Red Acropolis Big Door
Größe ca:
44x46 Units, FF 40x21
Verts des Frontfaces: 8
Hat Region-Einbauten:
Ja
Bugs:
8 duplicate / Unused Verts
Sonstiges:
-

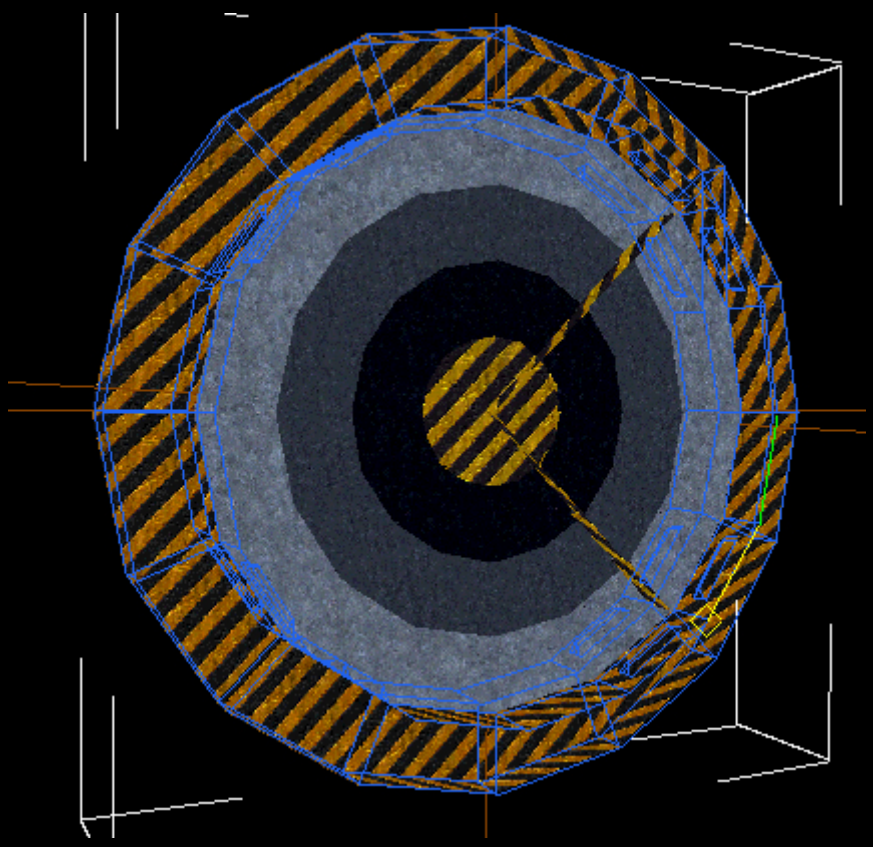




Türname:
PTMC Industrial 4
Größe ca:
22x21 Units, FF 20x20
Verts des Frontfaces: 8
Hat Region-Einbauten: Nein
Bugs: 8 Duplicate / unused Verts
Sonstiges:
Blastable Door

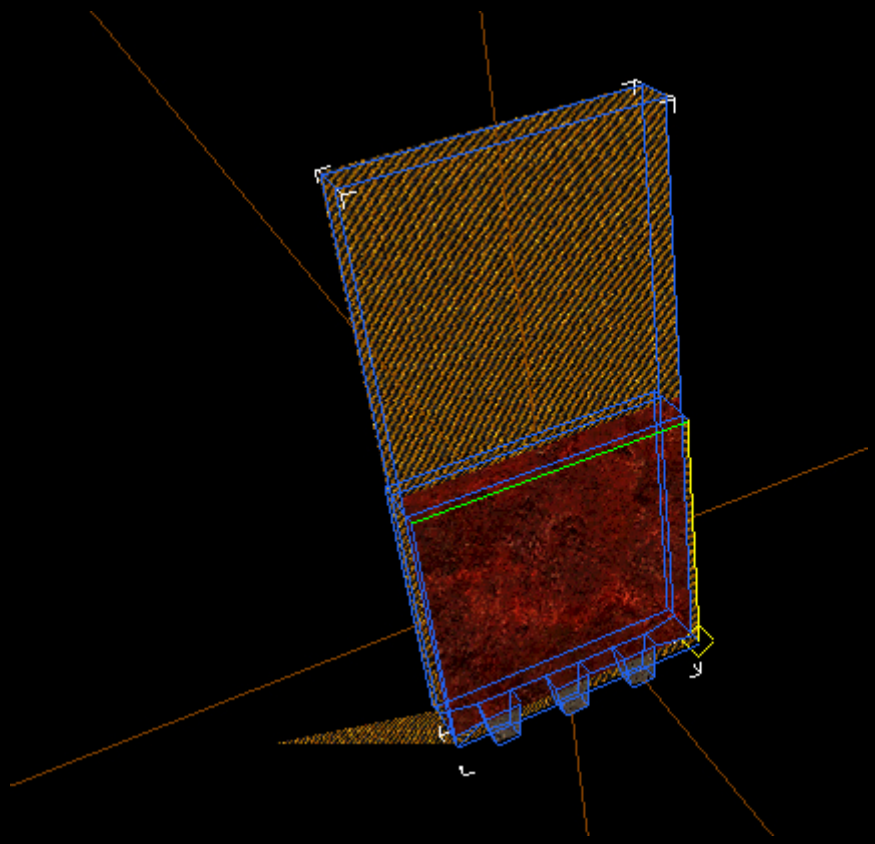
Türname:
NovakBlastServiceGrate
Größe ca:
28 Units in der Diagonalen, FF 18x17.
Verts des Frontfaces: 6
Hat Region-Einbauten:
Nein
Bugs:
6 duplicate / Unused Verts
Sonstiges:
Blastable

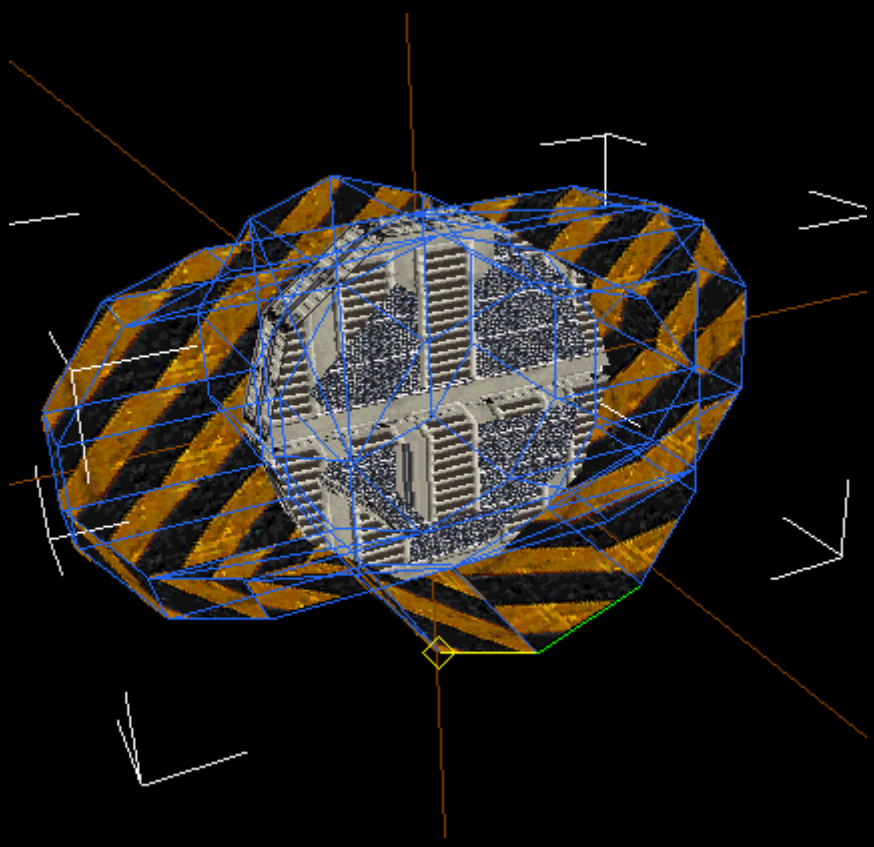




Türname:
Transfer Door
Größe ca:
96 Units Durchmesser, FF 86 im Durchmesser.
Verts des Frontfaces: 16
Hat Region-Einbauten: Ja
Bugs: 16 Duplicate / unused Verts
Sonstiges:
Tür-Objekt-Komponenten kein geschlossenes Polygonnetz.

Türname:
Lvl5BossDoor
Größe ca:
140x312 Units, FF 140x140
Verts des Frontfaces: 4
Hat Region-Einbauten:
Nein
Bugs:
4 duplicate / Unused Verts, 1 Concave Face
Sonstiges:
rieeeesig!
Das Concave am Fuß der Tür braucht bloß gesplittet zu werden.
Optimierung möglich.



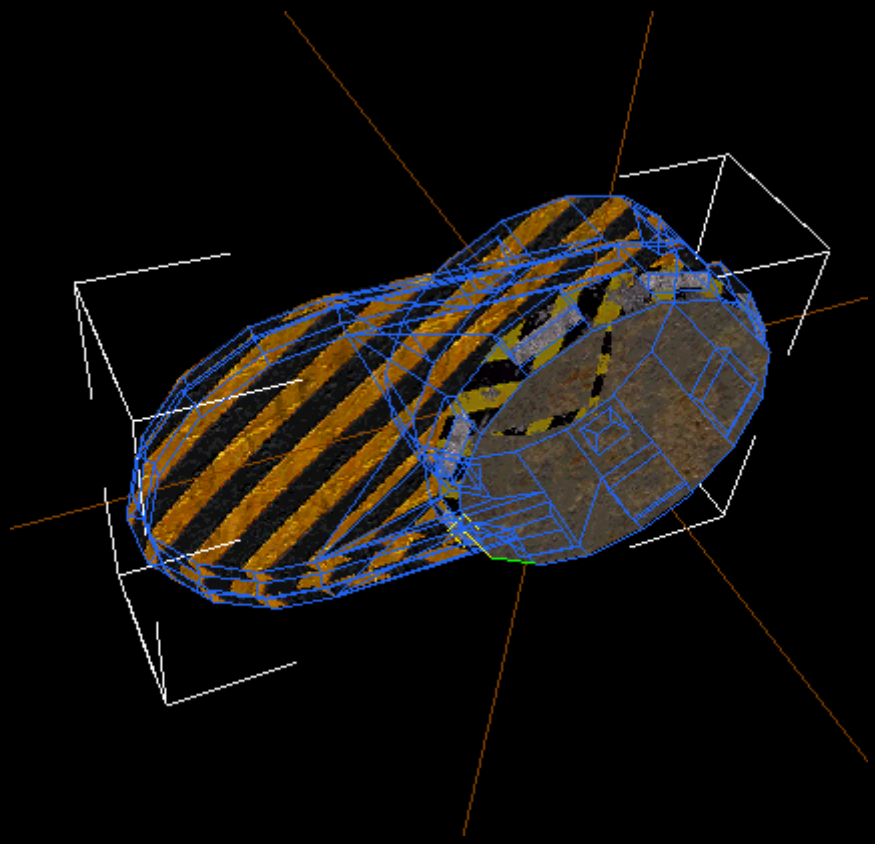


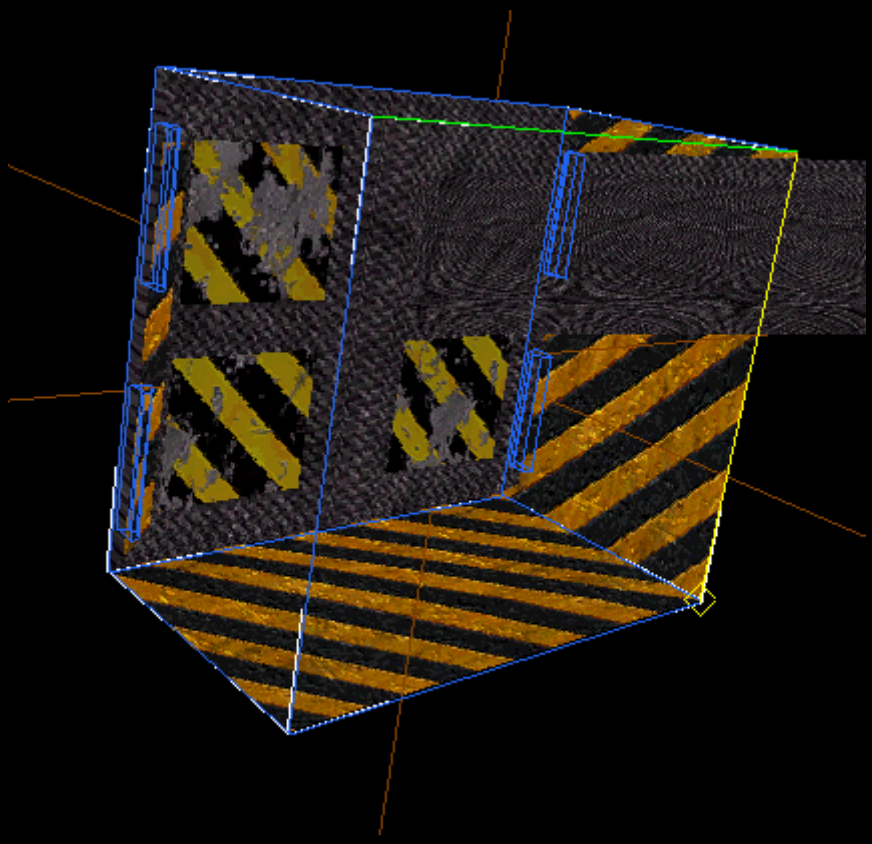
Türname:
SS_airlok_door
Größe ca:
31x15 Units, FF Durchmesser 15
Verts des Frontfaces: 10
Hat Region-Einbauten: Nein
Bugs: 10 Duplicate / unused Verts
Sonstiges:
Die Seite wo man weiterbaut muß zuvor vereinigt werden.
Optimierung möglich.

Türname:
piccusecretdoor
Größe ca:
35x20 Units, FF Durchmesser 19
Verts des Frontfaces: 16
Hat Region-Einbauten:
Ja
Bugs:
16 duplicate / Unused Verts, 3 T-Joints
Sonstiges:

Reparatur=Muß!
Optimierung empfohlen.

Ist prinzipiell die gleiche Tür wie die 'LUKE'S SECRET DOOR'.





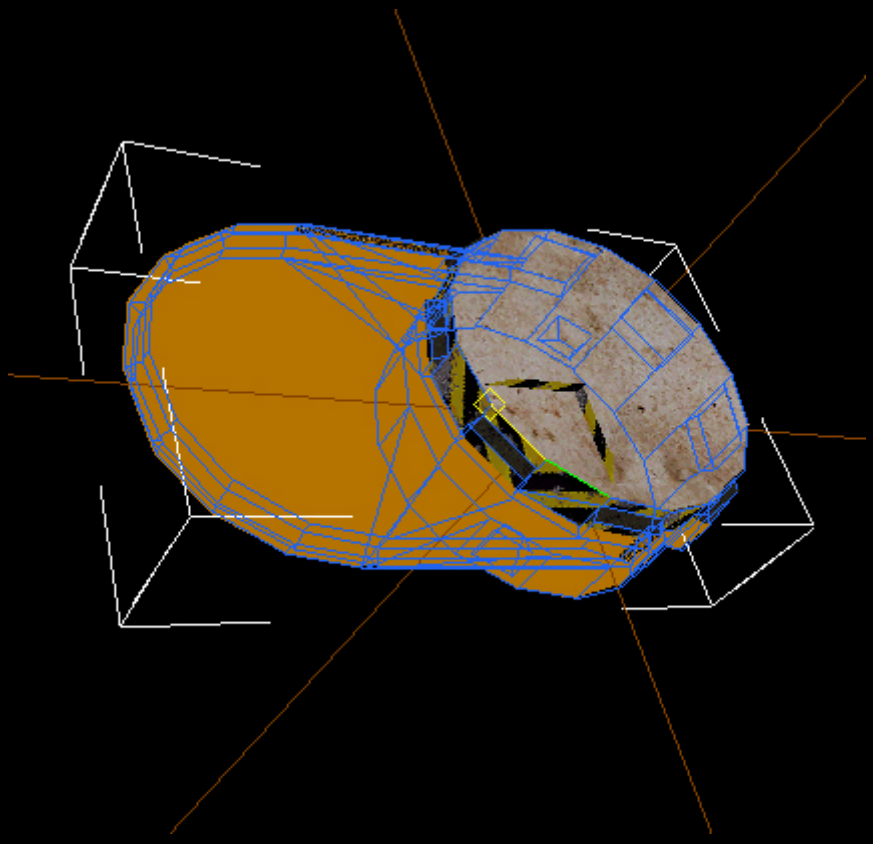
Türname:
deimostrapdoor
Größe ca:
28x26 Units
Verts des Frontfaces: 4
Hat Region-Einbauten: Ja
Bugs: 4 Duplicate / unusedVerts
Sonstiges:
das Türobjekt enthält 4 Concave Faces.

ist prinzipiell die gleiche Tür wie 'SEAN'S TRAPDOOR'.

Türname:
seoulsecretdoorround 2
Größe ca:
35x20 Units, FF Durchmesser 19
Verts des Frontfaces: 16
Hat Region-Einbauten:
Ja
Bugs:
16 duplicate / Unused Verts, 3 T-Joints
Sonstiges:

Reparatur=Muß!
Optimierung empfohlen.

Ist prinzipiell die gleiche Tür wie die 'LUKE'S SECRET DOOR'.



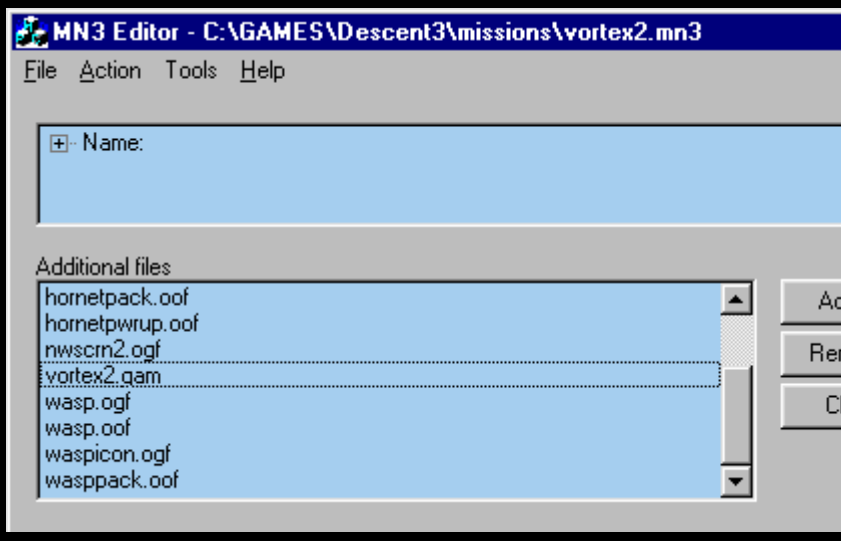
Custom-Objekte in D3Edit sichtbar haben

Papacat

Anm: dies hier zur Vollständigkeit, da man sich diese Arbeit seit Atans Entdeckung - siehe Customs, nochmal - eigentlich sparen kann.

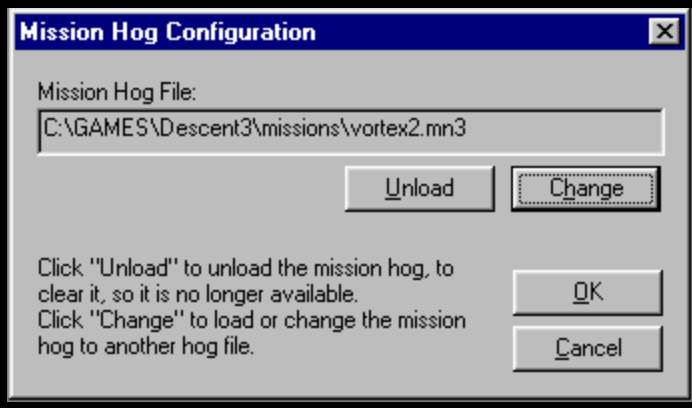
Benutzerdefinierte Texturen und Objekte können in D3Edit unkompliziert verwendet werden, doch sind sie während des Editierens nicht immer sichtbar; das macht Ausrichten und Positionieren naturgemäß etwas schwierig. Der hier beschriebene Vorgang wurde zuvor bereits mehrmals erwähnt, doch hier das Ganze nochmal in allen Einzelheiten.

Wenn Du mit dem Erzeugen der `.oof`'s die Du verwenden willst fertig bist und eine Custom-`.gam` erstellst, mußt Du eine `.hog`-Datei machen. Dies machst Du, indem Du eine `.mn3`-Datei erzeugst; denn diese IST eine HOG-Datei.
Starte einfach den MN3-Packager und füge alle Deine Custom `.oof`'s, `.ogf`'s, `.oaf`'s und die Custom-`.gam` über den Button **Add file(s)** hinzu. Du darfst KEIN Level hinzufügen oder irgendwelche Missionsinformationen. Speichere es unter einem beliebigen Namen. Ich verwende hier meine `.mn3` aus Vortex2 als Beispiel, aber Du kannst sie `Custom.mn3`, `Temp.mn3`... wie-auch-immer nennen. Denke nur daran, daß dies nicht Deine FINALE `.mn3` ist, sondern nur eine temporäre.



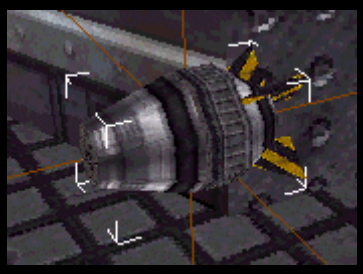
Starte D3Edit. Bevor Du anfängst oder irgendwas öffnest, mußt Du D3Edit erklären, daß es neben der `d3.hog` auch Deine `.mn3` (=`hog`) lädt, um Objekte und Texturen verfügbar zu machen.

Gehe nach Data->Configure A Mission Hog wie hier links. Dies öffnet den Mission Hog Configuration-Dialog. Die Textbox sollte anfangs leer sein. Klicke **Change** und navigiere zu Deiner `.mn3`-Datei. Wähle sie aus, klicke **Open** und ihr Pfad wird in die Textbox eingetragen. Dann klicke OK.

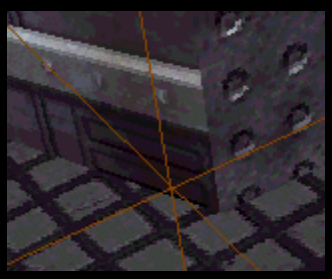


Deine hog-Datei (=die `.mn3`) ist nun geladen und für D3Edit verfügbar. Nun mußt Du die `.gam`-Datei von Deiner custom-`.mn3` auswählen.
Dau gehe nach Data -> Tablefile Manager. In der Mission Tablefile-Zeile klicke **HOG Browse**, nicht **File Browse**. Der Hog-File-Browser geht nun auf und listet die von geladenen `.hog`-Dateien verfügbaren `.gam`-Dateien. Wähle Deine aus der Liste und schließe den Tablefile Manager mit **OK**.
Das wars! Deine eigenen Objects und Texturen sind jetzt nicht nur verfügbar, sondern auch SICHTBAR in beiden, Den Auswahlpaletten als auch den texturierten Ansichten.

Hier siehst Du den Vorteil der Extra-5-Minuten um Deine `.mn3` zu machen. Das Objekt ist sichtbar, was beim Positionieren hilft.



Du kannst eine Custom-`.gam` auch ohne das Erstellen einer `.mn3` verwenden, aber dann siehst Du Deine Objekte nicht.



Das Gam-Format: The Specification File

Outrage

The Descent 3 Tablefile Format (.GAM) File Spec. (Text file version: 0.501)

(erweitert)

History:

Version	***Description***
0.503	Info for some Parameters added - Ragil Ral
0.502	Fixed error in Weapon page, terrain damage was reported as a float, but should be a byte.
0.501	(added stuff from Jeff's e-mail - aldel) Fixed error in Texture page, related to procedural textures. The loop reading the palette went 256 iterations, when it should only go 255.
0.50	Initial Creation. Very rough.

A Tablefile is a collection of pages that define the properties of textures, weapons, doors, ships, sounds and objects in the Descent 3 universe. A page is a chunk of data that are the properties for one such building block. Some pages contain data primitives along with property data. A data primitive is the raw data needed to define the building block. For instance, one type of primitive for objects is the .OOF (Outrage Object Format) file for the polymodel to be associated with the object. Byte order is little endian (INTEL format) unless stated otherwise.

Each page contains it's own version number. There is no versioning information for the Tablefile in general. Each page is responsible for it's own version control.

The following is a list of page types that you will find in a Tablefile, along with the value associated with them:

PageType	***Value***	***Description***
PAGETYPE_UNKNOWN	0	An empty page. Usually used to create an empty Tablefile. Also is usually the first page found in a Tablefile.
PAGETYPE_TEXTURE	1	Contains data and properties for a texture.
PAGETYPE_WEAPON	2	Contains data and properties for a weapon (player or robot).
PAGETYPE_DOOR	5	Contains data and properties for a door
PAGETYPE_SHIP	6	Contains data and properties for a player ship.
PAGETYPE_SOUND	7	Contains data and properties for a sound.
PAGETYPE_GAMEFILE	9	Contains data and properties for a game file. Game files are any kind of file that may be used by Descent 3 in some way. Any extra files (that are not a data primitive) in the D3.HOG file are enumerated in these pages.
PAGETYPE_GENERIC	10	Contains data and properties for a generic object. A generic object is an object of type robot, powerup, clutter or building.

The following page types are outdated and no longer supported. They are listed here because it is possible they may be in a Tablefile. Should you come across one of these pagetypes, it should just be skipped over:

PageType	**Value**
PAGETYPE_ROBOT	3
PAGETYPE_POWERUP	4
PAGETYPE_MEGACELL	8

TABLEFILE STRUCTURE

A Tablefile is a collection of the above listed pages formatted in the following way:

‚Offset' is the offset from the start of the file (for Page 0) or the offset from the end of the previous page.

Offset	*Size*	*Description*
0	1 byte	Page type (see above)
1	4 byte int	Page length in bytes (which includes the 4-byte length)
5	(x-4) bytes	Page data

This pattern is continued until the EOF (End of File). Pages are listed in no specific order, usually a PAGETYPE_UNKNOWN page will be page 0. For optimal performance (in Descent 3) it is suggested that pages that are dependent on another page come after those pages it is dependent on. The easiest way to handle this is to sort the pages by pagetype in the following way (from beginning of the file to the end):

```
PAGETYPE_TEXTURE
PAGETYPE_SOUND
PAGETYPE_WEAPON
PAGETYPE_GENERIC
PAGETYPE_DOOR
PAGETYPE_SHIP
PAGETYPE_GAMEFILE
```

PSUEDOCODE FOR TABLEFILE READING

```
FILE *file;

// open the table file
file = fopen(filename,"rb");

while (!feof(file))
{
      unsigned char ptype=FILE_READBYTE(file);
      int length=FILE_READINT(file);

      switch(ptype)
      {
            case PAGETYPE_TEXTURE:
                  tablefile_ReadTexturePage(file); break;
            case PAGETYPE_DOOR:
                  tablefile_ReadDoorPage(file); break;
            case PAGETYPE_SOUND:
                  tablefile_ReadSoundPage(file); break;
            case PAGETYPE_GENERIC:
                  tablefile_ReadGenericPage(file); break;
            case PAGETYPE_GAMEFILE:
                  tablefile_ReadGamefilePage(file); break;
            case PAGETYPE_SHIP:
                  tablefile_ReadShipPage(file); break;
            case PAGETYPE_WEAPON:
                  tablefile_ReadWeaponPage(file); break;
            /* old pagetypes, no longer valid
               and not supported */
            case PAGETYPE_ROBOT:
            case PAGETYPE_POWERUP:
            case PAGETYPE_MEGACELL:
            case PAGETYPE_UNKNOWN:
                  fseek(file,length-sizeof(int),SEEK_CUR); break;
      }
}
fclose(file);
```

THIS DOCUMENT IS NOT AS COMPLETE AS IT POSSIBLY COULD BE. IT IS RECOMMENDED THAT IF YOU ARE WRITING A TABLEFILE EDITOR/CREATOR OF SOME SORT TO ANALYZE THE VALUES OF THE DATA OF THE OBJECTS, TEXTURES, SOUNDS, ETC. IN THE D3.HOG IN YOUR DESCENT 3 DIRECTORY. DOING THIS, YOU WILL BE ABLE TO GET A GOOD IDEA OF THE RANGES OF THE VARIABLES AND WHAT FLAGS/SETTINGS DO WHAT.

Page Type Description

NOTE: For the remainder of the document:

the symbol "NAME"	represents a value of
"PAGENAME_LEN"	35
"MAX_MODULENAME_LEN"	32
"MAX_PLAYER_WEAPONS"	21
"MAX_WB_GUNPOINTS"	8
"MAX_WB_FIRING_MASKS"	8
"MAX_PROC_ELEMENTS"	8000
"MAX_WEAPON_SOUNDS"	7
"MAX_DSPEW_TYPES"	2
"NUM_MOVEMENT_CLASSES"	5
"NUM_ANIMS_PER_CLASS"	24
"MAX_WBS_PER_OBJ"	21
"MAX_OBJ_SOUNDS"	2
"MAX_AI_SOUNDS"	5
"MAX_DEATH_TYPES"	4

Data Size's and Description

Type	Size
`byte`	1 byte
`short`	2 bytes
`int`	4 bytes
`float`	4 bytes
`vector`	3 floats (x,y,z)
`matrix`	3 vectors (right,up,forward)
`string`	\0 (NULL) terminated string, of max length specified (max length includes the NULL terminator).
`physics_chunk`	physics information chunk. See "Physics Chunks".
`lighting_chunk`	lighting information chunk. See "Lighting Chunks".
`weapon_battery_chunk`	weapon battery chunk. See "Weapon Battery Chunks".

PAGETYPE_DOOR

Current Version: 3

`Data`

Type	Name	Description
`short`	Version	Version of the door page. Used for version control.
`string`	Door Name	Used by Descent 3 to refer to this door by name. max length = PAGENAME_LEN
`string`	Model Name	Filename of the .OOF file used for the polymodel of the door. max length = PAGENAME_LEN
`float`	Opening Time	Time in seconds it takes for the door to open
`float`	Close Time	Time in seconds it takes for the door to close
`float`	Open Time	Time in seconds the door stays open
`byte`	Flags	See "Door Flags" section
`short`	Hit Points	How many hitpoints this door has (for destroyable doors)
`string`	Open Sound	D3 Sound Page name for the sound to play when this door opens. max length = PAGENAME_LEN
`string`	Close Sound	D3 Sound Page name for the sound to play when this door closes. max length = PAGENAME_LEN
`string`	Module Name	Filename of the OSIRIS script for this door. This filename doesn't need an extension since it is dependent on the platform that Descent 3 is running for (.DLL for Win32, .so for Linux, etc.). If this is not set then the door will not open! By default this can be set to "Generic", which contains a default generic script for any door. max length = MAX_MODULENAME_LEN

(END OF PAGE)

Door Flags

Name	Value	Description
DF_BLASTABLE	1	this door can be destroyed
DF_SEETHROUGH	2	this door can be seen through even when closed

PAGETYPE_GAMEFILE

Current Version: 2

`Data`

Type	Name	Description
`short`	Version	Version of the gamefile page
`string`	Filename	Filename of the game file. max length = `PAGENAME_LEN`
`string`	Directory	Data directory where file should be located. This is used for Descent 3 development only. max length = `PAGENAME_LEN`

(END OF PAGE)

NOTE: Third party developers usually don't have to be concerned with Gamefile pages, as they were used for Descent 3 development. The information is given for completeness purposes.

PAGETYPE_GENERIC

Current Version: 27

Data

Type	Name	Description
short	Version	Used for version control of the page
byte	Object Type	Type of generic object: OBJ_ROBOT 2 OBJ_POWERUP 7 OBJ_CLUTTER 11 OBJ_BUILDING 16
string	Name	Table Name of the object. max length = PAGENAME_LEN
string	Image Name	High resolution .OOF filename for the object polymodel. max length = PAGENAME_LEN
string	Med. Image Name	Medium resolution .OOF filename for the object polymodel. max length = PAGENAME_LEN
string	Low Image Name	Low resolution .OOF filename for the object polymodel. max length = PAGENAME_LEN
float	Impact Size	
float	Impact Time	
float	Damage	
short	Score	Score gained by the player by destroying this object
-- The following section is for objects of type OBJ_POWERUP only --		
short	Ammo Count	How much ammo this powerup contains by default
-- End of section --		
string	NO LONGER USED	Just ignore this string. max length = PAGENAME_LEN
string	Module Name	The filename of the OSIRIS module that contains the default script for this object. No file extension needed as it depends on the platform Descent 3 is running on, as to what the real file extension is (Win32 = DLL, Linux = so, etc.). If this is not set, the object will have no default behavior. Usually this is set to "Generic". max length = MAX_MODULENAME_LEN
string	Script Name Override	Usually when the OSIRIS module (see Module Name) is queried for a script for this object, the Name is passed in for it to use for comparison. However, sometimes, you may want multiple objects to have the same behavior. By creating a behavior, giving it a name and putting the script in the OSIRIS module 'Module Name', you can set the Script Name Override to override the Name passed in to the module. max length = PAGENAME_LEN.
byte	Has Description	If this is 0, then the object has no inventory description. Else it does.
-- The following section of code is only if the object has a description --		
string	Description	Description for the object when it is in the inventory. Not used. max length = 1024 bytes.
-- End of Section --		
string	Icon Name	Not Used (make it a 0 length string). max length = PAGENAME_LEN
float	Medium LOD Distance	Distance at which the medium resolution model is used
float	Low LOD Distance	Distance at which the low resolution model is used
physics_chunk	Physics Information	
float	Size	Object size

`lighting_chunk`	Lighting Information	
`int`	Hit Points	The amount of shields this object has by default
`int`	Flags	See "Object Flags".
`int`	AI Flags	See "Object AI Flags".
`byte`	AI Class	AI Class for object. See "Object AI Classes".
`byte`	AI Type	AI Type for object. See "Object AI Types".
`byte`	AI Movement Type	AI Movement Type for object. See "Object AI Movement Types".
`byte`	AI Movement SubTypes	AI Movement subtypes for object. See "Object AI Movement Types".
`float`	AI FOV	cos() of the AI Field of View for object.
`float`	AI Max Velocity	Maximum velocity
`float`	AI Max Delta Velocity	Maximum delta velocity
`float`	AI Max Turn Rate	Maximum Turn Rate
`int`	AI Notify Flags	Flags for AI notification. See "Object AI Notification Flags".
`float`	AI Max Delta Turn Rate	Maximum Delta Turn Rate
`float`	AI Circle Distance	Distance the object should circle it's targets at
`float`	AI Attack Vel Percentage	
`float`	AI Dodge Percentage	
`float`	AI Dodge Vel Percentage	
`float`	AI Flee Vel Percentage	
`float`	AI Melee Damage 1	
`float`	AI Melee Damage 2	
`float`	AI Melee Latency 1	
`float`	AI Melee Latency 2	
`float`	AI Curiousity	
`float`	AI Night Vision	
`float`	AI Fog Vision	
`float`	AI Lead Accuracy	
`float`	AI Lead Varience	
`float`	AI Fire Spread	
`float`	AI Fight Team	
`float`	AI Fight Same	
`float`	AI Aggression	
`float`	AI Hearing	
`float`	AI Frustration	
`float`	AI Roaming	
`float`	AI Life Preservation	
`float`	AI Avoid Friends Distance	
`float`	AI Biased Flight Importance	
`float`	AI Biased Flight Min	
`float`	AI Biased Flight Max	

-- The following section is repeated MAX_DSPEW_TYPES times --

byte	Destroy Spew Flags	See "Destroy Spew Flags"
float	Destroy Spew Percent	Percent Chance that this object would spew
short	Destroy Spew Number	How many of this object to spew if we are to spew it.
string	Destroy Spew Object	Table name of the object to spew (for this slot). max length = PAGENAME_LEN.

-- End of Section --

-- The following section is repeated NUM_MOVEMENT_CLASSES times --

-- The following section is repeated NUM_ANIMS_PER_CLASS times --

short	From Frame	Starting frame of animation
short	To Frame	End frame of animation
float	Seconds Per Class	How long the animation should last

-- End of Section --

-- End of Section --

-- The following section is repeated MAX_WBS_PER_OBJ times --

weapon_battery_chunk	Weapon Battery Information	

-- End of Section --

-- The following section is repeated MAX_WBS_PER_OBJ times --

-- The following section is repeated MAX_WB_GUNPOINTS times --

string	Weapon Name	Table name of the weapon associated with this gunpoint, on this weapon battery. max length = PAGENAME_LEN.
short	To Frame	End frame of animation
float	Seconds Per Class	How long the animation should last

-- End of Section --

-- End of Section --

-- The following section is repeated MAX_OBJ_SOUNDS times --

string	Sound Name	Table Sound Name. max length = PAGENAME_LEN

-- End of Section --

-- The following section is repeated MAX_AI_SOUNDS times --

string	AI Sound Name	Table Sound Name. max length = PAGENAME_LEN

-- End of Section --

-- The following section is repeated MAX_WBS_PER_OBJ times --

-- The following section is repeated MAX_WB_FIRING_MASKS times --

string	Fire Sound Name	Table Sound Name for fire sound. max length = PAGENAME_LEN

-- End of Section --

-- End of Section --

-- The following section is repeated NUM_MOVEMENT_CLASSES times --

-- The following section is repeated NUM_ANIMS_PER_CLASS times --

string	Animation Sound Name	Table Sound Name for this animation. max length = PAGENAME_LEN.

-- End of Section --

-- End of Section --

float	Respawn Scalar	Used for powerups to scale how often they respawn in multiplayer games. Normal is 1.0.
short	Number of Death Types	Number of types of deaths possible for this object. Maximum value is MAX_DEATH_TYPES.

-- The following section is repeated 'Number of Death Types' times --

`int`	Death Flags	flags for this death type. See "Death Type Flags".
`float`	Death Delay Min.	Minimum amount of time for this death to last.
`float`	Death Delay Max.	Maximum amount of time for this death to last.
`byte`	Death Probability	Probability of this death happening (0 to 100, the probabilites for all the death types should add up to 100).

-- End of Section --

(END OF PAGE)

Object Flags

Name	Value	Description
OIF_CONTROL_AI	0x01	//this object uses AI
OIF_USES_PHYSICS	0x02	//this object uses physics
OIF_DESTROYABLE	0x04	//this object can be destroyed
OIF_INVEN_SELECTABLE	0x08	//this object can be selected in the inventory
OIF_INVEN_NONUSEABLE	0x10	//this object can not be used by pressing ENTER during the game
OIF_INVEN_TYPE_MISSION	0x20	//this object is for Mission objectives
OIF_INVEN_NOREMOVE	0x40	//this object should NOT be removed from the inventory when used
OIF_INVEN_VISWHENUSED	0x80	//this object will not have it contol type, movement type and render types
OIF_AI_SCRIPTED_DEATH	0x100	
OIF_DO_CEILING_CHECK	0x200	
OIF_IGNORE_FORCEFIELDS_AND_GLASS	0x400	
OIF_NO_DIFF_SCALE_DAMAGE	0x800	
OIF_NO_DIFF_SCALE_MOVE	0x1000	
OIF_AMBIENT_OBJECT	0x2000	//this object is just for show, & can be removed to improve performance

Object AI Flags

Name	Value	Name	Value
AIF_WEAPON1	0x00000001	AIF_TEAM_PTMC	0x00000000
AIF_WEAPON2	0x00000002	AIF_TEAM_REBEL	0x00010000
AIF_MELEE1	0x00000004	AIF_TEAM_HOSTILE	0x00020000
AIF_MELEE2	0x00000008	AIF_TEAM_NEUTRAL	0x00030000
AIF_STAYS_INOUT	0x00000010	AIF_ORDERED_WB_FIRING	0x00040000
AIF_GB_MIMIC_PLAYER_FIRING_HACK	0x00000010	AIF_ORIENT_TO_VEL	0x00080000
AIF_ACT_AS_NEUTRAL_UNTIL_SHOT	0x00000020	AIF_XZ_DIST	0x00100000
AIF_PERSISTANT	0x00000040	AIF_REPORT_NEW_ORIENT	0x00200000
AIF_DODGE	0x00000080	AIF_TARGET_BY_DIST	0x00400000
AIF_FIRE	0x00000100	AIF_DISABLE_FIRING	0x00800000
AIF_FLINCH	0x00000200	AIF_DISABLE_MELEE	0x01000000
AIF_DETERMINE_TARGET	0x00000400	AIF_AUTO_AVOID_FRIENDS	0x02000000
AIF_AIM	0x00000800	AIF_TRACK_CLOSEST_2_FRIENDS	0x04000000
AIF_ONLY_TAUNT_AT_DEATH	0x00001000	AIF_TRACK_CLOSEST_2_ENEMIES	0x08000000
AIF_AVOID_WALLS	0x00002000	AIF_BIASED_FLIGHT_HEIGHT	0x10000000
AIF_DISABLED	0x00004000	AIF_FORCE_AWARENESS	0x20000000
AIF_FLUCTUATE_SPEED_PROPERTIES	0x00008000	AIF_UVEC_FOV	0x40000000
AIF_TEAM_MASK	0x00030000	AIF_AIM_PNT_FOV	0x80000000

Object AI Classes

Name	Value
AIC_STATIC	0
AIC_PURE_PATH	1
AIC_AIS_FULL	2

Object AI Types

Name	Value
AIT_FLYLANDER	0
AIT_STALKER	1
AIT_EVADER1	2
AIT_EVADER2	3
AIT_STATIONARY_TURRET	4
AIT_AIS	5
AIT_MELEE1	6
AIT_BIRD_FLOCK1	7
AIT_HERD1	8

Object AI Movement Types

Name	Value	Description
// AI Movement flying types		
AIMF_NORMAL	0	
AIMF_PATH	1	
AIMF_HELICOPTER	2	
AIMF_HOVERCRAFT	3	
AIMF_JET	4	
AIMF_PLAYERLIKE	5	// For NPC ships
AIMF_BUDDYLIKE	6	// For theif/buddy bots
// AI Movement walking types		
AIMW_RESTRICTED_FLAT	0	// Specify a min/max angle of incline (so we can restrict movement -- i.e we can even do ceiling only robots)
AIMW_RESTRICTED_LOW_ANGLE	1	
AIMW_RESTRICTED_HIGH_ANGLE	2	
AIMW_NON_RESTRICTED	3	
AIMW_UNDERWATERONLY	4	// Stays in water
AIMW_WATERSURFACE	5	

Object AI Notification Flags

Name	Value	Description
AIN_NEW_MOVEMENT	(1<<1)	
AIN_OBJ_KILLED	(1<<2)	
AIN_WHIT_BY_OBJ	(1<<3)	
AIN_SEE_TARGET	(1<<4)	
AIN_PLAYER_SEES_YOU	(1<<5)	
AIN_WHIT_OBJECT	(1<<6)	
AIN_TARGET_DIED	(1<<7)	// In code, it only notifies if the target is gone
AIN_OBJ_FIRED	(1<<8)	
AIN_GOAL_COMPLETE	(1<<9)	
AIN_GOAL_FAIL	(1<<10)	
AIN_GOAL_ERROR	(1<<11)	
AIN_HEAR_NOISE	(1<<12)	
AIN_NEAR_TARGET	(1<<13)	
AIN_HIT_BY_WEAPON	(1<<14)	

AIN_NEAR_WALL	(1<<15)	
AIN_USER_DEFINED	(1<<16)	// Processed in script
AIN_TARGET_INVALID	(1<<17)	// Goal is killed
AIN_GOAL_INVALID	(1<<18)	
AIN_SCRIPTED_GOAL	(1<<19)	
AIN_SCRIPTED_ENABLER	(1<<20)	
AIN_ANIM_COMPLETE	(1<<21)	
AIN_BUMPED_OBJ	(1<<22)	
AIN_MELEE_HIT	(1<<23)	
AIN_MELEE_ATTACK_FRAME	(1<<24)	
AIN_SCRIPTED_INFLUENCE	(1<<25)	
AIN_SCRIPTED_ORIENT	(1<<26)	
AIN_MOVIE_START	(1<<27)	
AIN_MOVIE_END	(1<<28)	
AI_NOTIFIES_ALWAYS_ON	(AIN_ANIM_COMPLETE | AIN_NEW_MOVEMENT | AIN_PLAYER_SEES_YOU | AIN_GOAL_COMPLETE | AIN_GOAL_FAIL | AIN_GOAL_ERROR | AIN_USER_DEFINED | AIN_TARGET_DIED | AIN_TARGET_INVALID | AIN_BUMPED_OBJ | AIN_MELEE_HIT | AIN_MELEE_ATTACK_FRAME | AIN_TARGET_INVALID)	

Destroy Spew Flags

Name	Value
DSF_ONLY_IF_PLAYER_HAS_OBJ_1	1
DSF_ONLY_IF_NO_1	2

Death Type Flags

Name	Value	Description
//Not a bit flag, but an override value		
DF_DEFAULT	-1	//use the default death for this object
DF_UNUSED	0x0000001	//Unused flag
//Delay options		
DF_DELAY_FROM_ANIM	0x0000002	//delay time from death animation
DF_DELAY_SPARKS	0x0000004	//delay with sparks
DF_DELAY_LOSES_ANTIGRAV	0x0000008	//object gets gravity during delay
DF_DELAY_SMOKES	0x0000010	//delay with smoke
DF_DELAY_FLYING	0x0100000	//delay with object flying up into the air
DF_DELAY_FIREBALL	0x0200000	//delay with fireballs
DF_DELAY_FADE_AWAY	0x0400000	//fade away
DF_DELAY_NO_TUMBLE_FLY	0x2000000	//don't tumble while flying up in the air
//Options for what happens on death		
DF_FIREBALL	0x0000020	//there are fireballs when the object dies
DF_BREAKS_APART	0x0000040	//the object breaks into pieces when it dies
DF_BLAST_RING	0x0000080	//a blast ring is created when the object dies
DF_REMAINS	0x0000100	//the object does not go away when it does
DF_LOSES_ANTIGRAV	0x0000200	//object gets gravity on death
DF_FADE_AWAY	0x0800000	//fades away
//Explosion size if there is a death fireball		
DF_EXPL_SMALL	0x0000000	//use a small explosion
DF_EXPL_MEDIUM	0x0000400	//use a medium explosion
DF_EXPL_LARGE	0x0000800	//use a large explosion
DF_EXPL_SIZE_MASK	0x0000c00	
DF_EXPL_SIZE_SHIFT	10	
//What happens when this object hits something during delay		
DF_CONTACT_FIREBALL	0x0001000	//creates fireballs
DF_CONTACT_BREAKS_APART	0x0002000	//break apart
DF_CONTACT_BLAST_RING	0x0004000	//blast ring
DF_CONTACT_REMAINS	0x0008000	//stays around
//Whether the debris puts off smoke		
DF_DEBRIS_SMOKES	0x0010000	//the debris that's created smokes
//What happens to the debris when it times out or hits something		
DF_DEBRIS_FIREBALL	0x0020000	//creates fireballs
DF_DEBRIS_BLAST_RING	0x0040000	//blast ring
DF_DEBRIS_REMAINS	0x0080000	//stays around
//Sound option		
DF_DELAY_SOUND	0x1000000	//play sound at start of fade

PAGETYPE_SHIP

Current Version: 6

Data

Type	Name	Description
short	Version	Version of the ship page. Used for version control.
string	Ship Name	Descent 3 name of the ship. max length = PAGENAME_LEN
string	Cockpit Filename	Filename of the .OOF model for the cockpit. max length = PAGENAME_LEN
string	HUD Config Filename	.INF filename of the HUD configuration file. max length = PAGENAME_LEN
string	Model Name	Filename for the .OOF model that is the model of the ship. max length = PAGENAME_LEN
string	Dying Model	Filename for the .OOF model of the ship when it is dying. max length = PAGENAME_LEN
string	Med. Model Name	Filename for the .OOF model that is the medium res (less poly) version. max length = PAGENAME_LEN
string	Low Model Name	Filename for the .OOF model that is the low res (really low poly) version. max length = PAGENAME_LEN
float	Med. LOD Distance	Distance at which the medium res model version of the ship is used
float	Low LOD Distance	Distance at which the low res model version of the ship is used
physics_chunk	Physics Information	The physics information for the ship
float	Size	Size of the radius of the ship
float	Armor Scalar	Armor scalar for the ship (Pyro-GL = 1.0f)
int	Flags	Ship flags. See "Ship Flags" section.

-- The following section is repeated MAX_PLAYER_WEAPONS times --

byte	Fire Flags	Firing flags for this weapon. See "Ship Firing Flags" section.
string	Fire Sound Name	D3 Tablefile Sound name for the sound to play when firing. max length = PAGENAME_LEN.
string	Release Sound Name	D3 Tablefile Sound name for the sound to play when the trigger is released. max length = PAGENAME_LEN.
string	Spew Powerup Name	The generic object name of the powerup object to spew out for this weapon if the ship is destroyed and the weapon is to be spewed. max length = PAGENAME_LEN.
int	Max Ammo	Maximum amount of ammo this ship can carry for this weapon.
weapon_battery_chunk	Weapon Battery Information	This chunk of data contains the weapon battery information for this weapon.

-- The following section is repeated MAX_WB_GUNPOINTS times --

string	Fire Sound Name	The D3 Tablefile sound name for the sound to play for this weapon for when this gunpoint is firing the weapon. max length = PAGENAME_LEN.

-- End of Section --

-- The following section is repeated MAX_WB_FIRING_MASKS times --

string	Weapon Name	The D3 Tablefile weapon name for the weapon associated with this weapon battery slot. max length = PAGENAME_LEN.

-- End of Section --

-- End of Section --

(END OF PAGE)

Ship Flags

Name	Value	Description
SF_DEFAULT_ALLOW	1	//Allowed by default

Ship Firing Flags

Name	Value	Description
SFF_FUSION	1	// fires like fusion
SFF_ZOOM	4	// Zooms in
SFF_TENTHS	8	// Ammo displays in tenths

PAGETYPE_SOUND

Current Version: 1

Data

Type	Name	Description
short	Version	Version number, used for version control.
string	Name	D3 Tablefile name of this sound. max length = PAGENAME_LEN.
string	Filename	.WAV file for this sound. max length = PAGENAME_LEN.
int	Flags	Sound flags. See section "Sound Flags".
int	Loop Start	Sample value for start of a loop (looping sounds)
int	Loop End	Sample value for end of a loop (looping sounds)
float	Outer Cone Volume	Percentage of volume for outer cone.
int	Inner Cone Angle	Degrees for inner cone.
int	Outer Cone Angle	Degrees for outer cone.
float	Max Distance	Maximum sound clipping distance.
float	Min Distance	Minimum sound clipping distance.
float	Import Adjust	Percentage of original .WAV file sound volume to use when loading the sound.

(END OF PAGE)

Sound Flags

Name	Value	Description
SPF_LOOPED	1	// Sound is looped
SPF_FIXED_FREQ	2	// No doppler shift
SPF_OBJ_UPDATE	4	// Sound updates with attached object movements
SPF_FOREVER	8	// Always plays in high-level, this flag should be ignored in low-level
SPF_PLAYS_EXCLUSIVELY	16	
SPF_PLAYS_ONCE	32	
SPF_USE_CONE	64	
SPF_LISTENER_UPDATE	128	// Sound updates with listener movements
SPF_ONCE_PER_OBJ	256	

PAGETYPE_TEXTURE

Current Version: 7

Data

Type	Name	Description
short	Version	Version of this page, used for versioning control.
string	Name	D3 Tablefile name for this texture. max length = PAGENAME_LEN
string	Bitmap Name	Filename of the .OGF file for this texture. max length = PAGENAME_LEN.
string	Destroy Name	Filename of the .OGF file to be used as a 'destroyed' version of this texture. max length = PAGENAME_LEN.
float	Red	Red component of the light this texture gives off (0 to 1.0)
float	Green	Green component of the light this texture gives off (0 to 1.0)
float	Blue	Blue component of the light this texture gives off (0 to 1.0)
float	Alpha	Alpha value of the entire texture (0 to 1.0, 0 being transparent).
float	Speed	Speed of the texture (lets it slide) (0 to 1.0)
float	U slide	Speed of the sliding of this texture in the U direction.
float	V slide	Speed of the sliding of this texture in the V direction.
float	Reflectivity	Reflectivity value of this texture, used during radiosity lighting (0 to 1.0)
byte	Corona Style	Which type of corona this texture should give off (used if the texture gives off light). A, B, C or D.
int	Damage	Amount of damage per second this texture does if an object collides with it.
int	Flags	Texture flags. See "Texture Flags" section.
-- The following section is to be read only if the texture is a procedural texture --		
// Palette section		
-- The following section is to be repeated 255 times --		
short	16 bit color	Color entry for this palette color
-- End of section --		
byte	Heat	'Heat' intensity
byte	Light	???
byte	Thickness	???
float	Evaluation Time	???
float	Osc. Time	???
byte	Osc. Value	???
short	Num. Proc Elements	Number of procedural elements. max value = MAX_PROC_ELEMENTS
-- The following section is to be repeated 'Num. Proc Elements' times --		
byte	Type	Type of procedural element. See "Procedural element types" section.
byte	Freq.	Frequency of procedural element
byte	Speed	Speed of procedural element
byte	Size	Size of procedural element
byte	x1	X1 value for procedural element
byte	y1	Y1 value for procedural element
byte	x2	X2 value for procedural element
byte	y2	Y2 value for procedural element
-- End of Section --		
-- End of Section --		
string	Sound Name	D3 Table Name of the sound to be associated with this texture. max length = PAGENAME_LEN.
float	Sound Volume	Volume of the sound

(END OF PAGE)

Texture Flags

Name	Value	Description
TF_VOLATILE	1	
TF_WATER	(1<<1)	
TF_METAL	(1<<2)	// Shines like metal
TF_MARBLE	(1<<3)	// Shines like marble
TF_PLASTIC	(1<<4)	// Shines like plastic
TF_FORCEFIELD	(1<<5)	
TF_ANIMATED	(1<<6)	// Set for animated textures (.OAF files)
TF_DESTROYABLE	(1<<7)	
TF_EFFECT	(1<<8)	
TF_HUD_COCKPIT	(1<<9)	
TF_MINE	(1<<10)	
TF_TERRAIN	(1<<11)	
TF_OBJECT	(1<<12)	
TF_TEXTURE_64	(1<<13)	
TF_TMAP2	(1<<14)	
TF_TEXTURE_32	(1<<15)	
TF_FLY_THRU	(1<<16)	
TF_PASS_THRU	(1<<17)	
TF_PING_PONG	(1<<18)	
TF_LIGHT	(1<<19)	
TF_BREAKABLE	(1<<20)	// Breakable (as in glass)
TF_SATURATE	(1<<21)	
TF_ALPHA	(1<<22)	
TF_DONTUSE	(1<<23)	
TF_PROCEDURAL	(1<<24)	
TF_WATER_PROCEDURAL	(1<<25)	
TF_FORCE_LIGHTMAP	(1<<26)	
TF_SATURATE_LIGHTMAP	(1<<27)	
TF_TEXTURE_256	(1<<28)	
TF_LAVA	(1<<29)	
TF_RUBBLE	(1<<30)	
TF_SMOOTH_SPECULAR	(1<<31)	
TF_TEXTURE_TYPES	(TF_MINE + TF_TERRAIN + TF_OBJECT + TF_EFFECT + TF_HUD_COCKPIT + TF_LIGHT)	
TF_SPECULAR	(TF_METAL + TF_MARBLE | TF_PLASTIC)	

Procedural element types

Water procedurals:

Name	Value
PROC_WATER_NONE	0
PROC_WATER_HEIGHT_BLOB	1
PROC_WATER_SINE_BLOB	2
PROC_WATER_RAINDROPS	3
PROC_WATER_BLOBDROPS	4

Fire (non-water) procedurals:

Name	Value
PROC_NONE	0
PROC_LINE_LIGHTNING	1
PROC_SPHERE_LIGHTNING	2
PROC_STRAIGHT	3
PROC_RISING_EMBER	4
PROC_RANDOM_EMBERS	5
PROC_SPINNERS	6
PROC_ROAMERS	7
PROC_FOUNTAIN	8
PROC_CONE	9
PROC_FALL_RIGHT	10
PROC_FALL_LEFT	11

PAGETYPE_WEAPON

Current Version: 8

Data

Type	Name	Description
short	Version	Version of this page, used for versioning control.
string	Name	D3 Table name of the weapon. max length = PAGENAME_LEN.
string	HUD Image	Filename of the HUD icon for this weapon. max length = PAGENAME_LEN.
string	Fire Image	Filename of the .OOF file used for the actual weapon model. max length = PAGENAME_LEN. (May be an .OGF if the WF_IMAGE_BITMAP flag is set)
string	Particle Name	Texture Name of the image to use as particles. max length = PAGENAME_LEN.
byte	Particle Count	How many particles the weapon creates
float	Particle Life	The lifetime of each particle
float	Particle Size	The size of each particle
int	Weapon Flags	See "Weapon Flags" section.
string	Spawn Name	Name of the weapon this weapon spawns. max length = PAGENAME_LEN.
byte	Spawn Count	How many weapons to spawn.
string	Robot Spawn	Name of the robot object to spawn on weapon death. max length = PAGENAME_LEN.
string	Alt. Spawn Name	An alternative weapon to spawn. max length = PAGENAME_LEN.
byte	Alt. Spawn Chance	Percentage chance (0 - 100) that the alternate weapon will be spawned.
float	Gravity Time	How long to create a gravity field (see BlackShark)
float	Gravity Size	How big of a radius is the gravity field.
float	Homing FOV	Cos of the Field of View for homing weapons.
float	Custom Size	Custom size of weapon (to override polymodel size)
float	Size	Weapon 'blob' size.
float	Thrust Time	How long this weapon thrusts
physics_chunk	Physics Information	
float	Terrain Damage Size	Radius of terrain damage this weapon does to the terrain
byte	Terrain Damage Depth	How deep is the damage on the terrain
float	Weapon Alpha	How alpha'd the weapon is (0 to 1.0, 0 being transparent)
string	Explosion Image	D3 Texture Name of the texture to use when the weapon explodes. max length = PAGENAME_LEN.
float	Explode Time	How long the explosion for this weapon lasts
float	Explode Size	Size of the explosion
float	Player Damage	How much damage this weapon does to a player.
float	Object Damage	How much damage this weapon does to a generic object.
float	Impact Size	The radius of the sphere used when the weapon impacts and explodes.
float	Impact Time	How long the impact affect lasts.
float	Impact Player Damage	How much damage the impact explosion does to a player.
float	Impact Generic Damage	How much damage the impact explosion does to a generic object.
float	Impact Force	Amount of force caused by the impact explosion.
float	Lifetime	How long the weapon lives
lighting_chunk	Lighting Information	
float	Recoil Force	Recoil force applied to the object (used in Force Feedback also)

-- The following section is repeated MAX_WEAPON_SOUNDS times –		
string	Sound Name	D3 Table Sound name of the sound associated with this sound slot. max length = PAGENAME_LEN.
-- End of Section		
string	Smoke Name	D3 Texture Name of the texture to use for the smoke trail image
string	Scorch Image	D3 Texture Name of the texture to use for the scorch mark left by this weapon. max length = PAGENAME_LEN.
float	Scorch Size	Size of the scorch mark to leave
string	Icon Name	name of the icon for this weapon. max length = PAGENAME_LEN

(END OF PAGE)

Weapon Flags

Name	Value	Description
WF_HUD_ANIMATED	(1<<0)	
WF_IMAGE_BITMAP	(1<<1)	// whether or not the firing image is a bitmap or model
WF_SMOKE	(1<<2)	// Weapon drops smoke as it moves
WF_MATTER_WEAPON	(1<<3)	// This a matter weapon, as opposed to an energy weapon
WF_ELECTRICAL	(1<<4)	// This weapons fires as an electrical storm
WF_IMAGE_VCLIP	(1<<5)	// This weapon fire image is a vclip
WF_SPRAY	(1<<6)	// This weapon is a spray, like a flamethrower
WF_STREAMER	(1<<7)	// This weapon has a streamer effect attached
WF_INVISIBLE	(1<<8)	// This weapon is invisible
WF_RING	(1<<9)	// This weapon is drawn ring style
WF_SATURATE	(1<<10)	// Saturate this bitmap weapon
WF_BLAST_RING	(1<<11)	// Creates a blast ring upon explosion
WF_PLANAR_BLAST	(1<<12)	// Blast bitmap takes on the walls plane
WF_PLANAR	(1<<13)	// This weapon doesn't always face you
WF_ENABLE_CAMERA	(1<<14)	// This weapon can be used for missile camera
WF_SPAWNS_IMPACT	(1<<15)	// This weapon spawns others on impact
WF_SPAWNS_TIMEOUT	(1<<16)	// This weapon spawns others when it times out
WF_EXPAND	(1<<17)	// This weapon expands when exploding
WF_MUZZLE	(1<<18)	// This weapon produces a muzzle flash when fired
WF_MICROWAVE	(1<<19)	// This weapon makes a microwave effect on the victim
WF_NAPALM	(1<<20)	// This weapon does a napalm effect to objects it touches
WF_REVERSE_SMOKE	(1<<21)	// The smoke trail gets smaller as it ages
WF_GRAVITY_FIELD	(1<<22)	// This weapon has a gravity field
WF_COUNTERMEASURE	(1<<23)	// This weapon is a countermeasure
WF_SPAWNS_ROBOT	(1<<24)	// This weapon spawns a robot upon death
WF_FREEZE	(1<<25)	// This weapon slows a ship/object down
WF_TIMEOUT_WALL	(1<<26)	// This weapon times out like a wall hit
WF_PLANAR_SMOKE	(1<<27)	// This weapon has a planar smoke trail instead of a blob
WF_SILENT_HOMING	(1<<28)	// This weapon does not give a homing lock sound
WF_HOMING_SPLIT	(1<<29)	// This weapon homes when it splits
WF_NO_ROTATE	(1<<30)	// This weapon does not rotate as a bitmap
WF_CUSTOM_SIZE	(1<<31)	// This weapon uses a custom size

Physics Chunks

`Data`

Type	Name	Description
`float`	Mass	
`float`	Drag	
`float`	Full Thrust	
`int`	Flags	See "Physics Flags".
`float`	Rotational Drag	
`float`	Full Rotational Thrust	
`int`	Number of Bounces	Maximum number of bounces for this item (-1 for infinite)
`float`	Initial Velocity	
`vector`	Initial rotational velocity (3 floats: x,y,z)	
`float`	Wiggle Amplitude	
`float`	Wiggles Per Second	
`float`	CoEfficent Restitution	
`float`	Hit Death Angle	(the sin of the angle between 0 and 90)
`float`	Max Turn-Roll Rate	
`float`	Turn-Roll ratio	

(END OF CHUNK)

Physics Flags

Name	Value	Description
`PF_TURNROLL`	0x01	// Roll when turning
`PF_LEVELING`	0x02	// Level object with closest side
`PF_BOUNCE`	0x04	// Bounce (not slide) when hit will
`PF_WIGGLE`	0x08	// Wiggle while flying
`PF_STICK`	0x10	// Object sticks (stops moving) when hits wall
`PF_PERSISTENT`	0x20	// Object keeps going even after it hits another object (eg, fusion cannon)
`PF_USES_THRUST`	0x40	// This object uses its thrust
`PF_GRAVITY`	0x80	// Effected by gravity
`PF_IGNORE_OWN_CONC_FORCES`	0x100	// Ignore it's own concussion force
`PF_WIND`	0x200	// Effected by wind
`PF_USES_PARENT_VELOCITY`	0x400	
`PF_FIXED_VELOCITY`	0x800	
`PF_FIXED_ROT_VELOCITY`	0x1000	
`PF_NO_COLLIDE_PARENT`	0x2000	// this object cannot collide with its parent
`PF_HITS_SIBLINGS`	0x4000	// this object can collide with its siblings (like a bomb) // chrishack -- add flag to weapon page
`PF_REVERSE_GRAVITY`	0x8000	// this object flys upward with gravity
`PF_GRAVITY_MASK`	(`PF_REVERSE_GRAVITY | PF_GRAVITY`)	
`PF_NO_COLLIDE`	0x10000	// No collisions AND NO RELINKS -- DANGEROUS TO USE if not used correctly
`PF_NO_ROBOT_COLLISIONS`	0x20000	// No collisions occur with robots
`PF_POINT_COLLIDE_WALLS`	0x40000	// When colliding with walls, make our radius zero
`PF_HOMING`	0x80000	// This object (weapon) homes
`PF_GUIDED`	0x100000	// This object is guided
`PF_IGNORE_CONCUSSIVE_FORCES`	0x200000	

PF_DESTINATION_POS	0x400000	
PF_LOCK_X	0x800000	
PF_LOCK_Y	0x1000000	
PF_LOCK_Z	0x2000000	
PF_LOCK_P	0x4000000	
PF_LOCK_H	0x8000000	
PF_LOCK_B	0x10000000	
PF_LOCK_MASK	(PF_LOCK_X | PF_LOCK_Y | PF_LOCK_Z | PF_LOCK_P | PF_LOCK_H | PF_LOCK_B)	
PF_NEVER_USE_BIG_SPHERE	0x20000000	
PF_NO_SAME_COLLISIONS	0x40000000	
PF_NO_DOOR_COLLISIONS	0x80000000	

Lighting Chunks

Data

Type	Description
float	Light Distance (how far the light gets casted, in Units)
float	Start Red component (between 0 and 1.0)
float	Start Green component (between 0 and 1.0)
float	Start Blue component (between 0 and 1.0)
float	Time Interval
float	Flicker Distance
float	Directional Field of View (Part of Hemisphere, between 0 and 1.0 - this area will NOT be lit)
float	End Red component (between 0 and 1.0)
float	End Green component (between 0 and 1.0)
float	End Blue component (between 0 and 1.0)
int	Flags - see "Lighting Flags"
int	Time bits - ??
byte	Angle -- (NOT USED ??? - not working)
byte	Lighting Render Type: Static Lighting: 0 Gouraud Shading: 1 Lightmaps: 2

(END OF CHUNK)

Lighting Flags

Name	Value	Description
OLF_FLICKERING	1	
OLF_TIMEBITS	2	
OLF_PULSE	4	
OLF_PULSE_TO_SECOND	8	
OLF_FLICKER_SLIGHTLY	16	
OLF_DIRECTIONAL	32	// Directional light - casts light in a cone
OLF_NO_SPECULARITY	64	// Object does not have specular light cast on it

Weapon Battery Chunks

`Data`

Type	Name	Description
`float`	Energy Usage	Amount of energy used by this weapon battery slot
`float`	Ammo Usage	Amount of ammo used by this weapon battery slot
-- The following section is repeated `MAX_WB_GUNPOINTS` times --		
`short`	Gunpoint Weapon Index	???
-- End of Section --		
-- The following section is repeated `MAX_WB_FIRING_MASKS` times --		
`byte`	Fire Masks	
`float`	Fire Wait Time	
`float`	Animation Time	
`float`	Animation Start Frame	
`float`	Animation Fire Frame	
`float`	Animation End Frame	
-- End of Section --		
`byte`	Number Masks - ???	
`short`	Aiming GunPoint Index	Gunpoint to use for aiming with this battery
`byte`	Aiming Flags	???
`float`	Aiming Dot	???
`float`	Aiming Distance	???
`float`	Aiming XZ Dot	???
`short`	Flags	See "Weapon Battery Flags"
`byte`	Quad Fire Mask	Firing mask override to use when this weapon battery is set with the Quad fire flag.

Fire masks: this is a bit mask as to what gunpoints the given weapon battery should fire from when used. Like if the value is 0x32 Then the firing mask for that weapon battery is gunpoints: 0011 0010 or 1,4 and 5 will fire when this weapon battery is fired.

Weapon Battery Flags

Name	Value
`WBF_SPRAY`	1
`WBF_ANIM_LOCAL`	2
`WBF_ANIM_FULL`	4
`WBF_ANIM_MASKS`	6
`WBF_RANDOM_FIRE_ORDER`	8
`WBF_GUIDED`	16
`WBF_USE_CUSTOM_FOV`	32
`WBF_ON_OFF`	64
`WBF_USE_CUSTOM_MAX_DIST`	128
`WBF_USER_TIMEOUT`	256
`WBF_FIRE_FVEC`	512
`WBF_AIM_FVEC`	1024
`WBF_FIRE_TARGET`	2048

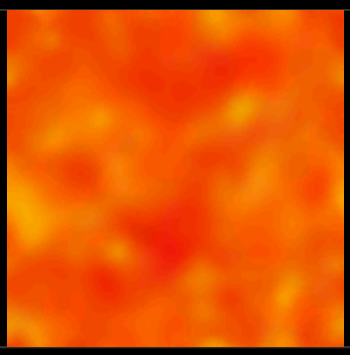
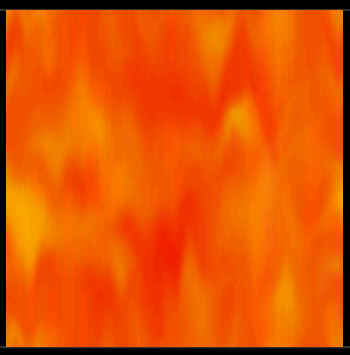
Texturenkatalog - das überleg' ich mir aber noch...

RagilRal

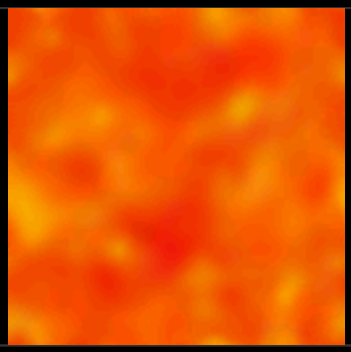
Oftmals entspricht die Optik einer Textur nicht so ganz dem was man sich ursprünglich erwartet hat, da sie ingame bzw. in voller Größe dann doch etwas anders aussieht als gedacht. Daher, hier (fast) alle Stock-Textures von D3. Nicht dabei sind jene, die im Editor die Test-Texture zeigen und jene, die ich eventuell übersehen habe. Oh my, was tu ich mir da bloß an... 🙂

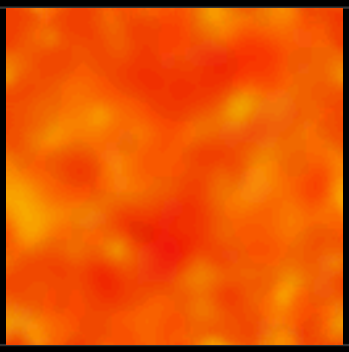
Name im Editor
Name der `ogf`
Wenn Licht, R/G/B

Lava Textures

	
Boiling Lava	Magma_Flow
SeanLavaPools.ogf	SeanLavaFalls.ogf
5.0/4.0/2.0	5.0/4.0/2.0

Procedural Textures

			
BlueMagneticField-V	CED_CoreSkin01	BluePlasma	MatCen
LightCrossing01x.ogf	DomeSkin.ogf	BluePlasma.ogf	MatCen.ogf
1.0/1.5/2.0	2.0/5.0/9.0	2.0/3.0/4.0	30.0/15.0/45.0

			
Name im Editor	Boiling Lava	Magma_Flow	Magma_Flow
Name der ogf	SeanLavaPools.ogf	SeanLavaFalls.ogf	SeanLavaFalls.ogf
Lichtwerte	5.0/4.0/2.0	5.0/4.0/2.0	5.0/4.0/2.0

Shortcut-Liste

Other
- save Room = Ctrl + s
- save Level = Ctrl + alt + s

View Control
- cycle through windows = Ctrl + Tab
- pan = Shift + left mouse
- zoom = Shift + Ctrl + left mouse or mouse wheel
- scroll left = a
- scroll right = d
- scrolling = arrow keys
- zoom in = w
- zoom out = s
- Toggle Wireframe/Textured View = t
- grid smaller = Page Down
- grid larger = Page Up
- grid size = 1,2,3,4,5,6,7 on alpha-block
- Move camera to current face = Shift + c
- center Room and reset View = RAlt + c
- center current face = Shift + c
- center Mine/Room = c

Mode select
- face: ctrl + f
- vertex: ctrl + r
- object: ctrl + g
- path: ctrl + h
- Toggle Vert/FaceMode: Tab
- object or path node : LClick

Selecting next...
- vertex = v
- face = f
- object = o
- room = r
- portal = p
- path = h
- node = n
- select previous above items add shift + (key)

Mark items in vertex, face, and object modes
- mark selected = space
- unmark all = u
- invert markings = i
- mark all = m
- mark worldview = m

Marking multiple items
- Dragging a rectangle around vertices and faces will mark all that lie entirely within the rectangle.
- alt + drag to unmark

Modify items

Moving marked items (vertices faces, objects, nodes in 2D views)
- numpad keys or Shift + arrow key :
- left = 4 or Shift + ←
- right = 6 or Shift + →
- up = 8 or Shift + ↑
- down = 2 or Shift + ↓
- twist left = 1
- twist right = 3
- objects rotate left = 7
- objects rotate right = 9
- Rotate around current vertex = Ctrl+1 / Ctrl+3
- Rotate around current face center = Ctrl+Alt+1 / Ctrl+Alt+3
- move item parallel to current edge = Ctrl+Numpad+/Numpad-
- move item parallel to current face's normal = Ctrl+Alt+Numpad+/Numpad-

Nodes
- move camera to current node = Shift+d
- select node = LClick

Vertices:
- place vertex (vertex mode) = Ins
- Rotate around current vertex = Ctrl+1 / Ctrl+3
- Rotate around current face center = Ctrl+Alt+1 / Ctrl+Alt+3
- Copy marked verts = ctrl + c
- Paste verts = ctrl + v
- Paste verts on top (of copied item) = ctrl + shift + v
- Mark all Verts in face = ctrl + m

Faces:
- flip marked face(s) = n
- copy face = ctrl + c
- paste face = ctrl + v
- paste on top (of copied item) = Ctrl + Shift + v
- place face (face mode) = Ins
- place face (vertex mode) = Shift + Ins
- combining co-planar faces = Ctrl+Shift+Click
- propagate texture alignment = Ctrl+Click
- center current face = Shift + c
- Mark all reachable Faces from current = 0
- Mark all marked faces Verts: Ctrl + m

Objects:
- Set camera view from Object = Ctrl+Shift+g

Ortho Window:
- Rotate around current vertex = Ctrl+1 / Ctrl+3
- Rotate around current face center = Ctrl+Alt+1 / Ctrl+Alt+3

Level Window:
- Set object move/rotate to X axis = Alt+X
- Set object move/rotate to Y axis = Alt+Y
- Set object move/rotate to Z axis = Alt+Z
- move object along set axis in positive dir = Alt + ↑
- move object along set axis in negative dir = Alt + ↓
- rotate object around set axis in positive dir = Alt + ←
- rotate object around set axis in negative dir = Alt + →

Rooms:

Moving placed rooms
- rotate = Alt + mouse
- slide = Alt + Ctrl + mouse
- stop moving = Alt
- Rotate: 1 / 3
- Move up/down: 8 / 2
- Move left/right: 4 / 6
- Set Move Step: Numpad+/Numpad-

Terrain:
- texture rotate counterclock- / clockwise= < / >
- replace all terrain cells matching current cell = Ctrl+Shift+Alt+LClick
- apply current texture to all selected cells = Alt+Shift+LClick
- select / deselect current cell = Alt+LClick
- select / deselect next sat (Top) = b / Shift + b
- select / deselect next sat (horizont)= x / Shift + x

Game Path

Ortho Window:
- Create new Path = Shift+Ins
- Insert GameNode = Ins
- Delete GameNode = Del
- Cycle GameNode = N / Shift + N
- Cycle Path = H / Shift + H
- Delete Path = Delete last Node of this Game Path

All Number keys are the ones from the numpad, unless otherwise stated; →, ←, ↓, ↑ = Arrow keys

_EOF