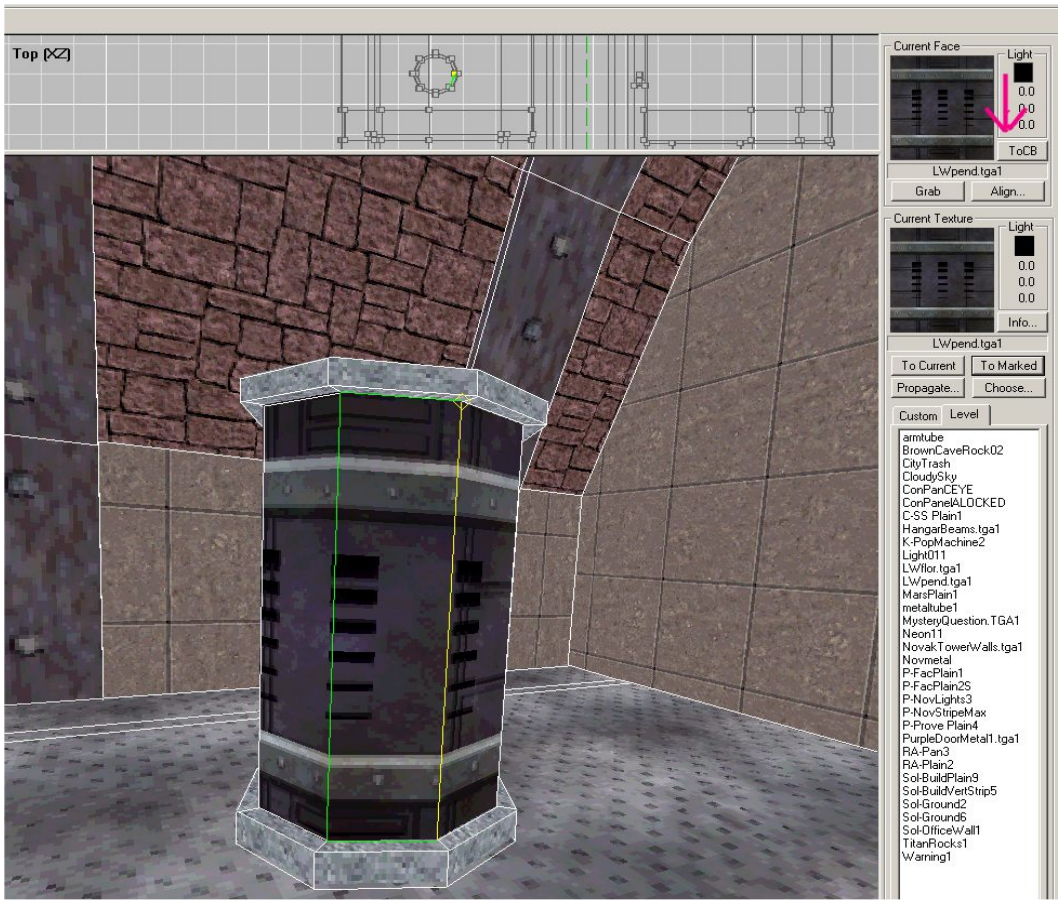
Micro-D3-Edit Tipps 3 (V1.0) AT AN

How to use ToCB in the texture bar

ToCB is designed to support DALLAS Scripting.
We are talking about replacing room textures via DALLAS script. ToCB is useful if you want to change a big amount of textures with only one line in a DALLAS script.
ToCB produces all required bits for you with only one mouse click an places them into the clipboard afterwards.
You then only need to s to paste these content into yourlevel.cpp. When you start DALLAS next time you can use this new function.

Here is a little example on how to use ToCB:
Target is to replace textures on a cylinder during the game.
Open the level room in which the cylinder is in and select one of the faces whose texture should be changed later. You shouldn't use the same texture anywhere else inside this room!



Now press the ToCB button (see the red arrow). If you haven't named that room yet, you'll now receive a message to do this. Enter the room's name inside the Room Properties dialog. We need to press this button again now. Although nothing happens visually, everything we need has been copied into the clipboard. If you don't have a yourlevel.cpp yet, you need to generate one first. While your level is still open start DALLAS and press Ok on any message, run the script, and then save it.

Yourlevel.cpp file can be found inside the D3-SDK/Osiris folder. Open yourlevel.cpp with a text editor (like Notepad) and search for the following text inside the file:

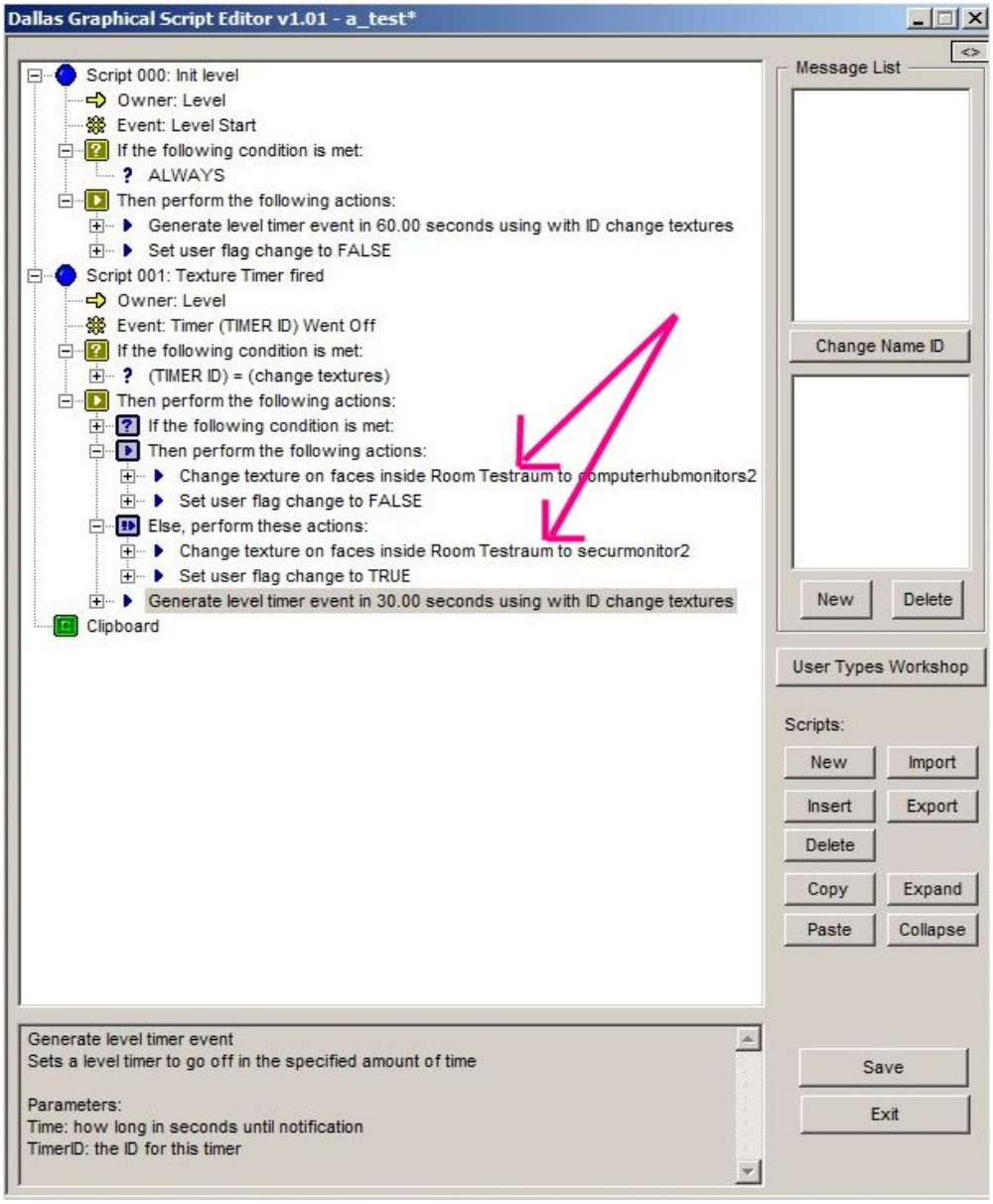
```
// ================================================================
// Start of Custom Script Block - DO NOT EDIT ANYTHING BEFORE THIS
// ================================================================
/**{CUSTOM_SCRIPT_BLOCK_START}** DO NOT EDIT! **/

// Enter your custom script code here
```

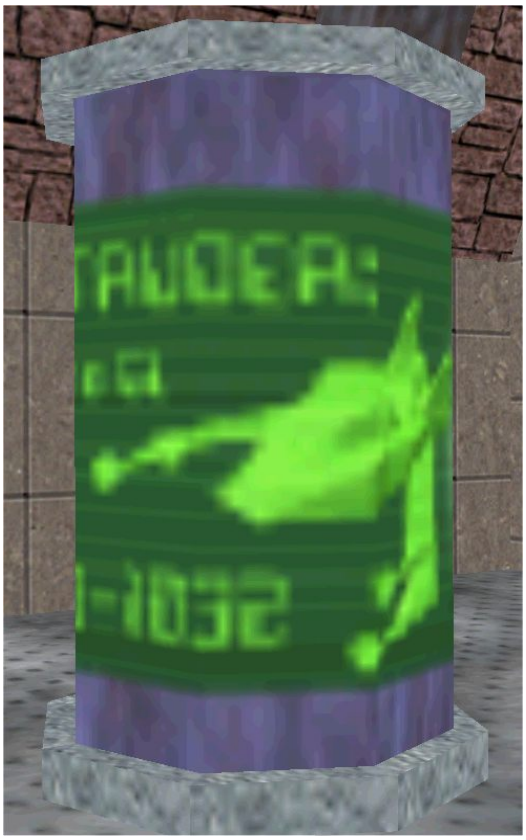
Place your cursor here and paste (Strg+V)
You'll get something like this:

```
//Place the generated code inside your level.cpp Custom Script Block
#define NUM_FACES 11

typedef struct {
char *room_name;
int		room_id;
int		face_num;
} tTextureInfo;

tTextureInfo texture_info[NUM_FACES] = {
{"Testraum",	0,	92 },
{"Testraum",	0,	97 },
{"Testraum",	0,	102 },
{"Testraum",	0,	107 },
{"Testraum",	0,	112 },
{"Testraum",	0,	117 },
{"Testraum",	0,	122 },
{"Testraum",	0,	127 },
{"Testraum",	0,	503 },
{"Testraum",	0,	507 },
{"Testraum",	0,	514 },
};
/*
$$ACTION
Custom
Change texture on faces inside Room Testraum to [u:Texture]
aSetTextureForRoomFaces
Sets the faces to a given texture

Parameters:
  TextureName: the texture to assign
  Room: Not needed
$$END
*/

bool TextureInfoInitialized = false;

void aSetTextureForRoomFaces(int texture_id)
{
int j;

		if(!TextureInfoInitialized) {
		for(j=0;j < NUM_FACES;j++)
		texture_info[j].room_id = Scrpt_FindRoomName(texture_info[j].room_name);
		TextureInfoInitialized=true;
		}

		for(j=0;j<NUM_FACES;j++)
		aRoomSetFaceTexture(texture_info[j].room_id,texture_info[j].face_num,texture_id);
}
```

Save yourlevel.cpp and open the DALLAS script editor.

Here's an example script that shows how this new function can be used. We're going to switch between two different textures with a timer.
The red arrows point at our custom function we pasted into yourlevel.cpp.
We can find it inside Action->Custom with the following name:
Change texture on faces inside Room Testraum to [u:Texture].
Inside DALLAS [u:Texture], DALLAS opens a box to select textures. We pick the 2 textures we would like to use. After running the DALLAS compiler, we place the generated yourlevel.dll and yourlevel.msg into the mission (mn3/hog) and save our mission. Now test-fly it inside the level and see if you like the result. You can change the timer values inside the example until you think it fits well.





Hint:
If you need this function inside your level more than once you need to change a few things in the generated script manually. Here are the lines you need to change:

```
#define NUM_FACES xx  => #define NUM_FACES_2 xx

} tTextureInfo;  => } tTextureInfo_2;

tTextureInfo texture_info[NUM_FACES] = {   =>  tTextureInfo_2 texture_info_2[NUM_FACES_2]

aSetTextureForRoomFaces  => aSetTextureForRoomFaces_2

bool TextureInfoInitialized = false;  =>  bool TextureInfoInitialized_2 = false;

void aSetTextureForRoomFaces(int texture_id) =>  void aSetTextureForRoomFaces_2(int texture_id)

if(!TextureInfoInitialized) { =>  if(!TextureInfoInitialized_2) {

for(j=0;j < NUM_FACES;j++) =>  for(j=0;j < NUM_FACES_2;j++)

texture_info[j].room_id = Scrpt_FindRoomName(texture_info[j].room_name);
 =>  texture_info_2[j].room_id = Scrpt_FindRoomName(texture_info_2[j].room_name);

for(j=0;j<NUM_FACES;j++)  =>  for(j=0;j<NUM_FACES_2;j++)

aRoomSetFaceTexture(texture_info[j].room_id,texture_info[j].face_num,texture_id);
 =>  aRoomSetFaceTexture(texture_info_2[j].room_id,texture_info_2[j].face_num,texture_id);
```

Increment _2 to _3 if you need it three times, etc.
This should do the trick.