Shooter With Level Changing
Tutorial

by Lee-Orr Orbach

Intro

In this tutorial you will learn how to change levels in your game. Changing levels is important because most games without levels are easy and boring. In addition, if you build a map that has everything in one large world, the computer will run slowly and the game will take too much disk space and RAM!

To try the tutorial, you don't need to know anything about the level editor, Wed (although some understanding of navigation in the 3D views does help.) You should know a little about C-Script, however. To learn the C-Script syntax, refer to the 3D Game Studio/A6 help and choose “c-script – learn c-script in 6 days – Sunday” and read it. If you know how to use the level editor and have your own project, go straight to the third section of this tutorial.

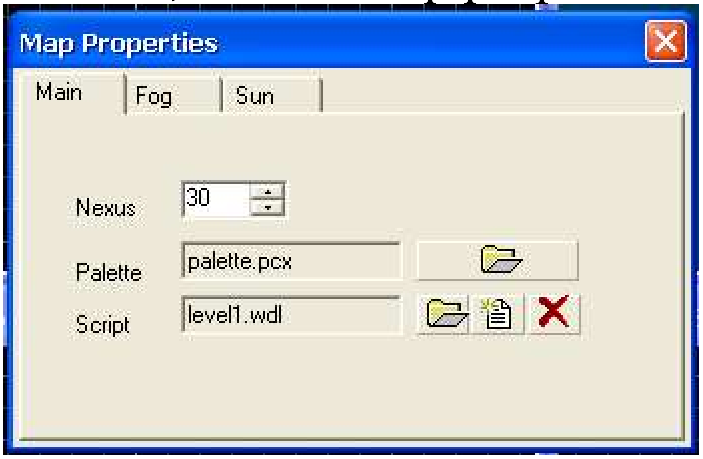
Level 1 – Street

To begin this tutorial, you shall learn how to make a new level in Wed, how to add objects and how to build and run a level.

Part 1: level setup and building

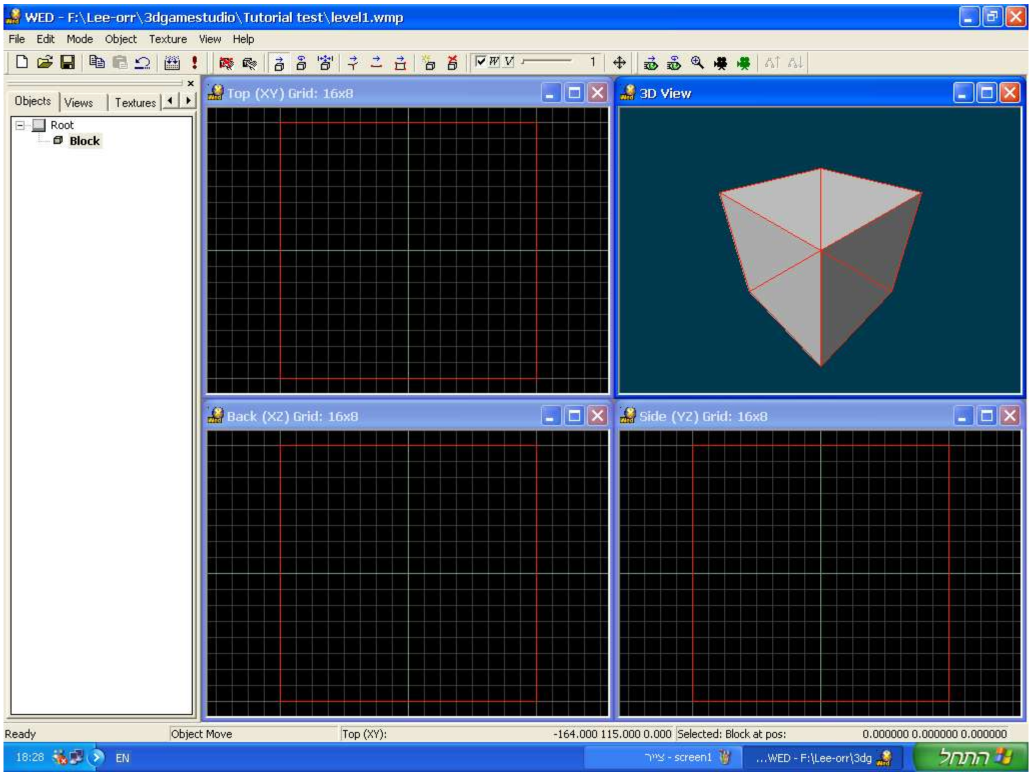
In this part, we shall make a cube to be used for our street, and add a script to the level.

1. Open Wed and make a new file. Save it right away, using a name such as "level1", with no embedded spaces.
2. In the file menu, Go-to "map properties."

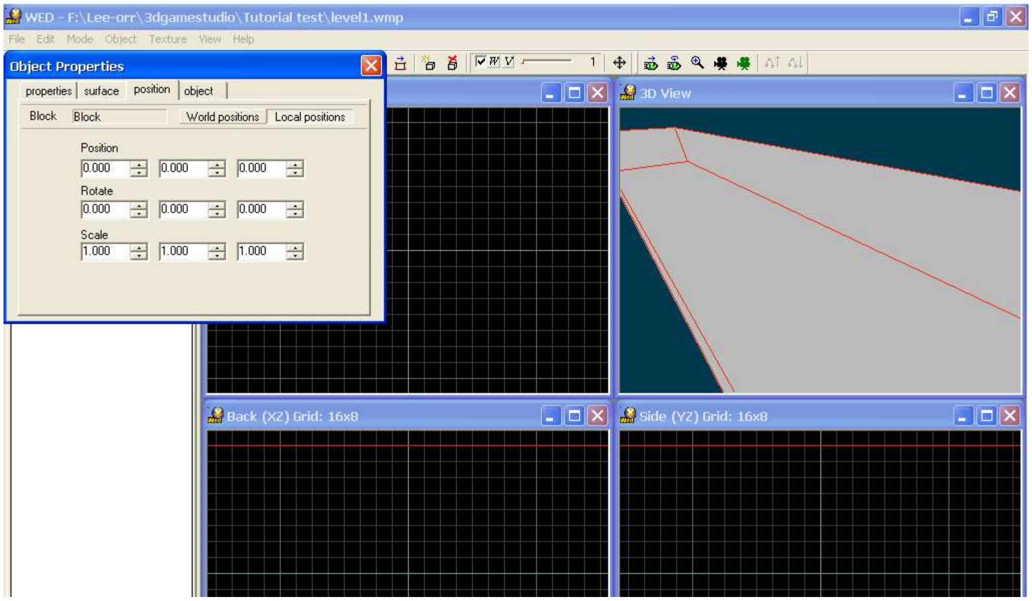


Nexus: 30
Palette: palette.pcx
Script: level1.wdl

3. Press the new script button and choose "old_template_project" and close window.
4. Make a large cube by going to "Object – Add Cube – Large" in the main menu.



5. Re-scale the cube to: 15, 2, 1 by right-clicking the cube, choosing properties, moving to the positions tab and changing the scale values. Note that as soon as you've entered a new value and hit the <enter> key – the size of the object is changed and the scale value is reset to 1.000 based on the new size.



Block: Block

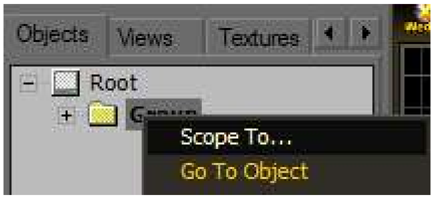
Position: 0.000 0.000 0.000
Rotate: 0.000 0.000 0.000
Scale: 1.000 1.000 1.000

6. Make the block hollow by choosing “Edit – Hollow Block” in the main menu with the cube still chosen. You get a group in the Objects tab.

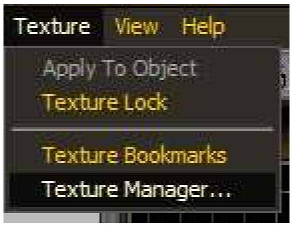
Part 2 - Texturing the Level

O.K., we finished building our first level, now, let’s give it some colors! The coloring process is called Texturing.

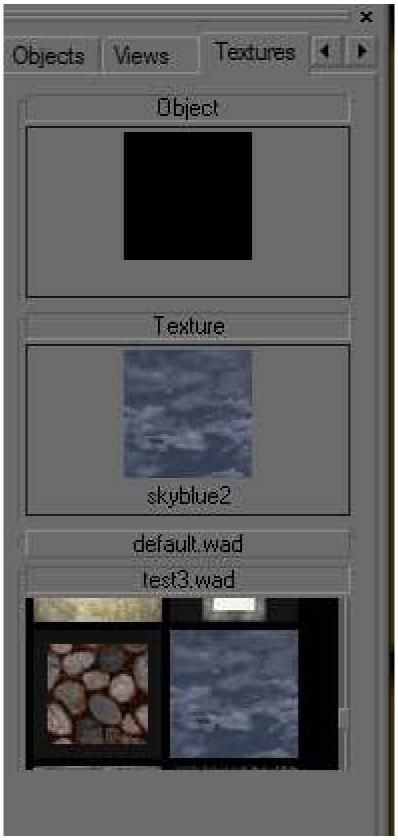
1. To Texture our street, we must scope into its group first. To scope into a group means to edit the group without affecting the rest of the level. To scope into the street group, right-click the group and choose scope to…



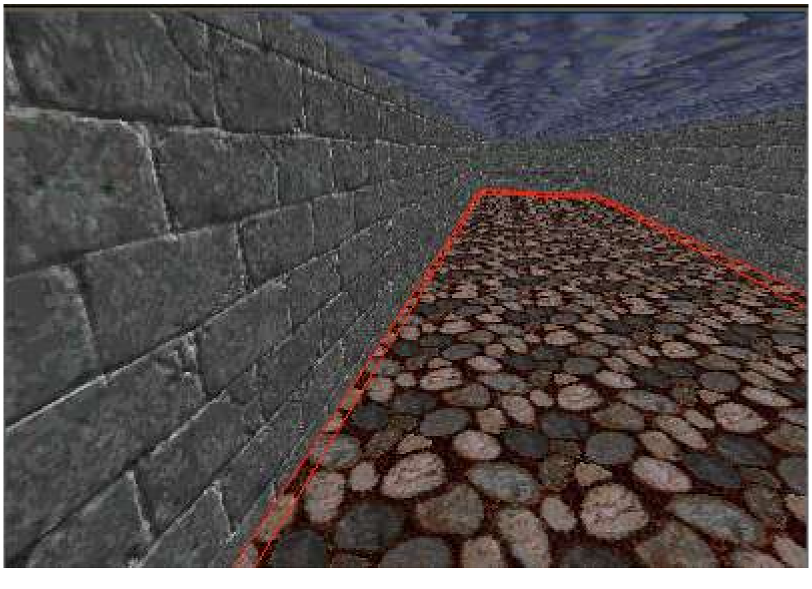
2. If the group didn't expand automatically then expand it.
3. Now, we are going to add a Wad file, a Wad file is a file with textures stored inside. To load a Wad file, choose "Texture – Texture Manager" from the main menu. Press the "Add Wad" button and choose the Test3.wad file in the work directory of 3D Game Studio / A6. (C:/program files/ 3dgamestudio/work or something similar)
4. Choose the top block in the "Objects" tab. Now, switch to the "Textures" tab and choose a sky texture from Test3.wad.



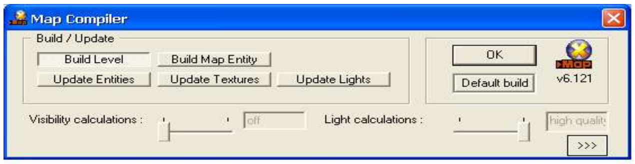
5. Double-click the sky texture.



6.Go back to the Objects tab and select the 4 blocks representing the walls. Do this by double clicking on the first block (under the sky block), and then <ctrl>click on each of the other blocks. Texture them with a wall texture.



7. Select the last box and give it a floor texture.Select the last block from the “Objects” tab and assign a floor texture to it.
8. Scope back to the root directory.
9. Check it out! All you need to do is follow steps 10 through 13, but if you want to continue right to adding entities without trying the looks, you can continue.
10. Choose “Object – Add Position” in the main menu to add a camera to look from. Then save the level.
11. Click “File – build WMB…”, select the “Build Level” button in the Map Compiler, and choose O.K.



Visibility calculations : off

Light calculations : high qualit

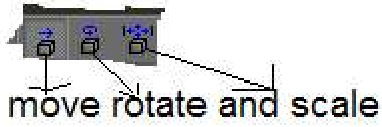
12. Wait until the word “time” will appear (with a number next to it) and press OK.
13. Click “File – Run Level” and wait for it to compile, and then you can see the level!

Part 3 – Adding Entities

In 3D Game Studio/A6 you call all the moving things entities. If you want an entity to move, like a player or enemy, you need to add behaviors and actions to the entity.

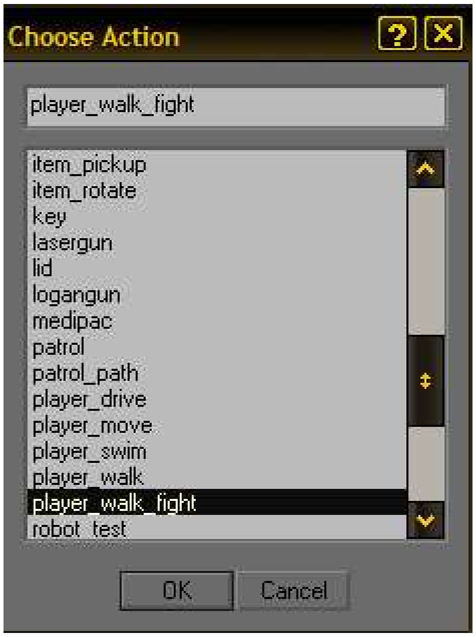
Sub-part 1 – the player

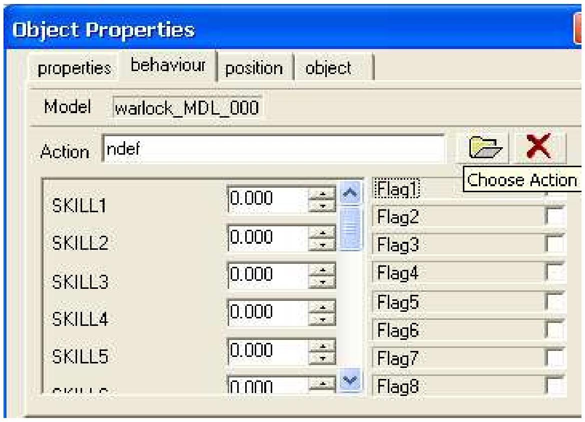
1. If you completed building the level in the previous part, delete the camera position by choosing it right-clicking and pressing the "Delete" button.
2. If you didn't, build it now, without the position (don't run the level.)
3. Go to "Object – Load Entity…" in the main menu
4. Choose "warlock.mdl" from the 3D Game Studio/A6 work directory.



5. Move the warlock to one side of the street and rotate him so he faces the other side. Take care he is no more than one quad(the spaces between the grids lines) from the floor bock.
6. Right-click the warlock entity in the objects panel and choose "Properties." Then select the behaviour tab.
7. Press the "choose-action" button.

8. From the list, choose "Player_Walk_Fight" and click OK.

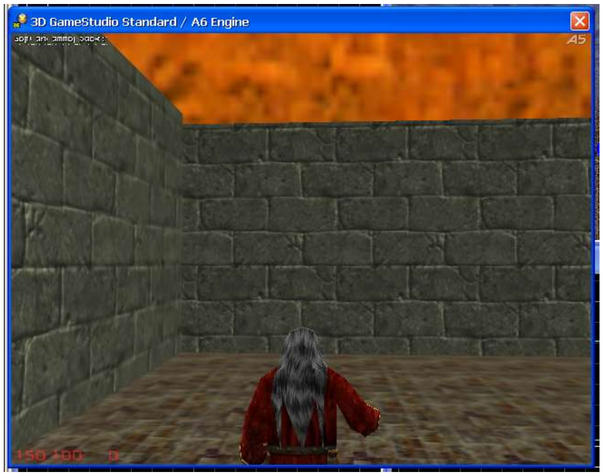




Model warlock_MDL_000
Action ndef
SKILL1 0.000
SKILL2 0.000
SKILL3 0.000
SKILL4 0.000
SKILL5 0.000

9. Close the window.
10. You're done with the player!!!!!!!!!! If you want to check it out, do the following:
11. Choose "file – build WMB…"
12. Select "update entities" and press OK.
13. Wait and run the game (arrows to walk, shift to run).

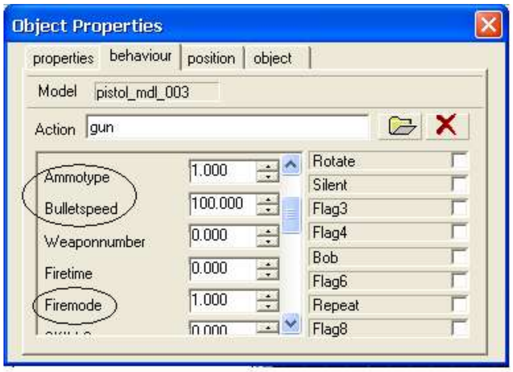
A scene while I moved around:



Sub-part 2 – going fishing

Now, we will add our weapon, the pistol. We need to add a behavior to our pistol model after loading it.

1. Press “object – load entity…” and load the “pistol.mdl” file from the work directory.
2. Move the pistol so it’s exactly where the player is.
3. Open the pistol's properties window in the behaviour tab.
4. Choose the gun action from the action list.
5. Change ammo-type to 1, bullet speed to 100 and fire mode to 1.



Model pistol_mdl_003

Action gun

Ammotype 1.000

Bulletspeed 100.000

Weaponnumber 0.000

Firetime 0.000

Firemode 1.000

Rotate ☐

Silent ☐

Flag3 ☐

Flag4 ☐

Bob ☐

Flag6 ☐

Repeat ☐

Flag8 ☐

6. Great, let’s add some bullet packs! To do that, load the “fisch1.mdl” module.
7. Open its properties window and choose “ammo_type1” action. Change pack amount to 50.
8. Copy it and distribute the copies around the level (in places the player can reach) let’s say, 7 times, and save. You've now completed your first

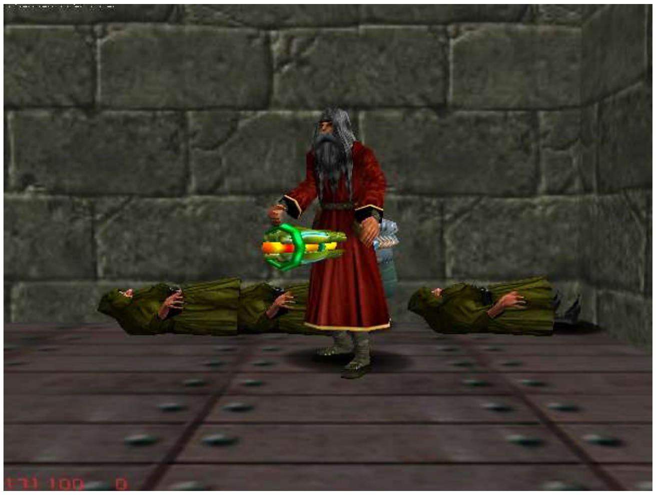
weapon! ! ! ! ! ! ! ! ! ! ! Note: make sure the bullet packs and the weapon are in places a player can reach

9. Re-build the level, with “update entities” pressed and than run the level, pick the packs and shoot around.

Sub-part 3 - enemies

I don’t think you need any intro for this, so let’s start!

1. Load “witch.mdl” from the work directory and open it’s properties window.
2. Add the “actor_walk_fight” action to the witch model.
3. Change the parameters: force = 2, health = 20 and armor = 20, we want our witches to be cowards in this level - they will be far stronger in the next one.
4. Copy the witches around the level. I made 7 witches so it won’t be too hard.
5. Save the game.



6. Build the level and play. Congratulations! You completed the first level! ! ! ! ! ! !

Level 2 – The Great Hall

In this level, you will learn how to make rooms, doors and stairs.

Part 1 – making the level

Here, you shall make a large hall and two corridors. You won't learn new techniques here, but gain some more practice.

1. Start a new level, save it, and go to the map properties window.
2. Make a new template script and close the window.
3. Build a new large cube and re-scale it to 10, 10, 1 so it looks like a hall.
4. Hollow the cube by selecting it and going to “Edit – Hollow Block”.
5. Make another large cube and scale it to 1, 15, 1 so it forms a corridor. Move to the left side of the hall and make sure the floors are at the same height.
6. Hollow the corridor as well.
7. Copy the corridor and move the copy so it's immediately under the first corridor.

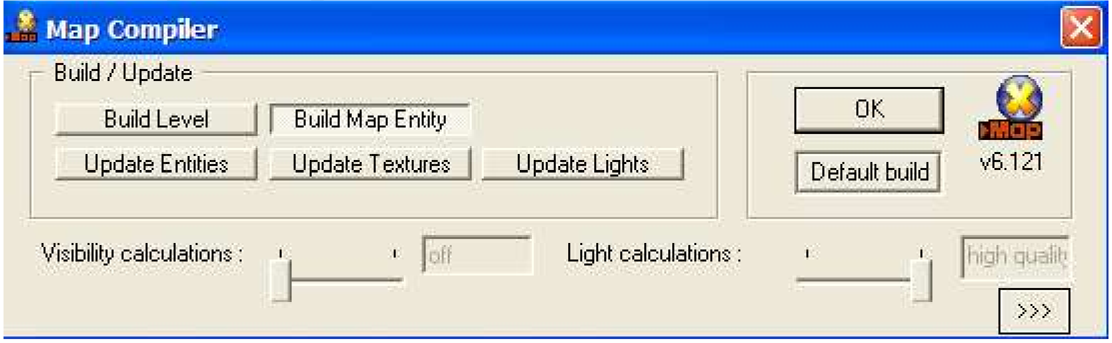
Part 2 – texturing the level

Again, just practice, and practice makes PERFECT!

1. Go to “Texture – Texture Manager” and add “standard.wad” from the 3D Game Studio root directory.
2. Choose the hall and scope to it.
3. Texture it as you want and do the same with the corridors.

<u>Part 3 – doors and stairs</u>

1. Save the level, and make a new level.
2. Create a small cube and resize it to: 0.2, 0.7, 1.2, so it looks like a door.
3. Add the standard Wad to the level and texture the cube so it looks like a door.
4. Save the level as “door.wmp”
5. Build the level by selecting “Build Map Entity” in the Map Compiler.



Visibility calculations : off

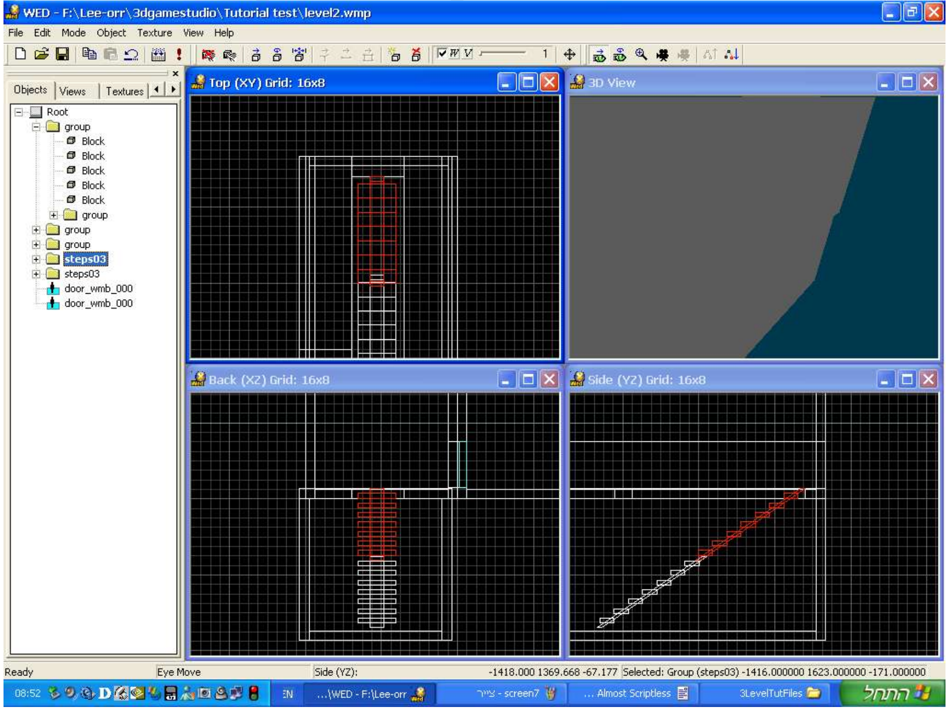
Light calculations : high qualit

6. Re-load the second level.
7. Make a small cube in the level.
8. Resize it to:0.7,1, 1.2, so it has the same size as the door on the x and z axes.
9. Move the cube to the side of the hall in which the hall touches the corridor. Ensure it's bottom touches the floor. This will become the opening between

the hall and the corridors. Make sure the block passes to the corridor side of the wall but still stays in the hall side as well.

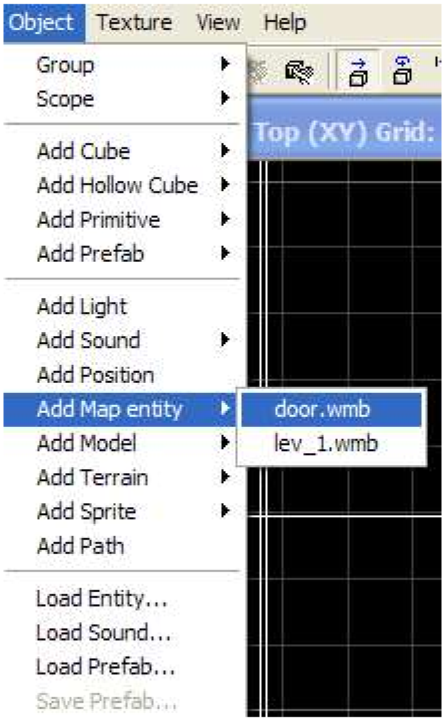
10. Goto Edit – CGS Subtract
11. Move the block, and wow! There is a hole over there!
12. Move the block to one side of the corridor, but not to the end of it.
13. Move it down a little so it passes the ceiling of the lower corridor.
14. Scale it to 2, 5, 1 and take care it doesn't get to the edge of the corridor!

16. CGS subtract the block and delete it.
17. Add a stair case by going to "Object – Add Prefab – upstairs – steps03.wmp"



Ready | Eye Move | Side (YZ): -1418.000 1369.668 -67.177 | Selected: Group (steps03) -1416.000000 1623.000000 -171.000000

18. Move the stairs so the top step is level with the top corridor's floor.
19. Add another staircase and move it so the highest step of the lower stair case is in exactly the same place as the lowest step in the higher stair case. Align the stairs with each other and with the middle of the opening between the corridors.
20. Add the doors by going to “Object – Add Map Entity – Door.wmb” (it will work only if you saved ALL the files used in this tutorial, levels and scripts, in the same directory. If not, move all the files, WITHOUT directories to the same directory).



21. Move the door to the hole in the wall of the hall wall and fit it in there (put half of the door in the wall.) and take care its almost level with the floor.

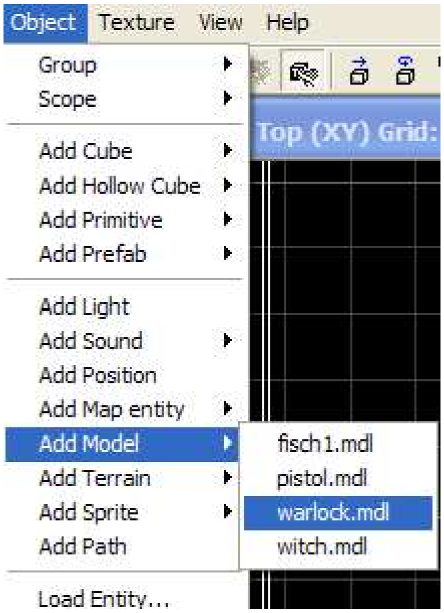
22. Add another door and do the same thing, but put it on the other side of the hole.
23. Choose each door one after the other and give them the “Door” action from the behaviour tab in the “object properties” window.

Part 4 – adding entities

practice, practice, practice ...

Sub-part 1 – adding the player

1. Add a warlock model. This time, it will be found in “Object – Add Module – warlock.mdl” because you already imported it to the first level (it will work only if it's in the same directory.)



2. move the warlock to the bottom corridor and put him next to the wall opposing the ladder.

3. Open the warlock's properties and choose the behaviour tab.
4. Add the “Player_Walk_Fight” action and close the window.

Sub-part 2 – the gun

1. Go to “Object – Load Entity” , find “mgun.mdl” and load it.
2. Move the mgun module so it's right in front of the warlock.
3. Give it the “weap_mg_animated” action, and change the weapon number to 2.
4. Load the fisch1.mdl module like you did when you loaded the player module.
5. Give it the ammo_type1 action.
6. Change pack amount to 100.
7. copy 7 times.

Sub_part 3 – the witches

1. Load the witch.mdl module.
2. Give her the actor_walk_fight action.
3. Change health to 50, armor to 50, force to 5 and close the window.
4. Copy the witch several times and distribute them so there will be some in the top corridor and some in the bottom corridor.

5. Save.
6. Load the witch module again.
7. Re-scale it so it's twice its current size (2,2,2), and then move it into the hall.
8. Change health to 70, armor to 100, and force to 5.
9. Save.
10. Build and run ("build level" - not "update entities")
11. You have succeeded in creating the second level! GREAT!!!!!!!!!!!

Scripting

Now, at last, you will add the level changing script. If you want to understand the syntax, please read the Sunday part of the "Learning C-Script in 6 Days" tutorial in the 3D Game Studio manual.

1. Open Sed (Script editor).
2. Load the "lev_1.wdl" script.
3. Scroll to the bottom and click right above the line full of these ////////////////////////////////.
4. Type these lines in (or copy them from here):

```
var playerpos[3];
var weapon_1;
function load_lev_2()
{
weapon_1 = weapon1;
msg_show("You Have Succeeded The Level!!! Wait Until Level 2 Finishes Loading", 5);
wait(2);
my = null;
level_load("secondlevelfilename.wmb");
wait(1);
if (weapon_1 != 0)
{
playerpos[0] = -1382.00;
playerpos[1] = -1523.000;
playerpos[2] = -221.000;
ent_create("pistol.mdl",playerpos, gun);
}
gun_select();
}
action level2
{
		MY.EVENT = load_lev_2;
		_doorevent_init();
		if(MY._FORCE == 0) { MY._FORCE = 5; }
		if(MY._ENDPOS == 0) { MY._ENDPOS = 90; }
}
```

Explanation:

`var playerpos[3];`

This line defines a vector (3 variables with the same name), we shall use it to check the player's position.

`var weapon_1;`

This variable will be used to save the status of the weapon when moving levels. It will contain 1 if the weapon is picked, and 0 if it isn't. This is needed because all variables assigned to objects are initialized automatically when changing levels.

`weapon_1 = weapon1;`

Check if the pistol (weapon1) is in the players hands. The variable weapon1 is pre-defined in the template script.

```
msg_show("You Have Succeeded The Level!!! Wait Until Level 2 Finishes
Loading", 5);
wait(2);
```

Shows a message for 5 seconds and waits 2 frame cycles.

```
my = null;
```

Makes the function continue, indicating it relates to an object outside of the scope of the level – so it won't be erased when changing levels.

```
level_load(“lev_2.wmb”);
wait(1);
```

Loads the level and waits one frame cycle. Lev_2.wmb is the filename we saved earlier.

```
if(weapon_1 != 0)
{
playerpos[0] = -1382.00;
playerpos[1] = -1523.000;
playerpos[2] = -221.000;
ent_create("pistol.mdl",playerpos, gun);
}
gun_select();
```

Checks if the weapon is in the player's hands, and if so, re-creates the pistol in the exact position of the player within the new level. Creating the weapon object in the exact position (x, y and z coordinates) of the player is necessary for the game to associate the weapon with the player. In this case, I checked the player's second level starting position in Wed, and the code above includes the absolute coordinates. You can find out the position of a player in a level by right-clicking on the player, selecting “properties” , going to the “position” tab and clicking on “world coordinates.” It is also possible for a script to find the coordinates of the player in run time, but this requires more complicated coding which is not explained in this tutorial.

The ent_create function creates an entity as follows: ent_create(module name, position in which to create it, action to assign to it);

```
action level2
{
MY.EVENT = load_lev_2;
_doorevent_init();
if(MY._FORCE == 0) { MY._FORCE = 5; }
if(MY._ENDPOS == 0) { MY._ENDPOS = 90; }
}
```

The MY.EVENT attribute reflects the function to be performed. In this case it is the loading of the new level. The _doorevent_int() takes care the level will load only if you are close enough to it and press <space>, like you open a door.

Playing

1.Save the script and exit.
2.Load the second level in Wed.
3.Go to the “Map Properties” window and choose the “Load Script” button.
4.Select “Lev_1.wdl”, close the map properties and save.
5.Load the first level.
6.Add a door entity, and place it at the far side of the corridor, away from the player.
7.
8.Goto the behaviour tab.
9.Load the level2 action.
10.Close the properties window.
11.Move the door to the side of the street which the player is facing.
Re-build the game and play.

YOU HAVE FINISHED THE TUTORIAL!!!!!!!!!!!!
CONGRATULATIONS!!!!!!!!!