TUTORIAL 2.1 Level Creation

TUTORIAL 2.1 – Basic level creation

To create levels for the HPL engine you use a 3D editor. Make sure you have read tutorial 1 so that you have your editor properly setup.

Lets get started, open your 3D editor and make sure units are set to meter.

Create a cube in the centre of the project and make it 6m*3m*6m(for me that is width, height and depth). This will be the room making up our first test level.

First thing you need to do is to reverse/invert the normal, resulting in them facing the centre of the cube rather than facing away from the centre. Make sure you enable Backface culling so that you can see into the polygon(when it's textured that is).

Now extract/separate the faces. If done right all the sides of the cube should be separate objects that you can select. This you do because HPL requires a level to not be a 100% solid object and also because we want to have different textures on the walls and ceiling. You can only have one texture on a single object.

Lets texture the cube, from now on called the room. Use the “tut_floor_ceiling_texture.jpg” for your ceiling and floor, use “tut_wall_texture.jpg” for your four walls. You can find them in tutorials/tutorial_2.

Last, but not least, the room needs a light. Create a single point light and place it in the centre of the room, perhaps closer to the ceiling than the floor. Some specifics on lights, there are 4 things you can do with basic lights in your 3D editor. I will not go into details on creating more advanced lights, this is only the very fundamental.

The colour, you set the colour for the light as you normally do. For our light give it a small touch of yellow but keep it bright white.

The next three things that you can do are not your normal way of using lights when 3D editing. To tell the game what size the light is, how much specular it has and if it casts a shadow you use the X, Y and Z settings for the lights SCALE.

Scale X: The size in meters, 0 → as high as you like(or almost at least) Scale Y: The amount of specular, 0 → 1 where 1 is full specular. Scale Z: Cast shadow, 0 or 1. 0 means no shadow, 1 means cast shadow. There is no in between.

For our light, set X to 5 and leave Y and Z to 1.

Save the project to anywhere you want. And finally export the level as a collada file to your mystuff directory. Name the level “mystuff_level.dae”. Let's see if the level works, launch HPLHelper and select the “Scenes” tab, click browse and locate your “mystuff_level.dae” file. Click view and the level should show up.

If it does not show up backtrack and see what could be wrong. The best way to quickly find what can be at fault is to open “hpl.log” located in the My Documents/Penumbra Overture/EpisodeX/ directory.

Scrolling down you should see an error saying what the problem was, often it's that it could not create or locate a file. This usually means that you have named something wrong or simply have forgotten to add your directories to the resource.cfg file using the “Settings” tab in HPLHelper.

This concludes how to create a simple room. Now we are going to look at how you open the level in Penumbra.

TUTORIAL 2.2 – Adding player start and references

To tell the game where the player is going to be when he enters the level we will need to create a small cube. Create a new cube, perhaps 0.5m*0.5m*0.5m the size is not important, it should be usable and not take up much space. The game uses the centre of the cube to know where the player is going to start.

Move the cube down to the floor, making the centre of the cube appear slightly above the ground. Move the cube along the floor so that it's closer to one of the walls, simply because we do not want the player to start in the centre of the room.

Now we need to tell the game that this is a special cube, rename the cube node to “_start_location1”. “_start” is the string telling the game that this is a start location and “_location1” is the name of the location.

Save the project and export the level to collada again and overwrite your previous “mystuff_level.dae”.

To see if the level is working properly with the new start location, go back to HPL Helper and go to the Scene tab. Open your level BUT before you click view, enter “location1” into the startposition: now you can click view and you should start at the location of your “_start “cube. If you want to face in another direction at the start of the level, rotate the _start cube in your 3D editor.

Let's move on an get a model into the level as well. When we have done this we will test the level by actually playing it.

In your 3D editor, select import collada and locate the “ref_mystuff_woodbox.dae “ reference file that you created in tutorial 1.4. This should import the file into your levels centre, make sure the model is called _ref_mystuff_woodbox_woodbox1 as we said it should be named in tutorial 1.4. Move the reference to a location of your choice, preferably not at the same place as the player location! The model will appear in the game exactly where you place the reference, so make sure the whole object is inside the room.

Save the project and export the level overwriting your previous “mystuff_level.dae” file. Using the “Scenes” tab in HPL Helper, you can open the level and you should see a big wooden box in it. If so, congrats on getting this far!

To test the level we will need to edit the “settings.cfg” file located in the My Documents/Penumbra Overture/EpisodeX/ directoy. Open the file in a regular text editor. Locate the following

<Map File=“level01_02_entrance.dae” StartPos=“link01”

And Change File to “mystuff_level.dae” and StartPos to “location1”. Save the config file and start Penumbra. Choose a new game and enjoy playing around in your new and cool level.

TUTORIAL 2.3 – Creating static geometry in a level.

This is very short and not really a tutorial. This only gives you some information on creating static objects in your level.

If you want to add static objects, for example a couple of water pipes, you model them into the level as you would normally model in a 3D editor. Give them a texture and export the level again, when you look at the level in the game now the pipes will be there and everything will look fine. If it crashes it's most likely because you have put a texture on it that is not in a directory added to the resoruce.cfg file.

All basic geometry can be created like this. But for more complex geometry and if you want to optimize you can do a couple of things. Geometry in the game automatically gets collisions that are exact copies of their original form. This is not very efficient, most of the time you want to replace the colliders with more simple colliders of your own. To do this you do the following:

Rename your original mesh it so that it ends with “_nocollide” or begins with “nocollide_”, this tells the game that it wont create any colliders for the mesh. Now create your own simple colliders by for example creating a cube and name it “_collider_box_1” group it with your original geometry and make sure the collider has the same texture as the geometry. Now the game will only use the colliders you have created. At times you might have smaller details, say a wooden stick lying on the floor, often in games you do not want it to have any collision at all. Append the “_nocollide”(or begin with“nocollide_” to it and make no new colliders to accomplish this.

You can also appen “_noshadow”(or begin with “noshadow_”) to a node, making the name “pipes_nocollide_noshadow” this tells the game that this object should not cast a shadow. This is very useful for small details or objects located in such a way that there really is no point in the casting a shadow. For full details check the content creation document.

TUTORIAL 2.4 – Using portals

Portals is a way to divide a level into areas that limits how much the engine is rendering. Portals is a must to optimize levels, in particular complex levels with lots of details. Lets use our level and add another room to it, splitting the level in to parts by creating a portal in between them. This tutorial gives some help, for more help and more details check the content creation document. For this tutorial that document is a must, portals are a pickle to fiddle with.

Open your project for mystuff_level. If you have not done anything specific, the 6 sides of the cube that you extracted should be in a group. If they are not, group them. Name the group “_roomFIRSTROOM”. “_room” tells the game that it's a room and FIRSTROOM is the name of the room, this could just as well be simple number or a combination.

Duplicate your group so that you get an exact copy of the room and move the group to the side of the first group. Rename this group “_roomSECONDROOM”. Make sure the rooms are side by side, touching each other along a wall. Delete the two walls that are against each other. This should create

a room that is twice the size of your original room.

Create a new point light in the new room, make it the same size, but just to ease telling the two rooms apart colour the light red. Now export the level overwriting you previous “mystuff_level.dae”.

Using HPLHelper open the level in the Scene tab and make sure StartPos: says location1 before clicking view. Now that you are looking at your level you will notice that the walls that you deleted are pure black. With the “W A S D” keyboard keys and your mouse you can fly around, fly torwards the black wall and through it. You should now enter the new room you created with the red light in it.

What this means is that you have told the engine that you have two rooms by creating two groups of polygons and calling the groups “_roomSOMETHING”. The game now only renders the room that you are in and not the rest of the level, of course you do not want it to be like this here. You want to be able to stand in FIRSTROOM and look into SECONDROOM. This is where the portals come in!

To create a portal, make a plane in your 3D editor. Rename the node “_portal1_roomSECONDROOM”, and put this plane into the group that you call _roomFIRSTROOM.

This is the first portal that belongs to the FIRSTROOM group. It leads to the room(group) _roomSECONDROOM. Now move the portal so that it covers the whole area between the two rooms, make sure the normal is pointing towards the centre of the FIRSTROOM.

Now make a duplicate of the portal plane. Move the new portal plane into the _roomSECONDROOM group. Rename it “_portal1_roomFIRSTROOM” and reverse/invert the normal so that it points to the centre of the SECONDROOM GROUP.

What you now have is two rooms, each with a portal plane pointing towards the centre of their own room group. The portal names says what room they lead two, meaning that these two portals leads to the each others groups. Save and export the level overwriting “mystuff_level.dae” and take the level for a spin using HPLHelper.

If all is right, you should now be able to see from the start location into the new room with the red light. If you move to the new room you should be able to turn around and look into the first room.

If it's not working, try and redo it and read this document carefully as well as the content creation documents part on portals. To learn how to use more than one portal in a room and how to link several rooms together you will have to read the content creation document.

This example did not create a portal system that is of much use from an optimizing view. Since you have two rooms connected like this, not matter where you are in the rooms you will always see the other room, the idea with portals is to split up a level into several rooms where many of the portals do not see each other. By doing this you limit how far the game renders, if all the portals see each other it would be the same as having the whole level in one group.

A tip to see how a level is divided is to open a level from Penumbra in the HPLHelper. For example if you open level01_04_storage.dae. When you view the level, press the 2 key on the keyboard. This will show the portals and the bounding boxes for the rooms. If you also press the 7 key, you will see how the much of the level the game is rendering. Move around a little and you can see how it dynamically changes what it renders of the level. If you analyse the level you might notice that there are some walls sticking out, these walls have a primary function to block portals from seeing each other, making the level more efficient to render. It's this type of level building you need to have in mind.

TUTORIAL 2.5 – Linking two levels together.

The last tutorial for level creation is how to link to levels together using the _start and _area_link boxes. To make this work you will need a script file for your level, there is an empty script file in tutorials/tutorial2 called “tut_level.hps”. Copy this file to your mystuff folder. Rename the file “mystuff_level.hps” and open it in a text editor. As you can see it's quite empty, it contains the different foundation parts that you use for different things when writing scripts for a level. For the scripts we are going to do now you will use this part:

```
void OnStart()
{

}
```

As you can see this area already has some script in it:

```
    //SET UP LINKS
    SetupLink(    "warp", //Link name
            "mystuff_level.dae", //New map
            "location1", //Positon on new map
            "","", //Stop and end sound.
            0.5f , 0.5f);//Fade out and in time (seconds).
```

Link name, this is the area that the player can interact with to initate the loading of a new level. New map, the name of the new level and location1 is the area where the player will start when entering that new level. If you so far have named you level ”mystuff_level.dae” and added a location1 _start to it the script is ready to be used with only a slight modification.

Links can also be used to travel between different locations in a single level. To try this out we are going to use the ”mystuff_level.dae” we already have and add a new area to it.

Open the project in your 3D editor, create a new cube that is about 1m*1m*1m in size and place it in an location on the opposite side of the room from your ”_start_location1” cube. Rename this new cube to ”_area_link_warp”. This will create an area in the game that will bring up the door icon when you are close, if you click with the left mouse button you will activate the area and be transportaed to location1 in the map ”mystuff_level.dae”. Since this is the map you are already in you will simply be warped back to your starting location.

To make it easier finding the ”_area_link_warp” cube, lets add a point light where the area is. Make it bright green and set the X scale to 2, making the radius of the light 2 meters.

Export your level and overwrite the old ”mystuff_level.dae”. Make sure that you have copied the script file to the same location and that you have neamed it ”mystuff_level.hps” as we said in the beggining of this chapter. Now launch Penumbra and start a new game, you should see a green light and if you walk over to it and interact you should be transported back to the start.

Lets do the same but use two different levels. First go back to your current project in the 3D editor and create another cube that you place in another location and rename it ”_area_link_travel”. Do the same here that you add a light to make it easier finding the location, this time make it bright blue. Export this level and overwrite the previous ”mystuff_level.dae”.

Now, in your 3D editor delete the refernece for the wood box and also change the main lights in the room(the ones that are 5 meters) to any other bright colour you like. We do this simply to quickly make a level that looks different from the one we already have.

Continue on and rename you ”_start_location1” to ”_start_newmap” and change your ”_area_link_warp” to ”_area_link_travelback” we do this not because we have to but because you need to practice renaming and later editing the script. Last delte the ”_area_link_travel” cube and the light for it as well. Export this level to collada and name it ”mystuff_level2.dae”.

You should now have to maps, mystuff_level1.dae and mystuff_level2.dae.

Go back to your ”mystuff_level.hps” script file and select the SET UP LINKS part and make a copy of it, your script file should then look like this:

```
void OnStart()
{

    //SET UP LINKS
    SetupLink(    "warp", //Link name
            "mystuff_level.dae", //New map
            "location1", //Positon on new map
            "","", //Stop and end sound.
            0.5f , 0.5f);//Fade out and in time (seconds).


    //SET UP LINKS
    SetupLink(    "travel", //Link name
            "mystuff_level2.dae", //New map
            "newmap", //Positon on new map
            "","", //Stop and end sound.
            0.5f , 0.5f);//Fade out and in time (seconds).
}
```

Edit the second SET UP LINKS so that the link name is ”travel”, the new map is ”mystuff_level2.dae” and position changed to ”newmap”. Save the script file.

You now have the script file ready for ”mystuff_level.dae” but you will also need one for your new ”mystuff_level2.dae”. Make a copy of the script file and rename it ” mystuff_level2.hps”. Open this new script file in a text editor and delete one of the SET UP LINKS so that you get a script file that looks like this:

```
void OnStart()
{

    //SET UP LINKS
    SetupLink(    "travelback", //Link name
            "mystuff_level.dae", //New map
            "location1", //Positon on new map
            "","", //Stop and end sound.
            0.5f , 0.5f);//Fade out and in time (seconds).
```

```
}
```

Rename the link, new map and position to be as they are above. ”travelback” is the name of the link in you new ”mystuff_level2.dae” level that will activate to take you back to your first map and make you start in ”location1”.

Save this script file. You should now have two levels and two script files. To test if it works, launch Penumbra and start a new game. If you click the green area you should warp inside your first level, if you click the blue area you go to the new level with out a wooden box in it and if you click the green area in it you should go back to the first level again.

If it worked, a big congratulations. If not, try again! These two levels and their script files can be found in the ”tutorials/tutorial2/safety”. This directory is not added to the resource.cfg so you can't run the levels from there, I added them just incase you can't get it to work so that you would have something to look at and investigate to get it to work.

Good going reading through this tutorial, it's the dullest and probably most difficult of them all. But now that it's over you should be able to create levels that have the right basic functions. In the three last tutorials we will look at material/texture files, particle effects and in the final one do some basic scripting using everything we have learned so far.