TUTORIAL 1.1 – Introduction

This first tutorial is a MUST TO READ. It deals with specific file and directory requirements for using the engine (HPL). Read it to save headache before starting to create models, maps and so forth.

When dealing with files and folders for the HPL directory there are a couple of important factors to know about. HPL does not care about directories! This means that if you place two files with the same name and extension at different locations, HPL will not be able to tell the difference and simply use the first one it finds.

Because of this it is very important to always give files unique names!

It seems odd? It's a choice made to allow moving files around and easing the locations of files when writing scripts and specific game files. This means that if you want to play a sound in the game you simply refer to the file name instead of having to refer to the exact path to the file.

HPL does not search all the directories on it's own, you have to tell HPL what directories there are for it to search. This is done with HPL Helper that edits the file ”resources.cfg”. To create a tutorial for this, lets use HPL helper to edit the “resources.cfg” file.

First go to the ”redist” directory and create a new folder called ”mystuff”. Launch HPL Helper and you should automatically be at the settings tab, this is the part of HPL Helper we want to use. Click ”Add” and locate your newly created directory and add it to the list. If done correctly you should now have a /mystuff at the bottom of the list.

Lets continue and add some more directories, first move the tutorial directory into the redist folder as well. We want to do this because it contains files we want to use for our tutorials. Using HPL Helper add the following directories:

tutorials *tutorials/tutorial_1* tutorials/tutorial_2 *tutorials/tutorial_3* tutorials/tutorial_4 *tutorials/tutorial_5*

Your resource directory list in HPL helper should now list these additional directories. When all is done press the “Save to File” button to save the settings. To see if you have done it right lets try and open a test model. Go to the ”Models” tab in HPL Helper and select browse. Navigate to tutorial_1 and select the ”tut_woodbox.dae” and test the view button. If all works correctly you should get a nice wooden box in the viewer and if you left click it you should be able to bounce it around.

That concludes the directory tutorial

TUTORIAL 1.2 – CREATING YOUR FIRST MODEL

Using HPL to create games requires about as much knowledge and time as using any other advanced game engine to create new content/mods. What differs HPL from the rest is that it does not use a specific map editor; rather it uses regular 3D editors. The only requirement is that the 3D editor supports the COLLADA format, http://collada.org. Following will be a kick-start on how to create a very basic model and to get it working in the game. Following that will be creating a level consisting of one room and making that room contain the model created. This tutorial will use pre-made textures; details for textures will come later.

To get started, search and locate a module/extension/library for your editor that gives it support for COLLADA. Install it properly and start your editor, make sure you are able to export and import collada files. It's important that you check the ”export polygons as triangles” option for exporting in collada.

First thing to do in your 3D editor is to make sure the units are set to equal 1 meter. The HPL-Engine uses the SI standard for all units.

Create a basic cube, with the dimensions 1m*1m*1m and leave it in the center of the project. This cube will need a texture, to keep it simple we have a texture ready for you to use

It's located in ”tutorials/tutorial_1/tut_woodbox.jpg”

Attach the texture to the cube, do not rename or move the texture. You should now have a pretty cube with a wooden surface. Save the project to a location of your choice, this work file can be stored anywhere you want.

Let's test the model and see if it works in the game. Export it as a collada file and save the file in your ”mystuff” directory, name it ”mystuff_woodbox.dae”. Now using HPL Helper open the file using the ”Models” tab.

If all is working OK you should have a wooden box in your view now. If you try and click it and move it around it does not work! If you recall from the previous tutorial it worked on that wooden box, it does not work with this one because the model we have created does not have a collider. This is something that we will need to add.

Going back to you 3D editor, make a copy of your wood box cube. Do not move it, leave it exactly in the same position as the original cube. Now rename the new node to ”_collider_box_1”. What this does is that you create a string for the game with ”_collider” and then you specify what type of collider it is with the ”box” and finally you name it ”1”. The starting “_” act as hint for the engine so that it knows that this geometry is special, the other “_” are separators between the different properties.

Different types of colliders are available, box, sphere, cylinder and so forth. For the exact list view the content creation document. More complex model can have multiple different colliders, you can make as many as you want to get the collision detection as detailed as possible, but the idea is to make it as simple as possible.

In your project you should now have 1 node that is the original box and 1 node that is the new _collider. Export the model again and open it using HPL Helper. If done correctly you should now have a box that can be moved around.

Not going into detail here, but it's good to always create a group for your mesh and the colliders. Later you might create models that have different parts. Say you create a door, where you want the frame to be a static object but the door an object that the player can open and close. For this to work you need to make the frame and doors separate objects in your 3D editor and group them with their respective colliders.

TUTORIAL 1.3 – ENTITY FILES

When you have a model finished you might want to add specific properties to it. Such as what type of material is the object, how much it weighs, how you interact with it and does it collide with the player? To do this you'll need to create an entity file, this is basically a regular text file containing all the information, an XML file to be specific.

To start off lets take a look at an existing entity file. Using a regular text editor (preferably wordpad or better, not notepad and never word or any other office like suite.) open the ”tut_woodbox.ent” file located in tutorials/tutorial_1.

This is a quite simple entity file, let’s take a look at what it contains.

```
<MAIN
    Name="tut_woodbox"
    Type="Object"
    Subtype="Normal"
/>
```

These are quite unimportant. For simplicity always put the name of the file in Name=””. Type and Subtype are not often changed. Type is ”Object” for most models, if you make a model that is going to be picked up, say the dynamite in Penumbra, you change it to Type=”Item”. Subtype is so far not used at all except as Normal. For details on different Types view the GameEntityVars document.

```
<PHYSICS
    SubName = "pCubeShape1"
    Collides = "true"
    HasPhysics = "true"
    Material="Dyn_Wood_Box"
    Mass="30"
    InertiaScale = "1 1 1"
    AngularDamping = "0.1"
    LinearDamping = "0.1"
    BlocksSound = "false"
/>
```

Most of these are self-explanatory. The SubName is important, is should always be the name of the mesh. It's of extra importance when you have a model with different parts, otherwise the engine wont be able to tell, for example, what part of a door that is static and not.

Material=”” says what material the object it, this is used to give it friction, sounds and such material specific things. A list of materials can be found in HPL Helper under the ”Materials” tab, by the drop down menu ”physics material”.

Angular and Linear Damping, lets play with them a little. Using HPL Helper, under the ”Models” tab open the ”tut_woodbox.ent” file. This will open the tut_woodbox.dae file BUT also give it the specific properties from the ent file. Now spin the box around a little, throw it around and notice how it behaves. Go back to the text editor and edit the entity file to have AngularDamping = “1” and LinearDamping = “1” and save the file. Go back to HPL Helper and open the the file again, now if you bounce and spin the box around you should notice that it wants to spin and move less/slower. This is

because we raised the damping factor. As always more exact details on everything can be found in the content creation document.

```
<GRAPHICS
    ModelFile = "tut_woodbox"
    CastShadows = "true"
/>
```

ModelFile =”” this is the name of the model (.dae) file the entity file is setting properties for. This means that you can create one model and then you can create several different entity files, each with different properties but they all use the same model.

```
<GAME
    InteractMode = "Push"
/>
```

Here you set different behaviours for the model. Push means that if you have the wood box in the game, you will only be able to push it around. If you change to ”grab” you will be able to pick it up and carry it around. If it's a door you use ”move” and that will give that function of being able to nicely move objects in a realistic manner with your mouse. These are typical doors, drawers, valves, sticks and so forth. This are can also contain several other settings, as always content creation doc and GameEntityVar are the places to search for more details.

The easiest way to get started with entity files it to always make a copy of an existing entity file and use that as a base for your new objects. Select an entity file for an object similar to the one you have created. In a long term perspective it is important to eventually understand what everything is in the entity file though.

So lets create an entity file for our ”mystuff_woodbox.dae” file. Start by copying the ”tut_woodbox.ent” file to your mystuff directory. Name it ”mystuff_woodbox.ent” and open it in a text editor.

```
<MAIN
    Name="tut_woodbox"
    Type="Object"
    Subtype="Normal"
/>
```

Change Name to mystuff_woodbox.

```
<PHYSICS
    SubName = "pCubeShape1"
    Collides = "true"
    HasPhysics = "true"
    Material="Dyn_Wood_Box"
    Mass="30"
    InertiaScale = "1 1 1"
    AngularDamping = "0.1"
    LinearDamping = "0.1"
```

```
    BlocksSound = "false"
/>
```

Make sure SubName is the name of your mesh in your 3D editor. To test having a different material, replace ”Dyn_Wood_Box” with ”Metal”.

```
<GRAPHICS
    ModelFile = "tut_woodbox"
    CastShadows = "true"
/>
```

Change ModelFile to ”mystuff_woodbox”. If you wish set CastShadows to false if you do not want the model to cast a shadow.

```
<GAME
    InteractMode = "Push"
/>
```

Set this to ”Grab” instead of ”Push”.

Now save the file and open it in HPL Helper. If everything works OK, you should see your wood box just as before. The only difference is that it should fall down to the ground by itself, if you open a .dae file directly it stays in the air until you interact with it.

Take not that in HPL Helper the InteractMode for Push and Grab makes no difference, this is only working when you have the object in the game. Which leads us to, of course, how to get the model into the game! For this to happen there are 1 more thing we need to do and that is the next chapter.

TUTORIAL 1.4 – REFERENCES

References are files that you create and import into your levels to tell the engine what entity it should load, where it should be loaded and what it should be named.

To create a reference open your ”mystuff_woodbox” project. This should be your work project and not the .dae file, but it really does not matter which.

Delete everything in the project EXCEPT the original mesh, the actual wood box. Rename the node to ”_ref_mystuff_woodbox_woodbox1”. As you might have guessed this is quite similar to how you created a collider earlier, ”_ref” this is the string telling the engine that it's a reference, ”mystuff_woodbox” is telling the game where to find the file and finally ”woodbox1” is what the object will be name in the game. The name is used when you for example write scripts.

Do not save the project, only export a new collada file. You can name it “ref_mystuff_woodbox.dae”. Now you have a reference file, but to see it in the game we need to create a level. But before moving on to Tutorial 2, where we create a level, we shall have a serious talk about exactly how reference files works.

The actual reference file is not important AT ALL. It's the ”_ref_mystuff_woodbox_woodbox1” that's important to get in the game. Where you store the reference file is of no importance, the reference file is only used by you to place objects correctly in the game. As far as the game is concerned you

can use any object you want as long as you name it correctly.

If you want to put a character in the game you do not need to have an actual character model as your reference, it works just as well with a cube as long as it is named ”_ref_characterfile_charactername”. But for you to know exactly where you place it in a level it's logical to use the exact character model as your reference, as well as being able to know what the reference is simply by looking at it. Therefore always create a reference that is an exact copy of the file you are referencing to.

The other important thing to know about references are: They point the game to the ENTITY file and not the MODEL file. When the game loads a level it finds all the _ref named objects and it then loads the entity files that the references points to. When loading the entity file it gets all the specific properties for the model and, as you might recall, the entity file contains the name of the model it sets it's properties for. Leading to the model being loaded into the game with the specific properties.

Now, lets create that level of ours so we can finally test a model in the game.