



# FROM BLENDER TO NETRADIANT



Written by:  
Sergey Bashkirov -aka - "z80"  
Antonio Latronico - aka - "toneddu2000"

Released under GPL License

# Chapter 1 - Installation

## I - Obtaining the script -

Download the latest exporter version: [link](#). It's zip-archive containing several files.

## II - Installing the script -

Unpack gained archive to blender scripts directory. One should enable show hidden files to find it. On Windows it is located in Blender/.blender/scripts. On linux this folder is hidden by default. For example, In Ubuntu linux it may be done pressing Ctrl + H, and in Windows selecting "show hidden files" flag in Explorer.

## III - Run the script -

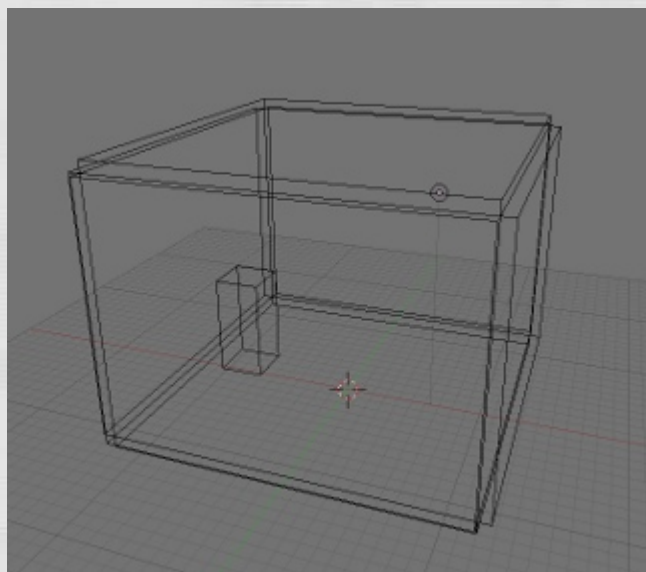
Now we have to run the script. Fire up Blender, slice a window and go to the Scripts Window, click on scripts - go up and click at the top "Update Menus". You will find the script in the export tab, named Nexuiz (.map). But we'll use it later. Now let's see how to create a default map for nexuiz.

# Chapter 2 - Using the exporter

## I - Blender units vs Netradiant units -

1. Create a box with these dimensions: 3.800 3.800 0.100 and name it **floor**.
2. Create a box with these dimensions: 0.100 3.800 2.800, name it **wall\_east** and snap it to the floor cube so they are united with their edges.
3. Duplicate **wall\_east** three times, name them **wall\_west**, **wall\_north**, **wall\_south** and snap each cube in the corners of the floor cube, as forming a room.
4. Duplicate **floor** and snap it to the top of all boxes and name it **ceiling**. Now scenes should be appear like fig 2\_01, if it's not, you made some mistake! :)

Fig 2 01 - Simple scene contained in the guide



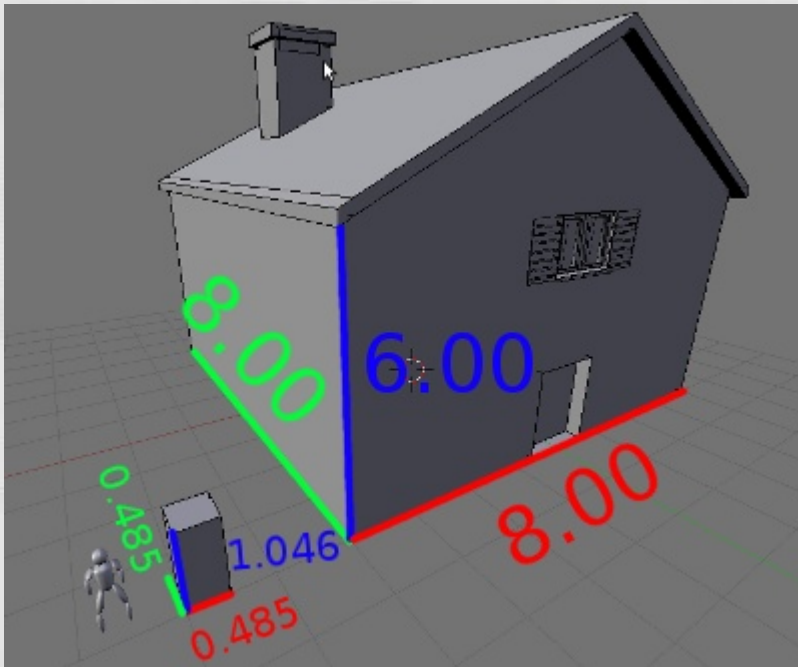
If you're in trouble use the `room_256_256_192_sample2.blend` file included in this guide. This box will be our first map and, once imported in Netradiant it has dimensions of a 256x256x192 room: it has dimensions multiplied on scale factor (32 by default) room. The scale factor is used for convenience. It need if one prefer working in metric reference frame instead of NetRadiant units.

Now, let's see different unit references to understand how to face working with units in blender and Netradiant.

An "info\_player\_attacker" (a normal entity for nexuiz that spawns a player) entity has

height 1.046, and 0.485 of width/depth. Instead, a small house will have these dimensions: 8.000 for width , 8.000 for depth , 6.000 for height. See figure 2\_02 for clarifications:

Fig 2 02 Comparison of Blender Units



## II - Getting materials -

Now, let's go back to our room and select `wall_east` object. Press `f5` and, in or browse to Links and Pipeline, section and name the Link to Object sub tab `concrete`. Then create in the texture tab a texture named `base` and set the Map input tab the UV button instead of Orco. This step is necessary to tune texture alignment in Blender UV Editor window. Now, press `f6` key and select an image texture. In the tab Image select in your `/Nexuiz/data/textures/` path the `evil2_basewall/dirtybrick.jpg`. Now select all the boxes except ceiling and floor, then select `wall_east` at last and `ctrl + L` - Make links to materials . For floor object choose a different material named `floormat`, as same procedure as above, select `evil2_basefloor/flr_cemtile_m.jpg`. select first ceiling, then floor and `ctrl + L`, make links to materials. Now we have to set up UVs for texture coordinates. If we don't do this, the exporter will export our map, but, once imported in Netradiant, it will pop up several error messages, saying that `texdef` is not well identified. So, select our first wall, `east_wall` for example, and go to edit mode (`TAB` key), edge mode, select all edges and press `Ctrl + e` and press Mark Seams. Now press `u` key and choose unwrap. Go to UV editor (`shift + f10` key)



and you'll see your uvs well cut; now open a image in the little box at the bottom of the uv editor and choose the one of the two that there should be there, `dirtybrick.jpg`. If you switch to 3d view (**shift + f5 key**) and pressing **Alt + z key** you should see your brick texture wrapping around the box. If you want to enlarge it, just go back to uv editor and select all uvs (**a key**) and scale them with **s key**. Don't worry if textures exit out of the UV layout, they work fine though. Do this for all textures, obviously for floor object use floor texture, opening it in the uv editor.

If you noticed, in the exporter , in the upper side, there's a comment about some properties that can be added to objects. Talk a little about shader property. What's a shader in Nexuiz? It's just a text file put in the `/scripts` directory saying something about the textures to use in that shader and what properties they have (transparency, lightmap, and so on). In this example we'll use a link to shader for our wall texture `dirtybrick.jpg`. When we assigned this texture first in the material panel, we pointed a link to `evil2_basewall/dirtybrick.jpg` because it was a texture but this time we do something different. Create a new scene (**ctrl + x key**) and draw a wall object. Press **f4 key** and go to properties panel. With the wall object selected, press **Add Property** button and a rectangular panel will show up at the bottom of the area. Choose **String** in the radio button on the left, in the field **name** write `texture[0]` and in the blank field write `evil2_basewall/dirtybrick` If you look at this last step, you'll see that we omitted `.jpg` at the end of the file name. This because we're pointing the exporter to shader name not to texture name. This will work for that shader file that has a different name instead of the texture name. If I, for example, create in the folder "mymetal" a diffuse texture named "metal.tga" and a bump texture named "metal\_bump.tga", but I write a shader named "metalpanel", with a lot of properties inside, I'll use **property** button to add a `texture[0]` property together with a `mymetal/metalpanel` path. Right? Otherwise for similar textures/shaders names is just needed to create a material and link it with the diffuse texture. The exporter will find bump, normal, gloss by itself.

Actually the procedure overwriting texture name is already automated. One may choose correct NetRadiant shader name, material index and press **"over"** button in the exporter's GUI panel. To do it one should 1) select an object, 2) open the exporter's GUI panel, 3) find **"overwrite texture"** text and fill **shader name** and **material index**, 4) press **over** button. After that when export this material texture file

name with index specified is overwritten with shader name specified for the object this procedure was performed on.

### III - Nurbs Surfaces -

Add new NURBS surface. NetRadiant NURBS surface's order is 2 and resolution is odd in an interval from 3 to 31. So when modifying NURBS surface in Blender keep this two notes in mind! And default NURBS properties are not compatible with NetRadiant. So if one tries exporting it as is the exporter would crop it's resolution to odd one in both directions.

### IV - Lights -

Due to NetRadiant light and Blender light are two big differencies some light properties Blender doesn't have could be set using Blender Logic section. To get default properties set one should call exporter's **prepare** button. After that logic properties page contains the following properties with default values: **sun**, **has\_target**, **target\_x**, **target\_y**, **target\_z**, **radius**.

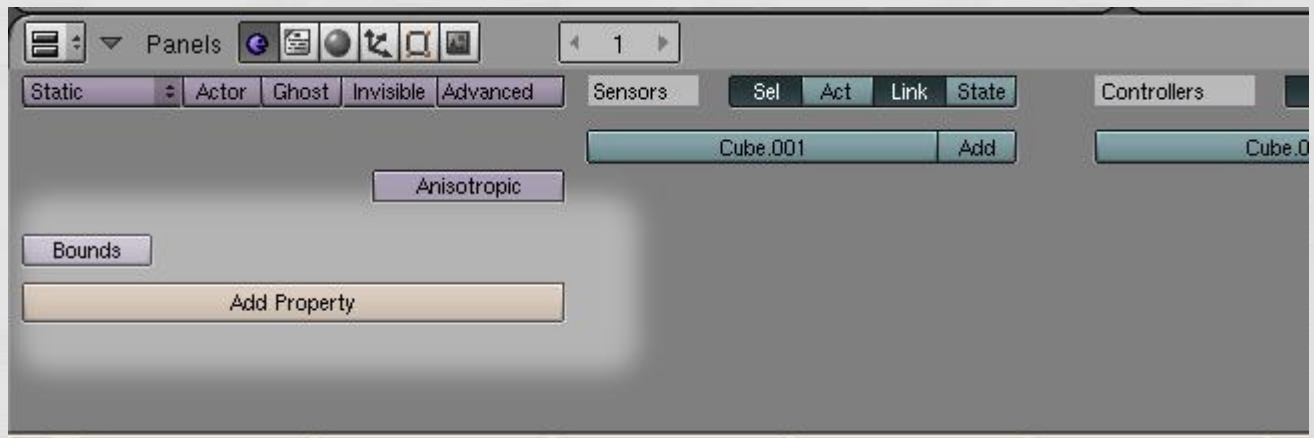
**sun** boolean property tells the exporter if it should treat this light as Sun NetRadiant light. If light is sun light then it passes through walls without any shadows and spreads everywhere. It's like ambient light in Blender.

**has\_target**, **target\_x**, **target\_y**, **target\_z**, **radius** are a set of properties for directional light. If **has\_target** property is true then on the properties from this set are taken into account. **target\_x**, **target\_y**, **target\_z** are directional light target point coordinates and **radius** is light spot radius.

Create a lamp in the middle of the room.

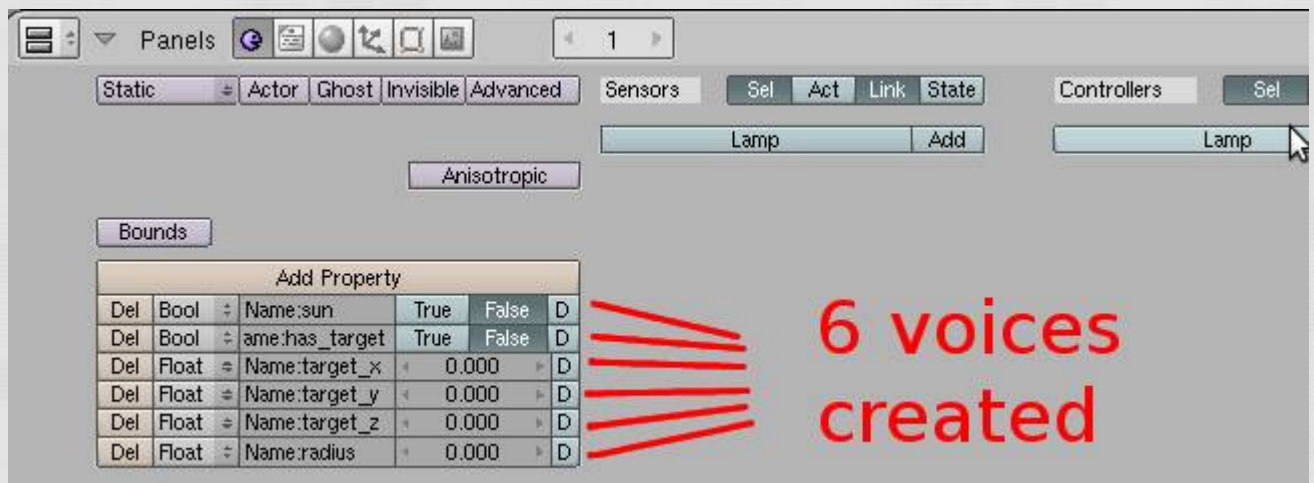
Select your lamp and press f4 to go to the panel. Stay here for a moment. Look at the figure 2\_04.

Fig 2\_04 - the Property Panel -



We'll touch only the property tab bottom, so you don't have to matter with the other buttons. With the light selected click on the right, on the exporter window, on the **Prepare** button and, magically, 6 voices have appeared in the property button on the left (on the 3d view). They are: sun, has\_target, target\_x, target\_y, target\_z, radius. You can find them here below in fig\_2\_05:

Fig 2\_05 - Light properties -



## V - Knowing the exporter -

The exporter has a few fields which should be filled before exporting anything. One should specify his(or her, if any :) ) Nexuiz absolute folder path. It is necessary because exporter creates shader names from texture file names specified as material nextures using Nexuiz path. It supposes all textures are in Nexuiz/data/textures folder (or any sub, subsub, subsubsub folder) when generating correct shader name for NetRadiant. (As you definitely know NetRadiant shader name is a relative file name without extension.)

Fig 2\_07 - Export tab



In the Nexuiz absolute path put (if you haven't set yet) your nexuiz path.

In the MAP file name, write your map file name, example room1.map.

scale factor tells the exporter how many NetRadiant units in one Blender unit. So if one prefers meters to units he (or she if any :) ) should set it to 32, if I'm not mistaken. (Because info\_player\_deathmatch entity height is 64 units and it corresponds to human height which is very approximately might be taken 2 meters).

extrude height is used when mesh is exported as a surface mesh. Each mesh could be exported in two ways. 1) the exporter may try subdividing it into several brushes and export them; 2) it could be exported face by face treating each face as individual object. In first case extrude height is not used. In the second case each face is transformed into pyramid with height being equal to extrude height.

If extrude downwards is true then pyramid's top is over the original face. In the opposite case it is over it.

console echo parameter tells the exporter whether it should print debug information in the console or not. That's all folks, kids!