

OVERVIEW

Arguably the greatest feature of Painkiller's PAIN Engine is its ability to render levels completely created in professional graphics packages. It allows artists and level designers to completely free their imagination - the tool is no longer a limiting factor. You can build your level using exactly same tools that are used to create CG special effects in Hollywood blockbusters. If you were to create a classic sculpture in any of the existing game engine editors you would quickly realize that the limited tools at your disposal are not enough. Well, the PAIN Engine allows you to create as much as *you* can, not as much as the level editor lets you. Same goes for lighting - you can use unlimited amounts of different light sources in your scene, you can tweak softness of shadows, tweak intensity and color of indirect reflected and refracted light on your lightmaps - whatever you like. And even though you can create very complicated data for your levels, the PAIN Engine will render them very efficiently, not only because it is so robust, but also thanks to the fact that you can manually set up zones, portals and antiportals that tell the engine to render only that part of level which is actually displayed on screen. No other tool can optimize your level better than you can.

Creating levels for Painkiller can be divided into following parts:

- Building level geometry
- Texturing
- Setting up lighting
- Rendering (baking) lightmaps
- Optimizing geometry visibility (zones, portals, antiportals)
- Importing level to PainEd and placing items (spawn points, ammo, teleports etc.)

Painkiller level creation and editing

In this tutorial you will learn how to create levels for Painkiller in Maya. I assume basic knowledge of Maya, but nevertheless even simple steps are described in detail so quite possibly even newbies will be able to follow this tutorial without any additional reading. In case there is anything unclear, you can find lots of tutorials on Maya on the Web. During the modeling and texturing phases only standard and easy editing tools are used. Lighting and lightmap baking is a bit more tricky but not difficult once you get the hang of it.

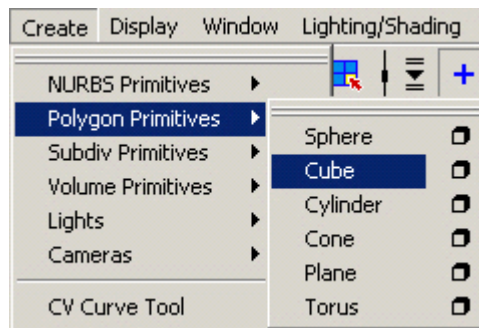
The tutorial is divided into several sections:

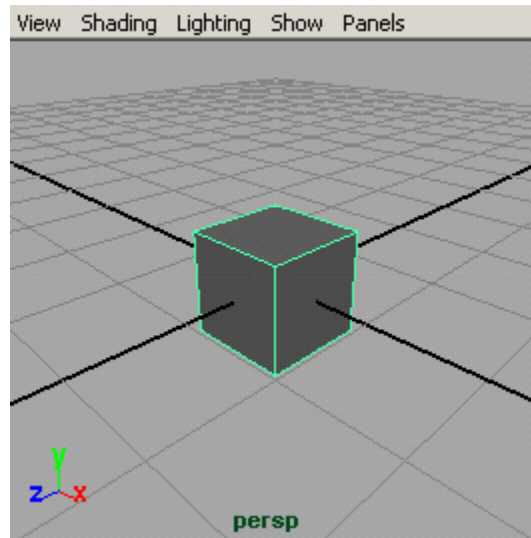
- Building Geometry
- Texturing
- Lighting
- Export
- Level Editing in PainEd
- Appendix 1: Creating Lightmaps in Maya

At the end of each section you will find additional, but often quite important notes.

Creating geometry

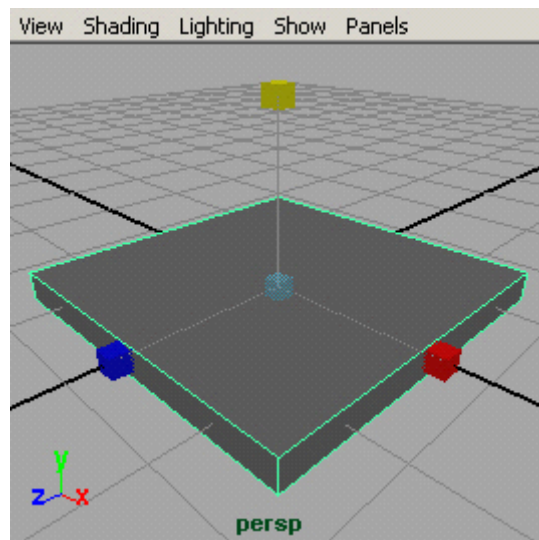
Let's build a simple scene consisting of few boxes. To create a box we choose from the main menu **[menu|create|Polygon primitives| cube]**.



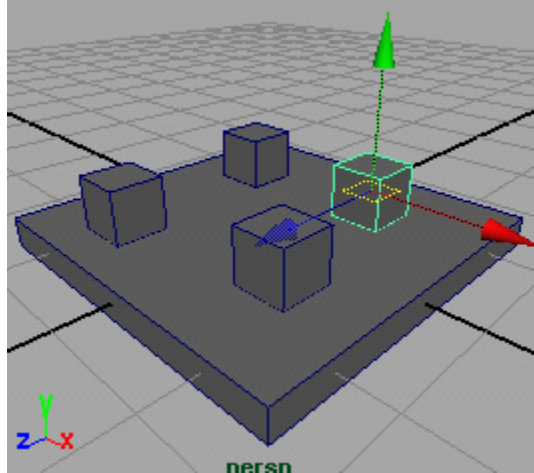


This will be the floor of our scene.

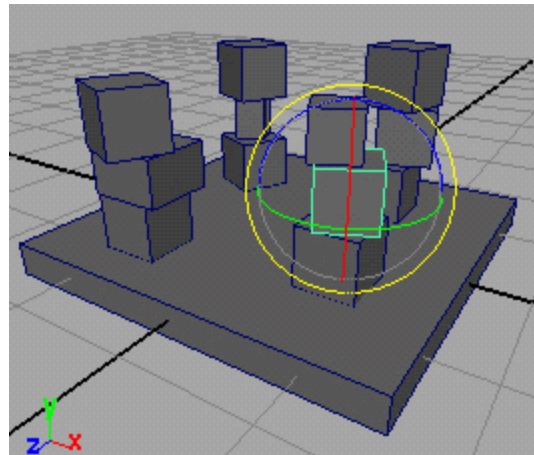
Let's edit the cube. First we scale it down along the Y axis - **scale** (shortcut "r") i scale it up along the X and Z.



We need some more geometry in our scene - lets create a new cube. Move it a bit to the corner of the 'floor', duplicate three times - **[menu | duplicate]** and place the new cubes near remaining floor corners - **move** (shortcut "w").



Duplicate even more of the cubes, move them up and rotate a bit - **rotate** (shortcut “e”)



Save the scene as level1.mb

Additional notes

- Only 'mesh' type of geometry can be exported to PainEd
- In Maya each object has two name fields. When you select an object you see it's 'transform' name. PainEd however uses the other name, for example when you create cube, you can see name "pCube1" while the name that PainEd reads is "pCubeShape1" (unless you renamed it of course). You can see the different names in **Attribute Editor** (shortcut “ctrl + a”)
- Mesh name can include additional strings that give those objects special attributes (like transparency, invisible barrier etc.) in PK. More on that in *Export* section.
- Single mesh can have up to 20000 triangles. Whole level scene can have up to 4096 objects but it is reasonable to

- keep no more than a couple of hundred objects.
- Remember to avoid long and skinny triangles - they are bad for rendering, bad for physics. Bad.
- Exported geometry is automatically triangulated, but you can also triangulate polygons manually to check if everything is OK - **[menu | polygons | triangulate]**.

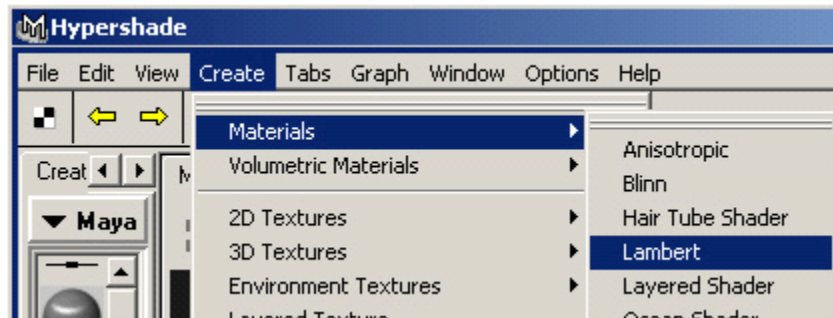
Practical notes

- It makes sense to delete history of your meshes from time to time - **[menu | Edit | delete by Type | history]**. It makes the scene file smaller, load times shorter and reduces the risk of errors in geometry.
- Polygons should not overlap or cross each other too much - it will waste lightmap space and increase likelihood of lightmap artifacts.
- Cleaning up geometry - **[menu|edit polygons|clean-up]** from time to time can help you maintain clean topology of your geometry.

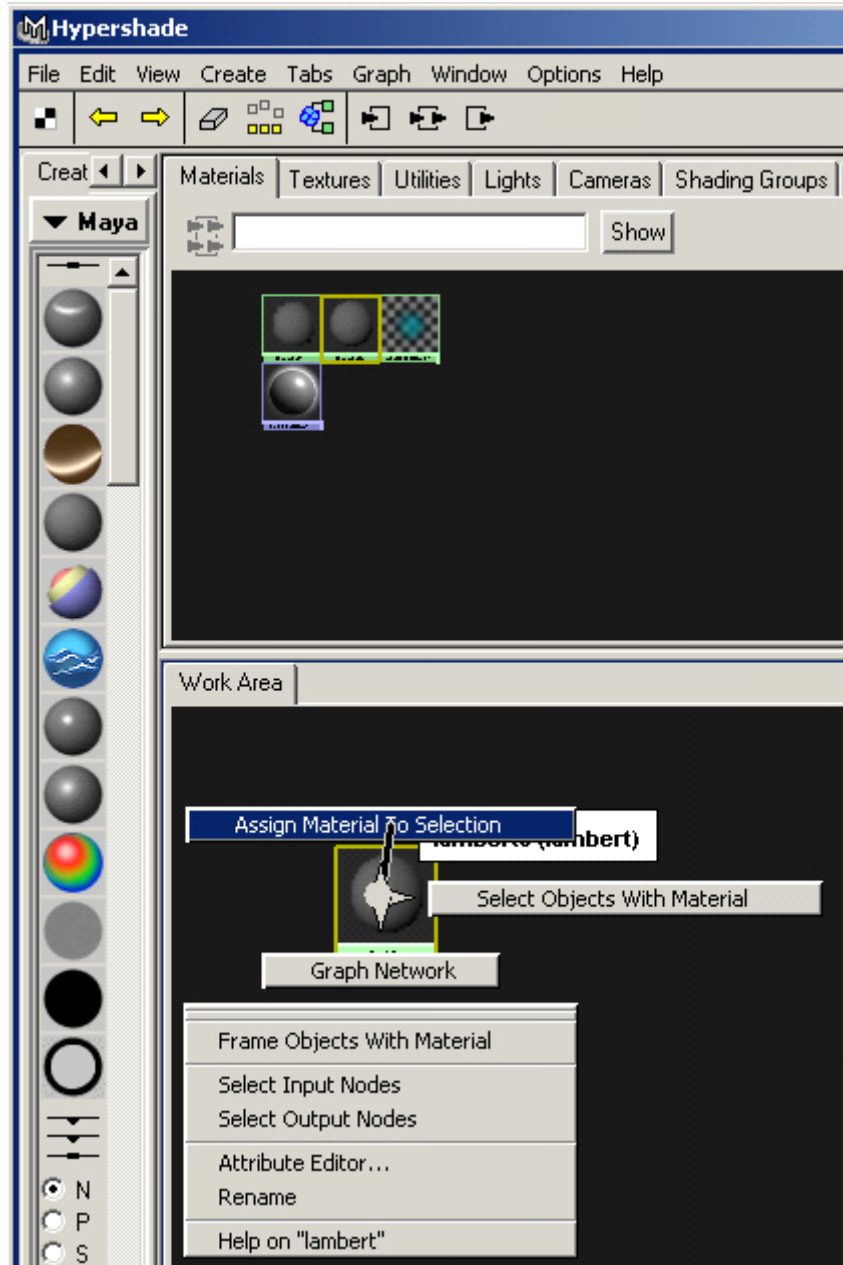
Texturing

To apply textures to our geometry we use HyperShade editor- **[menu|window|rendering editors | hypershade]**.

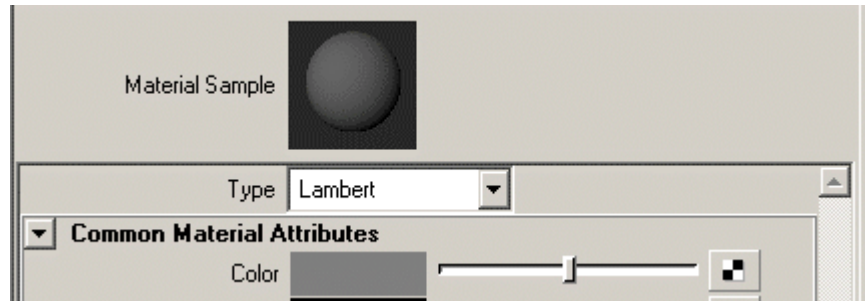
Let us create a new Lambert material:



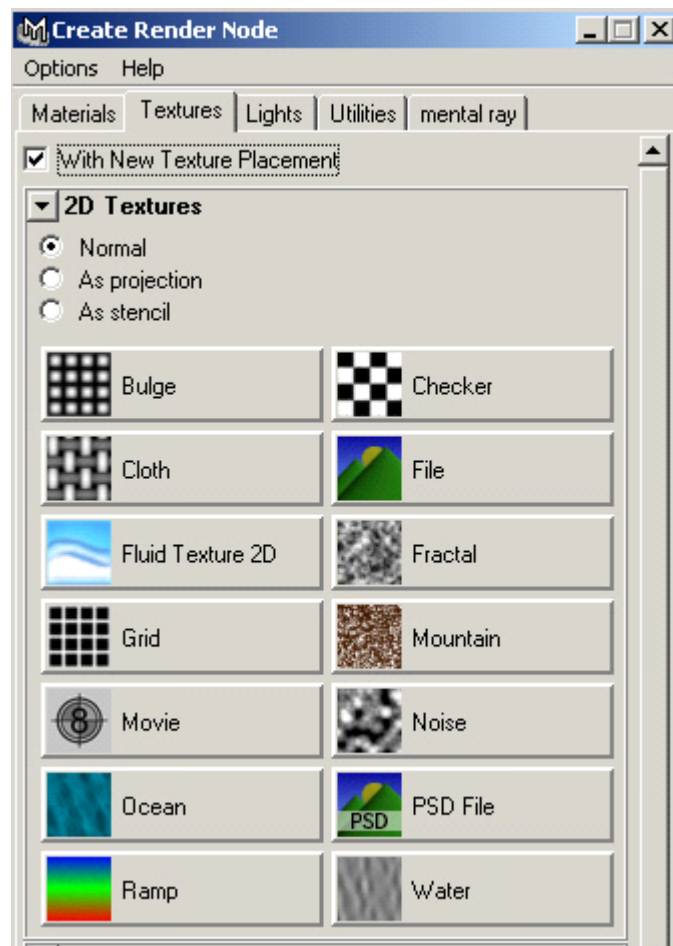
Select the floor object and assign material to it - in the **work area** of **hyperShade** move your mouse over material 'Lamber1', right-click it and choose **“assign material to selection”**.



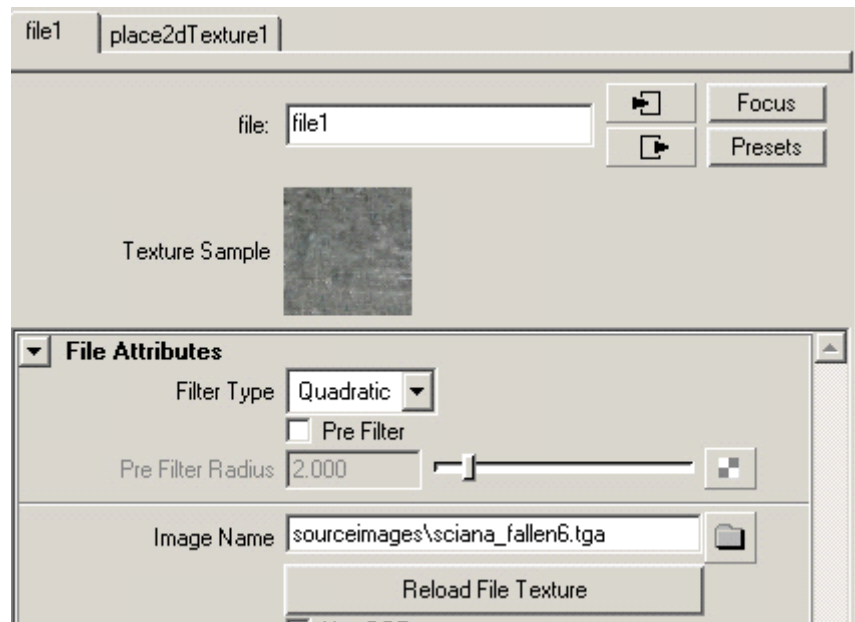
Doubleclick on your material in **hyperShade** and you will see another window - **Attribute Editor(AE)** of that node. Click the small checker button on the 'color' level.




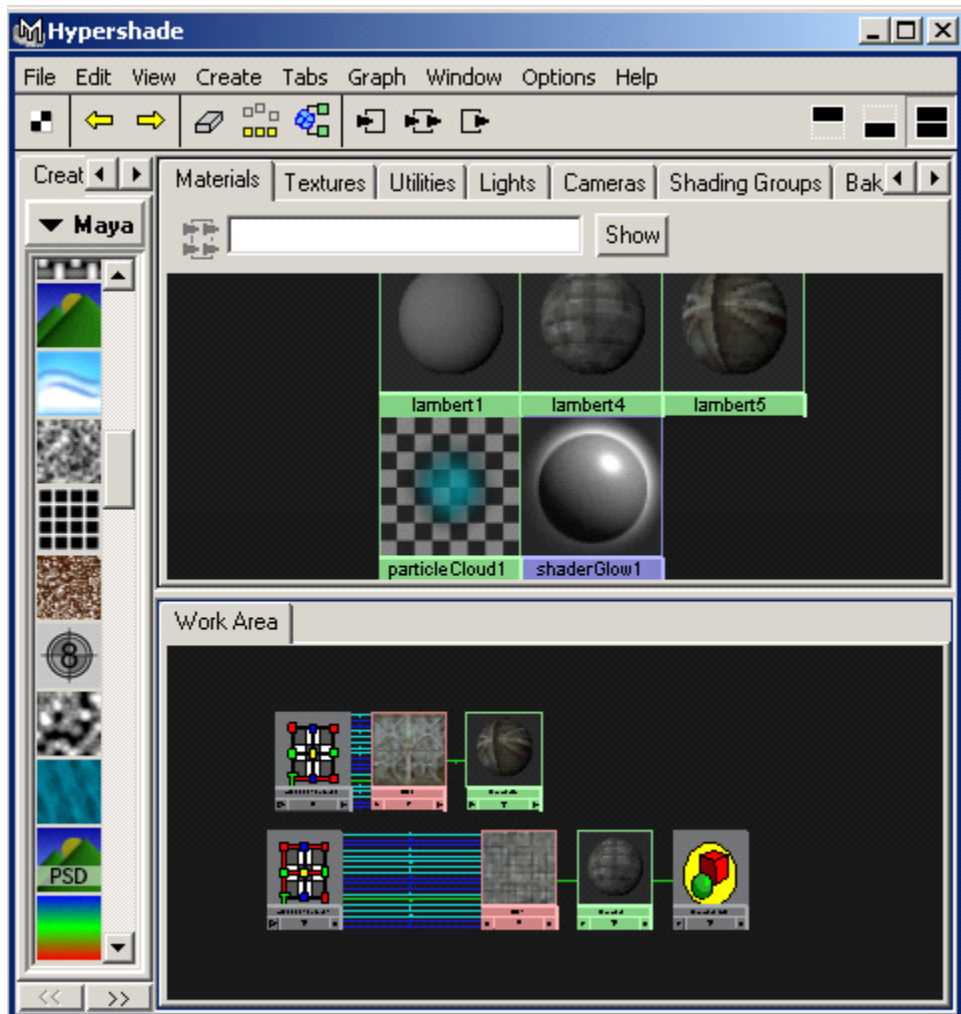
A new **“Create Render Node”** window opens. Click Textures tab then click on the File button in 2D Textures section..



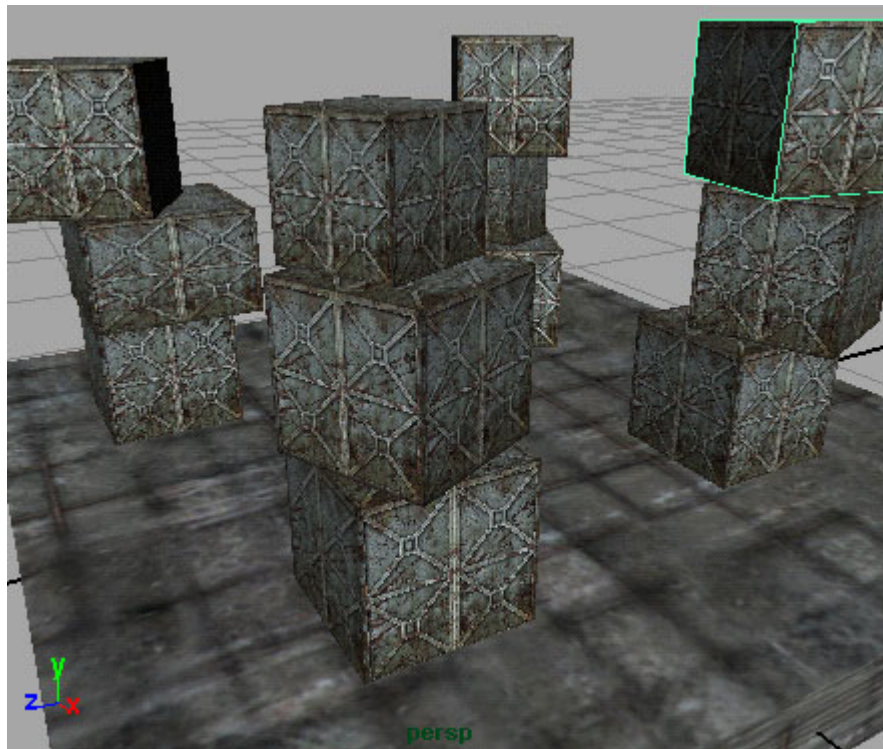
In the File Attributes section you can specify your needed texture file along with its path (you can browse your disk by clicking on the small button to the right).



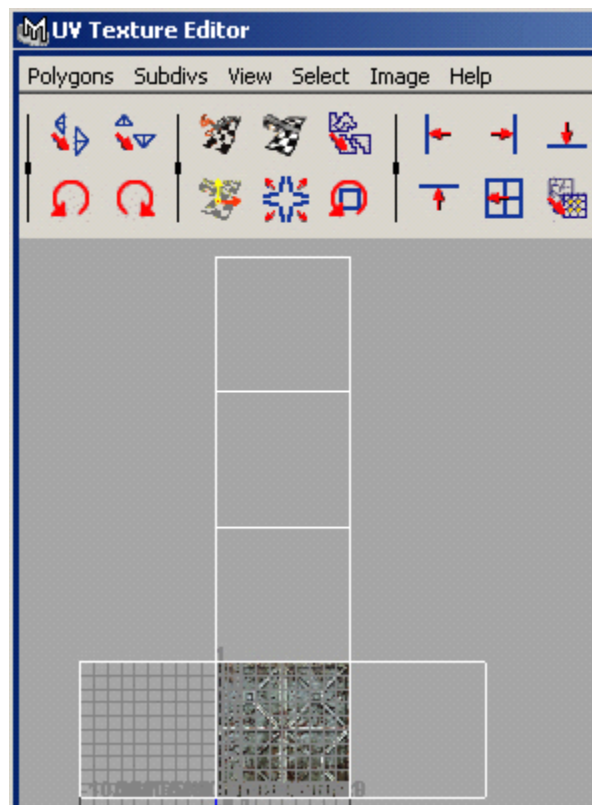
Create another material in the same way as in the steps above, select the remaining cubes in your scenes and apply this material to them. You can always check the graph of your material by clicking the  icon on the toolbar of **Hypershade**. It shows all materials and files assigned to the selected mesh.



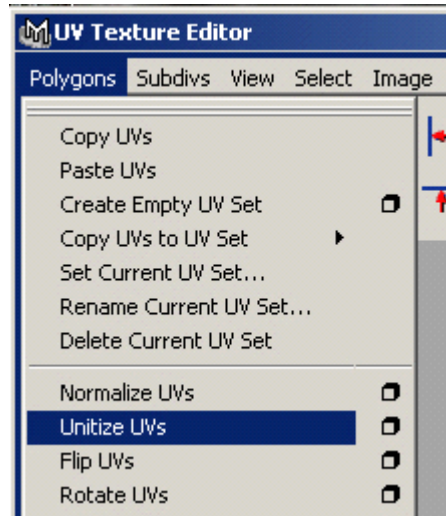
You can see the results of your texturing by switching your active Maya view port to textured mode - just press the shortcut key '6' to do that.



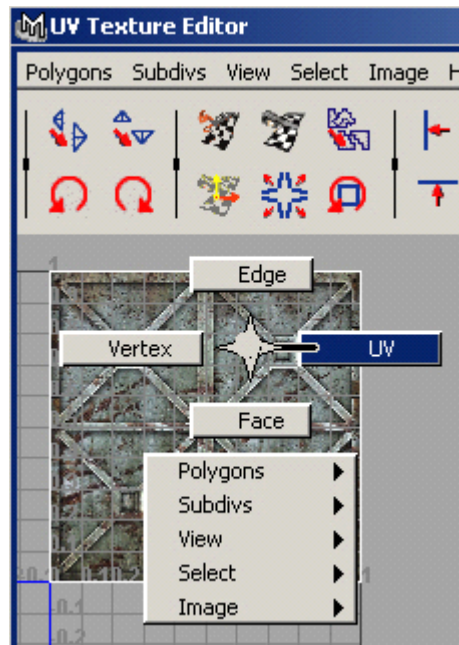
Now adjust mapping (alignment) of your texture. Select the boxes and open **[menu|window|UV Texture Editor]**.



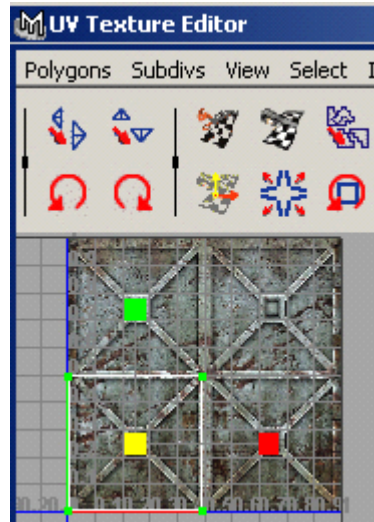
Select **Utilize UVs**



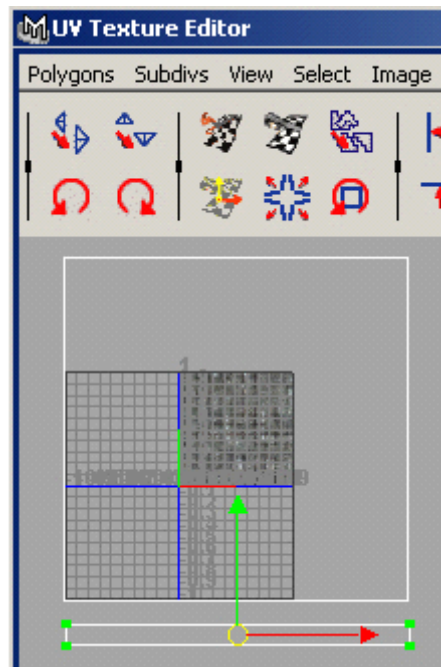
That will normalize mapping coordinates of each cube to a square. You can select UV points (vertices of the shapes that represent sides of your cubes) in **UV Texture Editor** by right-clicking and selecting UV from the pop-up menu:



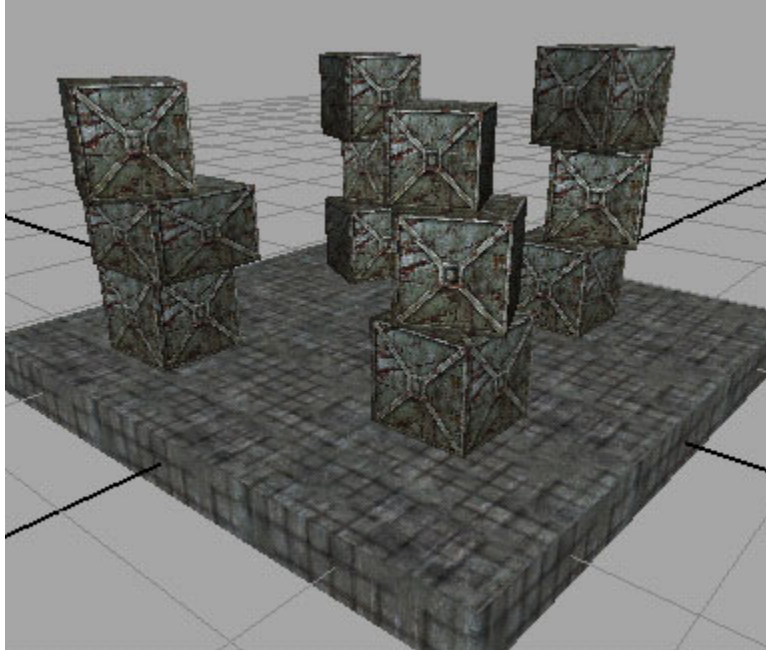
Using the **move** and **scale** tools adjust the UV points.



Adjust floor object mapping as well.



Now it looks like this:



Now we are ready to export our sample scene and load it into PainEd.

Practical notes:

- Remember to add texture to the 'color' attribute of the material, other attributes are not exported to PainEd.
- Transparency can be set for an object by adding "trans" to the object's name. Of course apart from the name change you also need a texture with alpha channel (transparency information). Sorting of visibility in PK is done by objects and not by faces, which means that if you want to have a transparent cube and be able to see one side through another you will need to have each side as a separate object.
- There are many different material types in Maya but you need to stick to Lambert material for PK purposes.
- If you want to tile the texture on your object just do that by scaling the UV coordinates. Other methods of storing texture tiling information (tiling parameters in file node or "place2d") are not supported by PainEd.
- PainEd ignores texture path and texture extension, so there is no problem if you assigned texture "c:\a\sourceimages\delta.bmp" as long as you have a corresponding .dds or .tga texture in PK\data\textures\levels\mylevel\). If there are multiple versions of texture in different formats available, PK uses this order: 1. .dds 2. .tga 3. .bmp

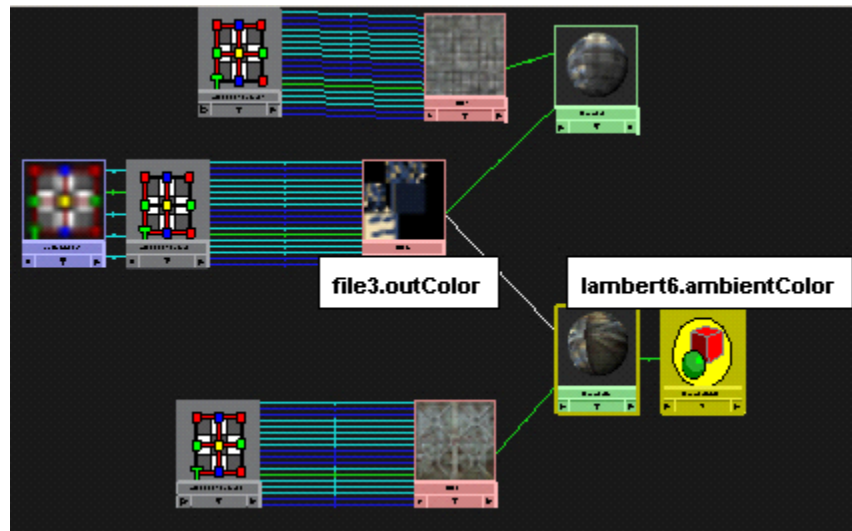
Lightmaps

Lightmaps are used for static lighting in Painkiller. They are bitmaps containing lighting information rendered (or 'baked') in Maya (see Appendix 1: creating lightmaps in Maya).

Practical notes:

- Lightmaps must be nodes of "file" type linked to "ambientColor" of object material.
- It is best to combine as many objects as possible and render a single large lightmap for all of them together - It renders faster in the engine than many small lightmaps.
- One object can have only one lightmap but many objects can share the same lightmap (after lightmap rendering you can separate previously joined geometry into separate objects).
- Lightmaps are placed on the geometry like any other textures, but instead of the 1st UV channel they use texture coordinates from 2nd UV channel. If an object doesn't have a 2nd UV set defined, then lightmap can't be linked to it and this object is lit in engine by realtime lighting.
- UV coordinates for lightmaps should not overlap, unless of course you wish to have exact same lighting on different parts of geometry.
- UV coordinates of lightmaps must be normalized (meaning they have to be contained in the 0-1 square).
- If a high level of brightness and intensity is desired on lightmaps then 'overbright' option has to be enabled in level settings in the editor. This setting multiplies brightness of lightmaps by two, which means that lights used for rendering lightmaps have to be around two times darker than usual. If normal brightness of lights is used for overbright in engine then color artifacts will occur.

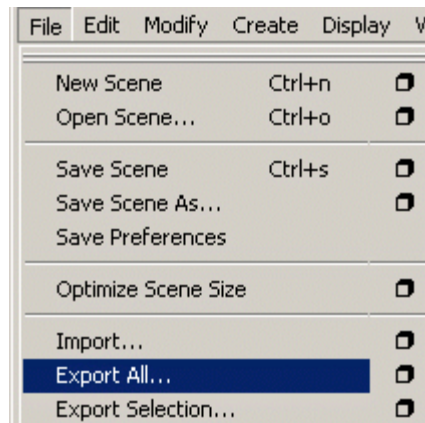
This is an example of two materials with lightmap on a single object.



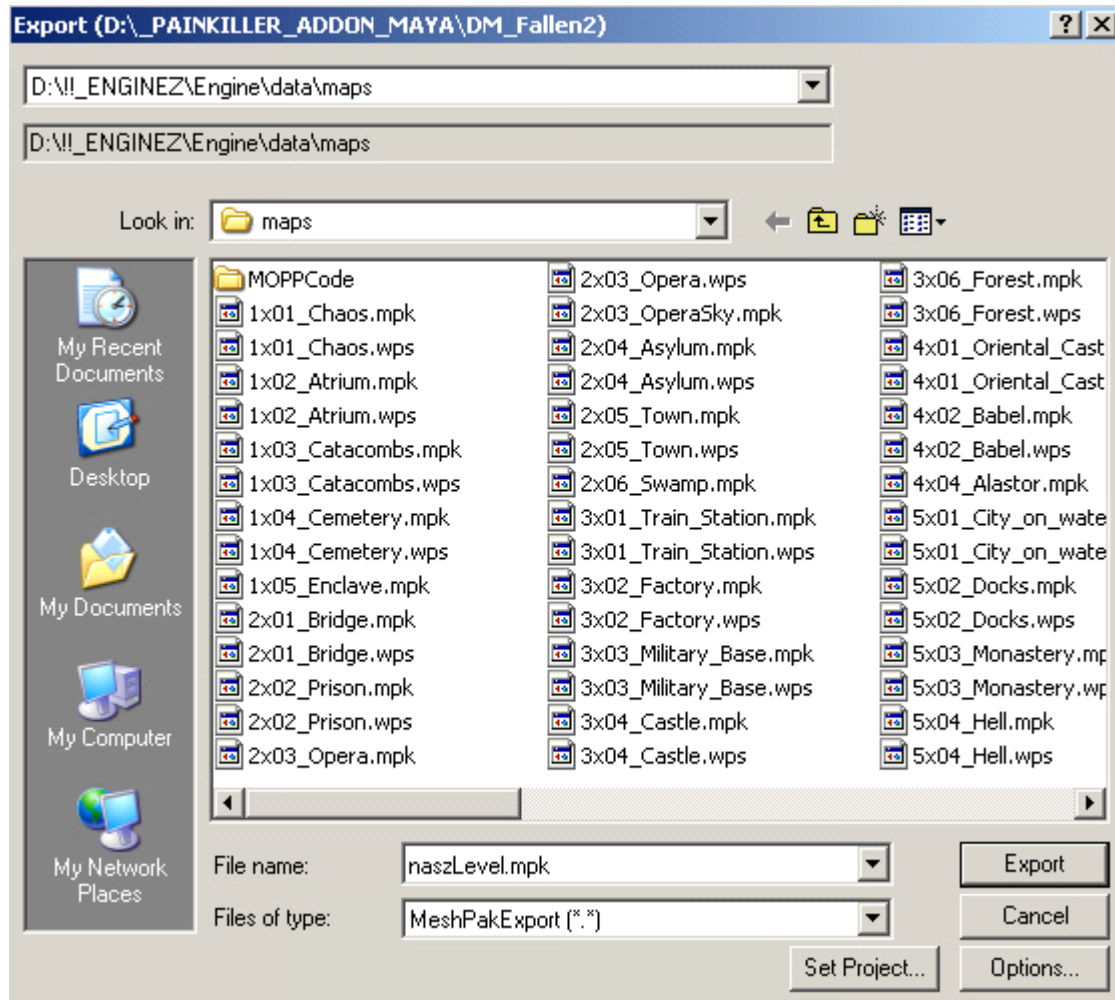
Export

Load **PCFMeshPlug.mll** (it should be placed in \bin\plug-ins\ dir) plugin in:
[menu|window|Settings/Preferences|Plug-in Manager]

You can either “**export all**” geometry or just “**export selected**”.



Export the map into directory “C:\Program Files\Dreamcatcher\Painkiller\data\maps\” (alter the path if you installed PK in different dir).



Textures and lightmaps must be placed in the directory "C:\Program Files\Dreamcatcher\Painkiller\data\textures\levels\mapname", where 'mapname' is the name of your exported .mpk map.

Practical Notes

- No matter how small or how huge levels you create in Maya you can always adjust them to the right scale in PainEd using Scale parameter. However it is best to stick to the default 0,3 scale and scale your geometry in Maya to fit editor scale.
- If you fall through the floor or other geometry in PainEd it means that after a geometry change the Havok representation of geometry was not updated. You have to delete the directory with your map's name in \data\maps\MOPPCode\ and reload the level. The engine will recreate the proper physics representation of your new geometry.
- File \MPK Substrings.txt in \Docs dir of the game contains the list of attributes that can be given to objects in your map.

Creating level in PainEd

File Paths:

Editor

"..\Painkiller\Bin\ PainEditor.exe"

Map geometry *.mpk

"..\Engine\Data\maps"

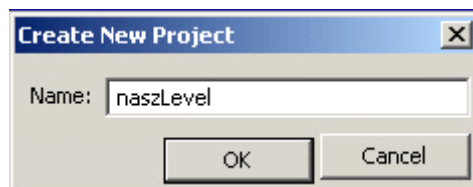
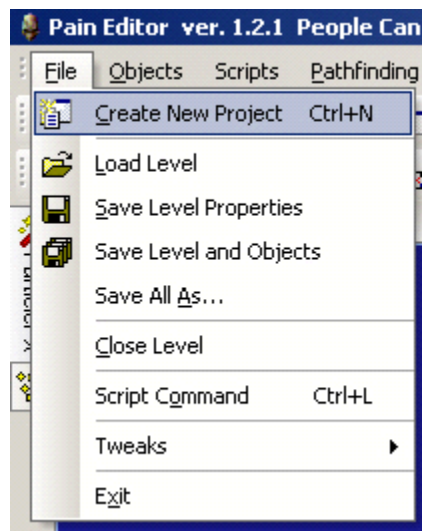
Level settings, items etc.

"..\Engine\Data\levels\levelname"

Level textures and lightmaps

"..\Engine\Data\textures\Levels\levelname\
"


Launch the editor and select "Create New Project".

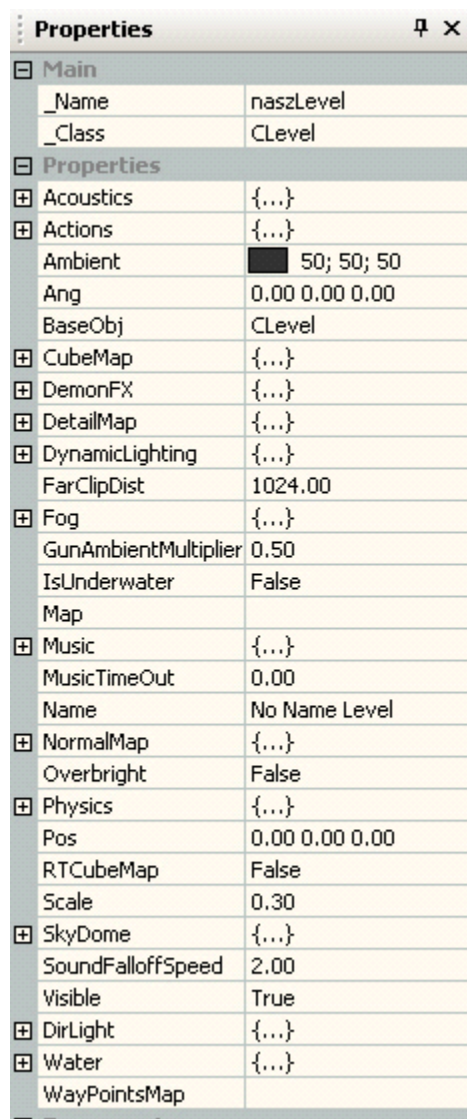


It is best to give your Project level exactly the same name as your .mpk exported map name - it will help you avoid confusion.

In the **Game objects** tab right-click on your level's name and choose **Save**.

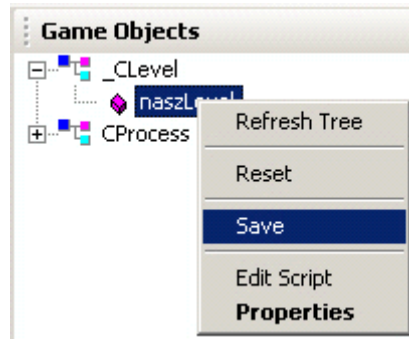


Reload your newly created level by clicking  (“**Reload Level Objects, All scripts And Map**”) on the toolbar. After that you can doubleclick on your level's name in **Game objects** tab and **Properties** of your Project level will open.

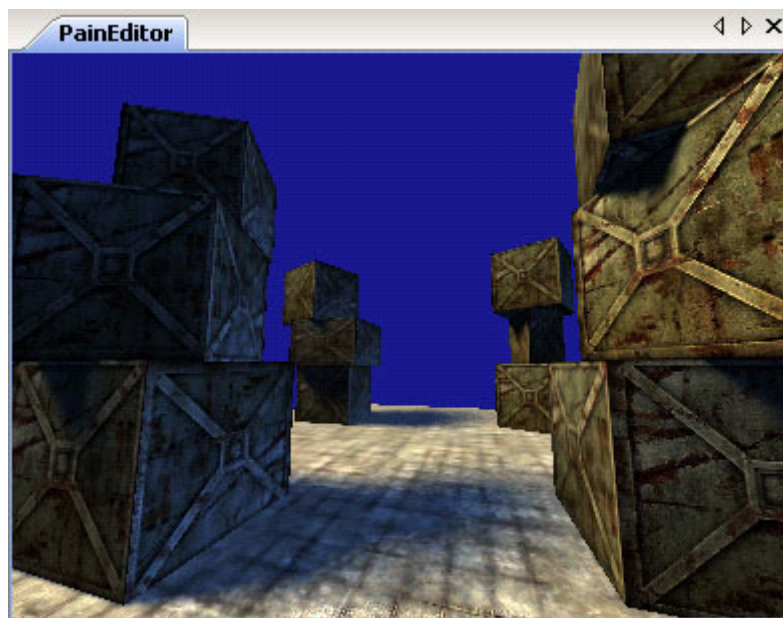


Click on the Map field and choose your .mpk map file. You should see your level

in main window now. Save your level like before.



If you placed all the textures and lightmaps in proper directories you should see your map in its whole glory. You can change the Overbright setting to True (this is the desired setting as it makes lighting appear more intense and vivid. After the change you have to save and reload your level. If the scale of your level is not right you can adjust it in the editor but it will be better to rescale your level in Maya as changing the default scale in the editor might influence physics behavior. You can hit "F" key and start walking around your level.



Here are the basics of working with PainEd:

You can rotate the camera of the viewport with RMB (Right Mouse Button)
You can move the camera in the viewport forward or backwards with LMB (Left Mouse Button)
You can move the camera in the viewport sideways with LMB+RMB buttons
You can select objects in the map by shift_clicking on it

You can move selected object in the map by dragging red, green or blue axis of the object with shift+LMB

You can rotate the selected object along the red, green or blue axis with shift+RMB

You can scale the selected object (like CBox, CEnvironment etc.) with shift+ctrl+LMB. Scale of items has to be changed numerically in the properties of each item.

After you have imported your map geometry into PainEd you need to place some respawn points. Hit Ctrl+O to bring the Create New Object window, select CArea class, type the name for the spawn point and hit OK. Respawn point is created, but you won't see it in the map until you click on Edit Areas icon in the toolbar. The small green square is your respawn point, the red line pointing from it is the direction that player is facing when he respawns. When you select this respawn point (Shift+LMB) you can change the direction of the red line with z and x keys.

After you're done placing and tweaking each respawn point you have to save it. You can either rightclick on the name of the respawn in Game Objects window on the right and choose Save, or if you want to save multiple respawns you can rightclick on the CArea class in same window.

We can now start adding items. Most of the item types are predefined and can be found in Templates tab of window on the right. There are all sorts of ammo boxes, armor, weapons, power-ups, even a jump pad. Many of the objects can have their properties tweaked, for example you can edit the strength of a jump pad. Remember to save each item or all of them together after tweaking.

You can also add teleports to your map. In PainEd Teleport is a virtual box of a CBox class. It means that teleport itself does not have any geometry – it's invisible. Therefore you must prepare some geometry for your teleport in Maya, and then place the actual teleport in the place of geometry, otherwise you won't be able to see a place from which you teleport! Apart from the teleporter itself you also need to define a place where the teleport will send you. This point is of identical class as spawn points (CArea class). After you have created the point of destination you simply provide its name in the properties of teleport and you're done.

These are essential parts of making a MP map. Save all of your work, launch MP and test your map! Keep in mind that unless you FIRST launch MP and THEN do editing you will be in SP physics mode which is significantly different from MP physics (for example rocket jumps in SP are disabled). After you're satisfied with your work you will definitely want to share it with others - easiest way to do it is to create a single .pkm pak containing all the data of your level. Bring up *Choose a level* window (File\Load Level), select your level from the list and click on the Create PKM button. This will generate a .pkm file in \data directory of your game.

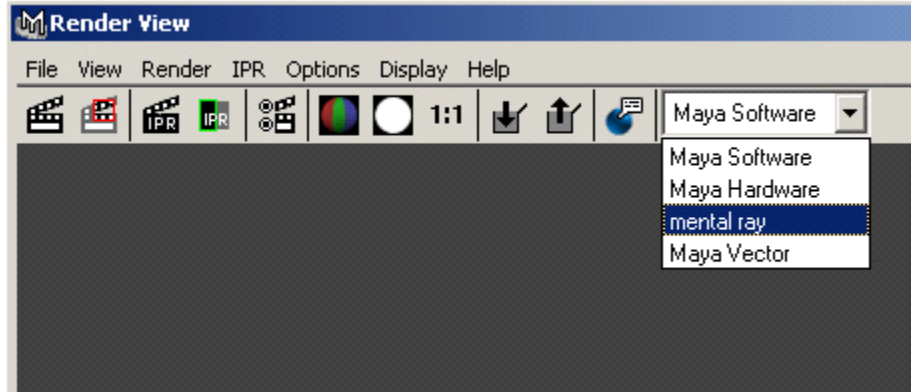
Appendix 1: creating lightmaps in Maya

This tutorial focuses on rendering (or baking) lightmaps using Mental Ray in Maya 6.0, however Mental Ray can be used for rendering lightmaps in Maya 4.5 and later versions.

1. Basic steps

Load plug-in Mayatomr.mll in [**menu|window|Settings/Preferences| Plug-in Manager**].

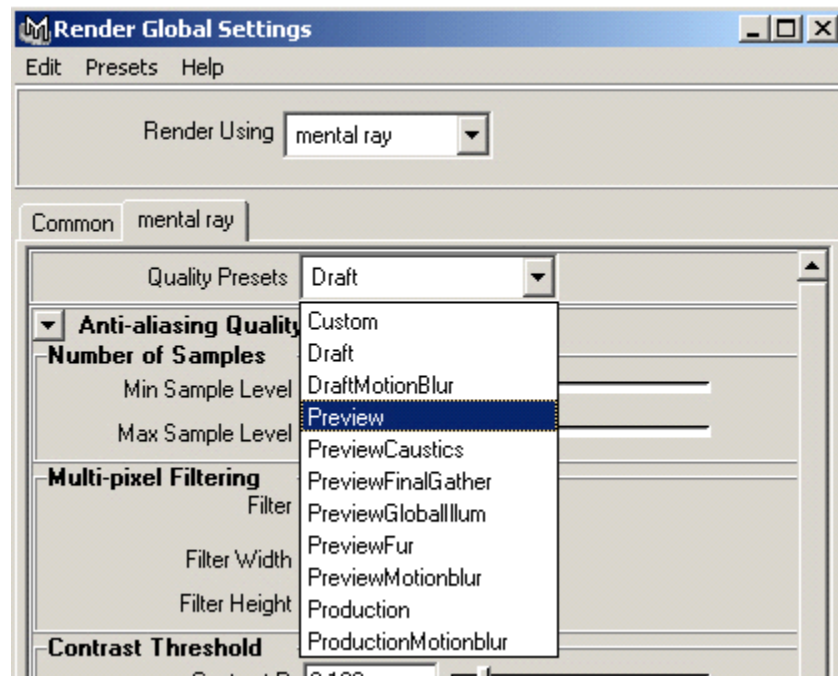
Open window [**menu|window|Rendering Editors| Render View**] and choose Mental Ray renderer.



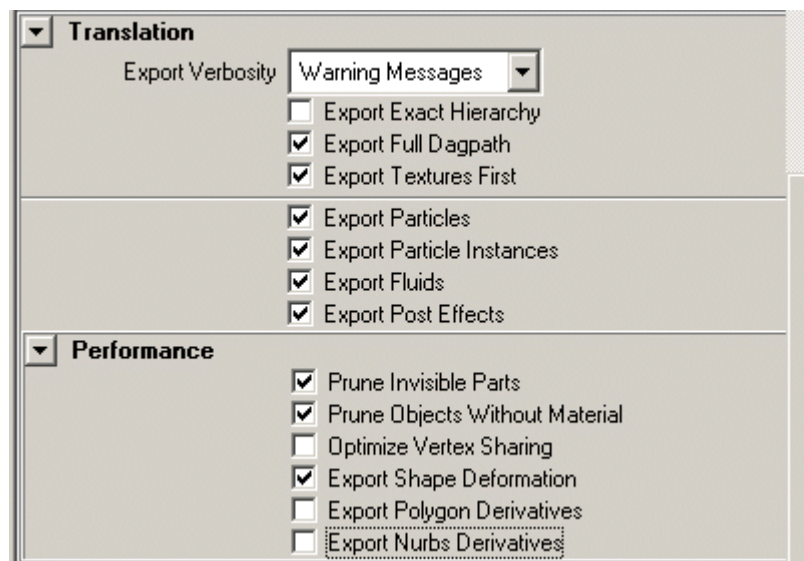
Open **Render Globals**



Open **mental ray** tab and choose "Preview" from "Quality Presets".



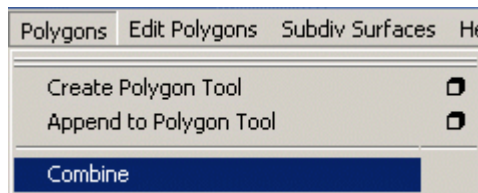
Set all options like in the screenshot below.



2. Combining object into larger entities

We want to have as many objects on a single lightmap as possible. The fewer lightmaps the faster your map will render. It's hard to tell exactly how many objects can share one 1024 by 1024 pixels lightmap, but usually we pack around 25000 triangles worth of geometry on such lightmap. Of course if your entire gigantic level consists of only simple cubes than things will be different - you'll have to experiment. For test baking of your lightmaps you can of course work on

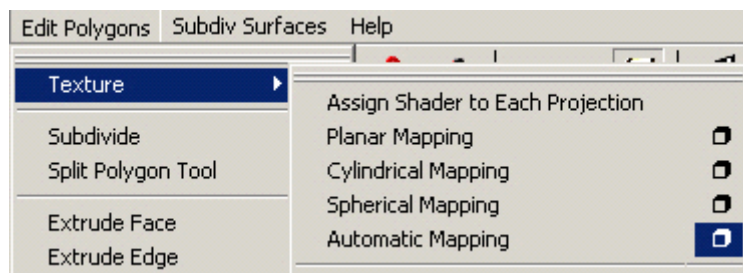
single small objects but keep in mind this is not optimal way. For this tutorial we will divide our scene into two lightmap objects. Choose all 'column' objects and combine them together: **[menu|Polygons|Combine]**.



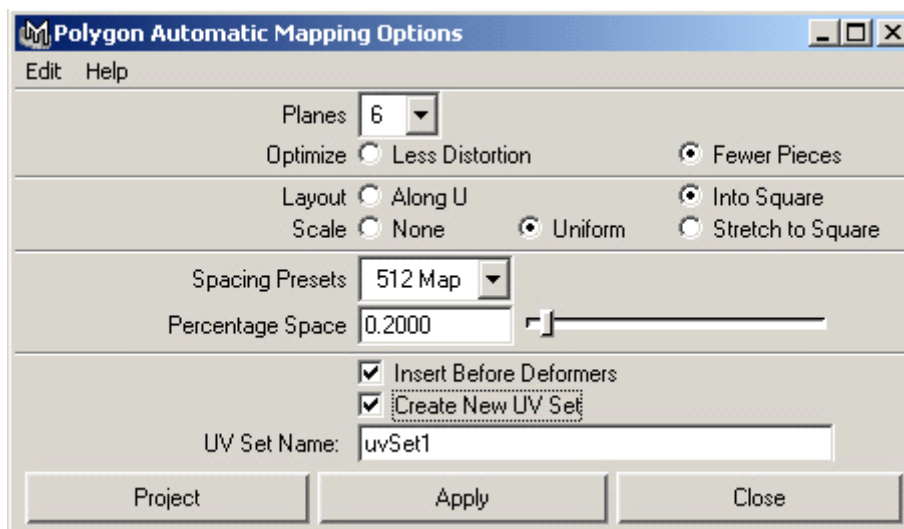
We leave our floor box as a second object with separate lightmap. Now our scene has two objects to bake lightmaps on.

3. Assigning second UV set

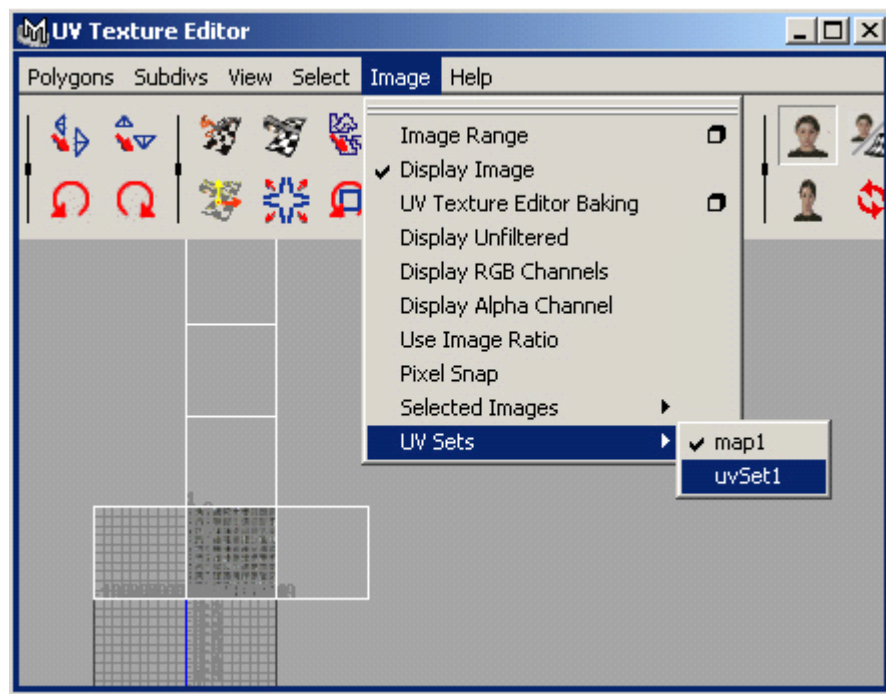
We have to create a second UV set that lightmaps will use. First we have to freeze any transformations of our geometry: **[menu|modify|Freeze Transformations]**. Now you can generate mapping coordinates for our lightmaps on second UV set - choose: **automatic mapping tool**.



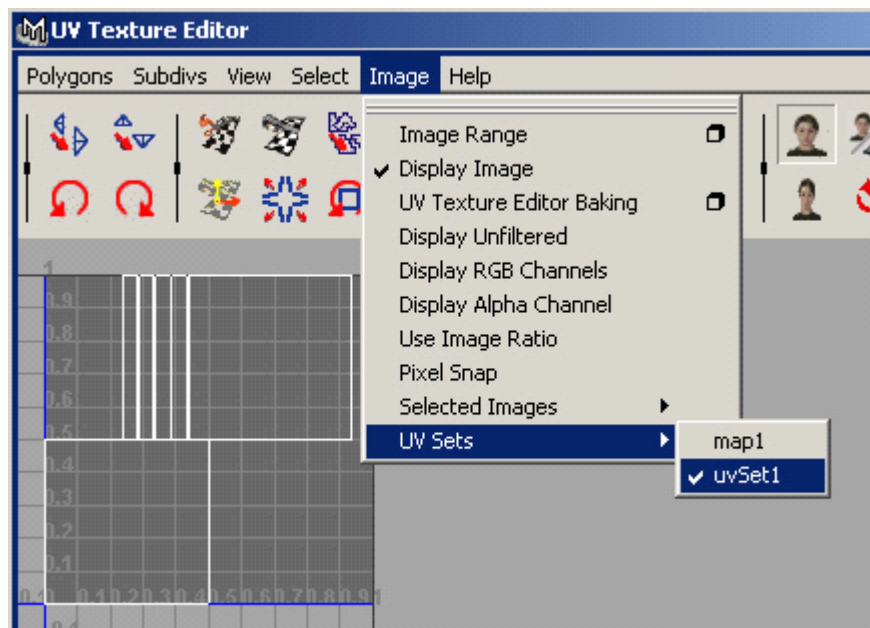
This option generates automatic mapping coordinates for lightmaps. We will layout the UVs for a 512x512 lightmap



After clicking **Project** or **Apply** we can check out the result: Open **[menu/window/UV Texture Editor]**.
Choose the new UV set:



Remember, if you want to re-generate automatic mapping (if you want to set different parameters etc.) you have to turn off **Create New UV set** otherwise you will end up with a third UV set.

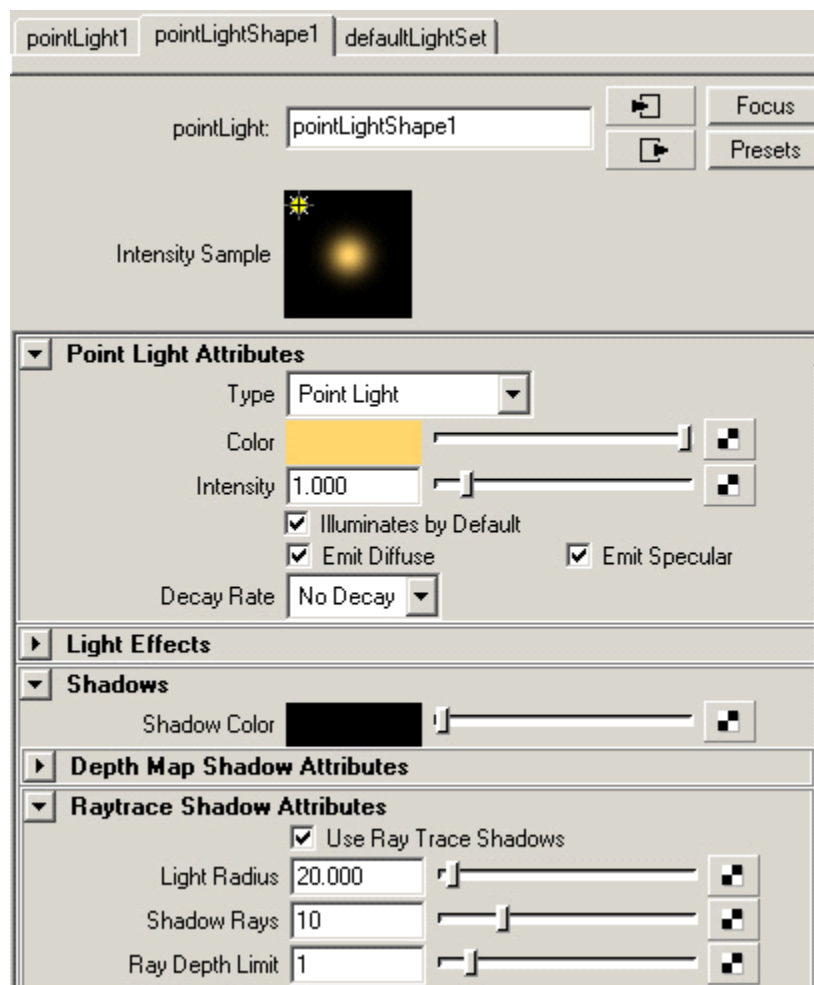


4. Separating materials

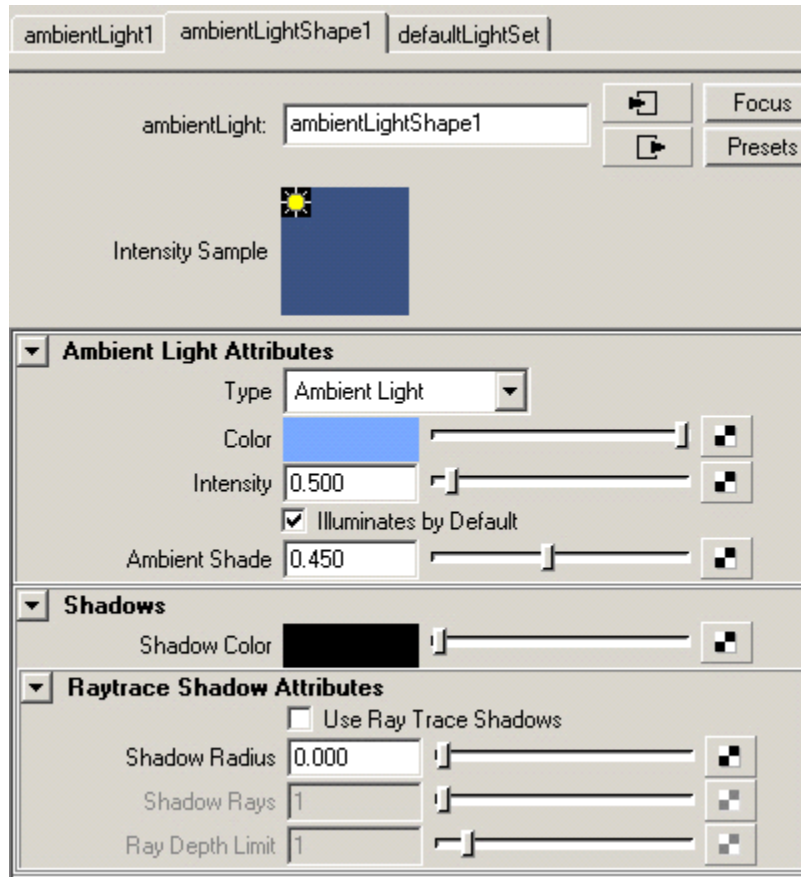
If different lightmap objects share common material (for example if you have two walls with the same brick material and you want each of the walls to have a separate lightmap), you have to copy those materials so that you have two identical but separate materials, otherwise you will have only one lightmap on all the objects with same material. Easiest way to do that is to export selected objects to a separate .mb maya file and then to import it back into the scene. The objects and their materials will still look the same but they will actually carry different material ID than they had before this operation. In our example it is not necessary as two objects in our scene have different materials.


5. Setting up lighting

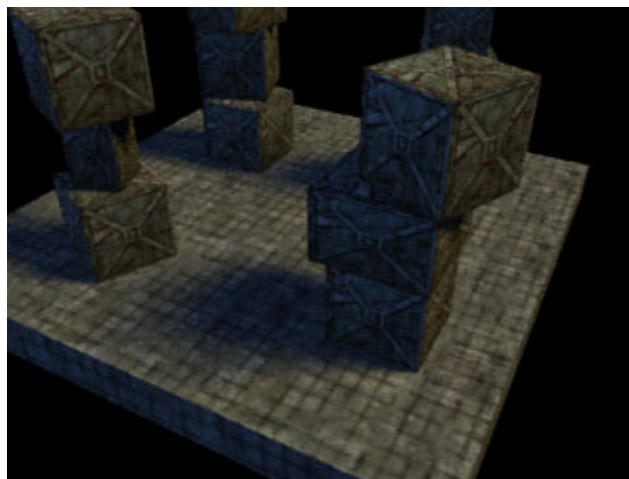
There are many ways to light your scene, we will show you the easiest way using simple point light source casting soft shadows. Add a point light source **Create\Lights\Point Light** with the following parameters:



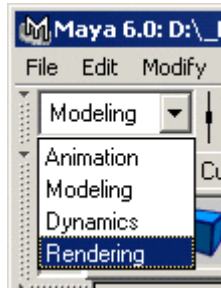
Create another blue ambient light **Create\Lights\Ambient Light** with the following parameters:



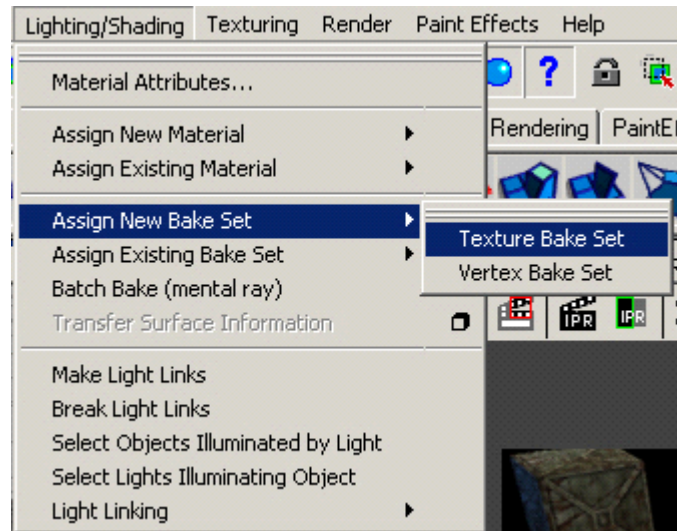
Place the lights reasonably in the scene, click render: **Render View**  and you will end up with something like this.



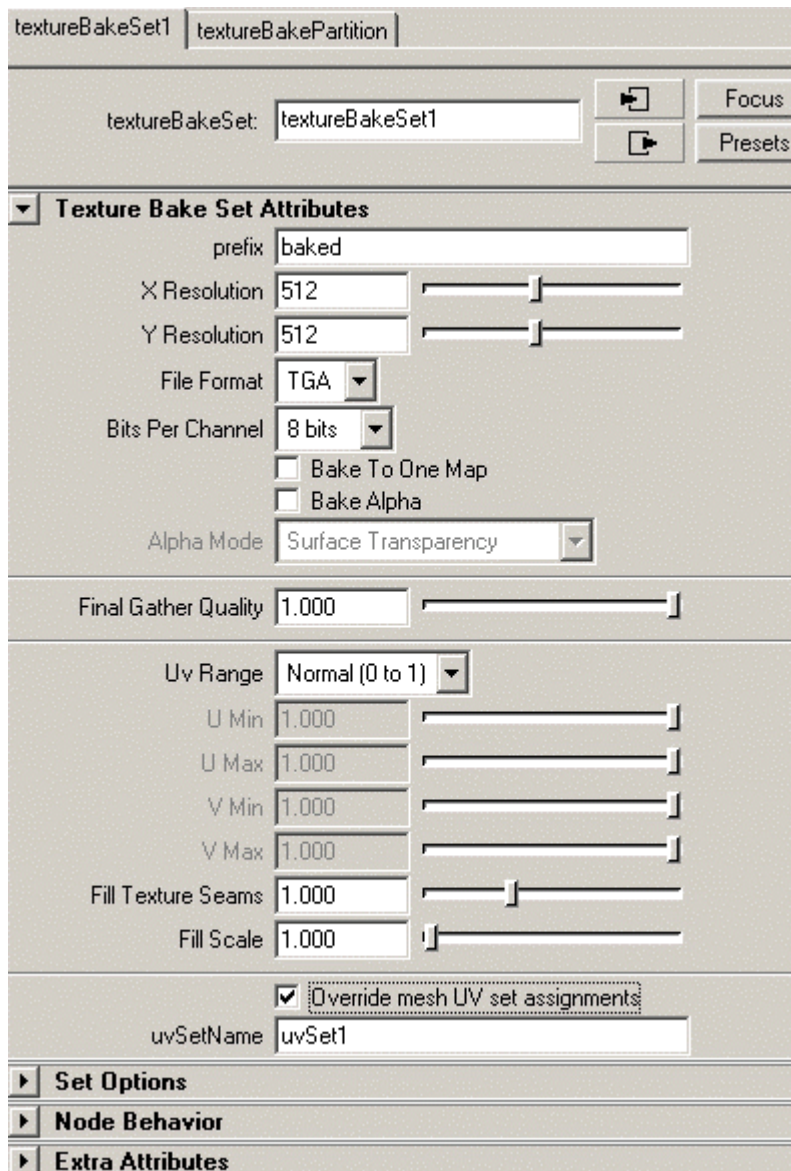
Change Maya module to rendering.



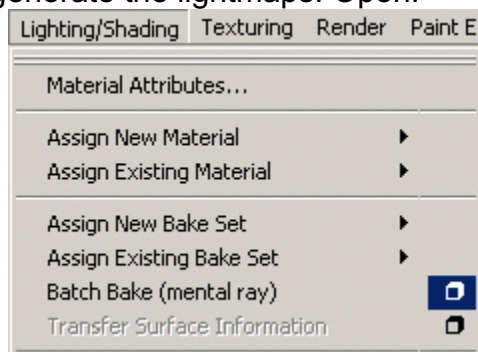
Select both objects and choose **Assign bake set**.



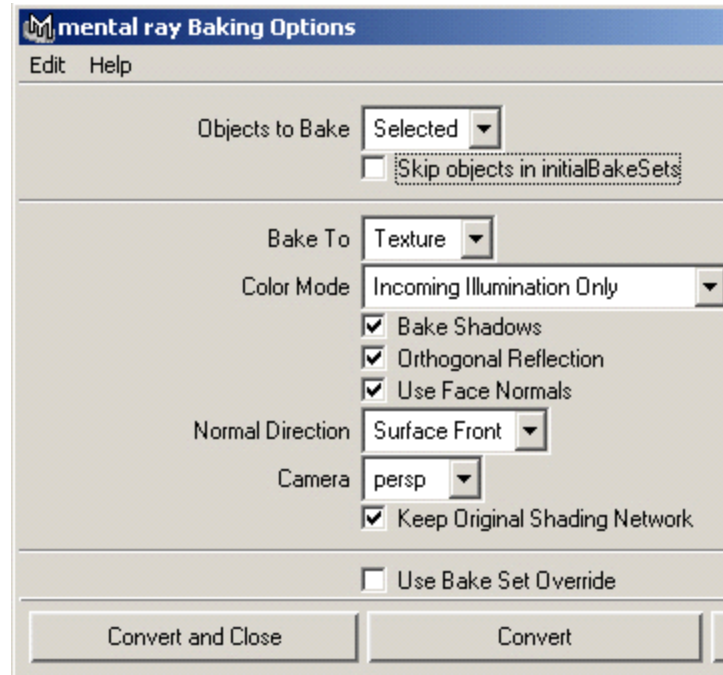
Now you can check all the settings for rendering of the **bake set**.



It is now time to finally generate the lightmaps. Open:

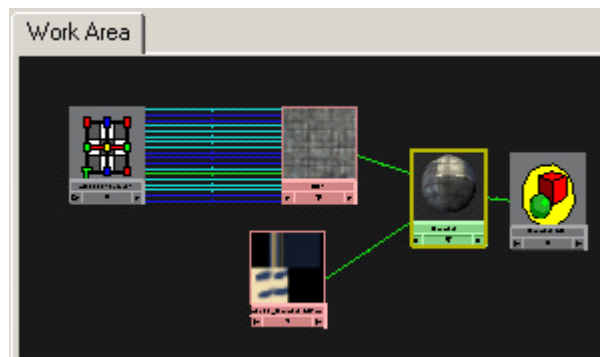


Set the following parameters and hit **Convert**.

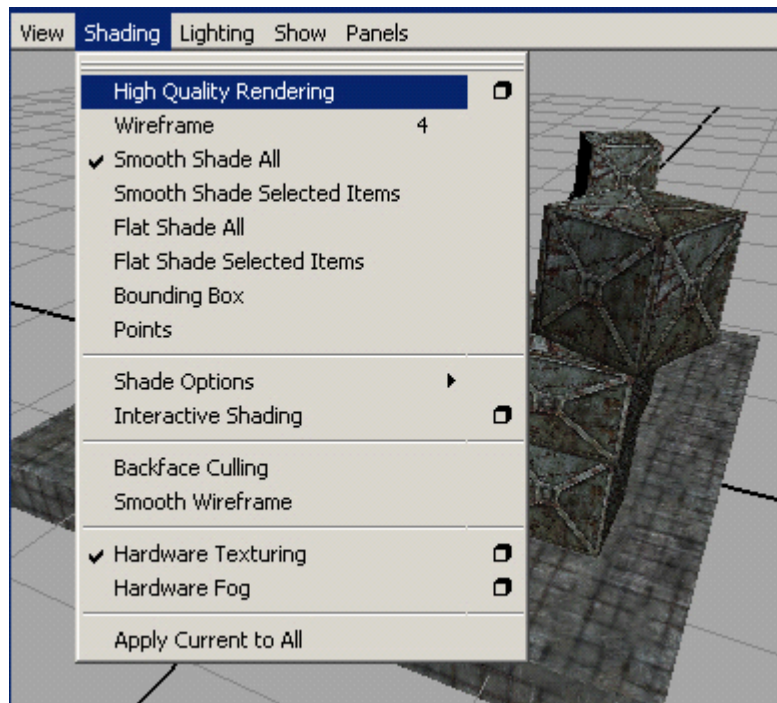


It takes some time for Mental ray to generate the bitmaps with lighting. Keep in mind that depending on lights, settings and geometry that you render lightmaps for it might take hours!

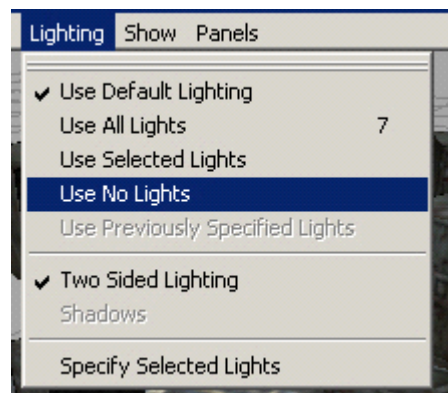
The rendered lightmaps can be found in the subdirectory of your Maya project “..\\mentalRay\\lightMap”. Create a new “file” node, load the rendered bitmap and link it to object material as “**ambient color**” attribute.



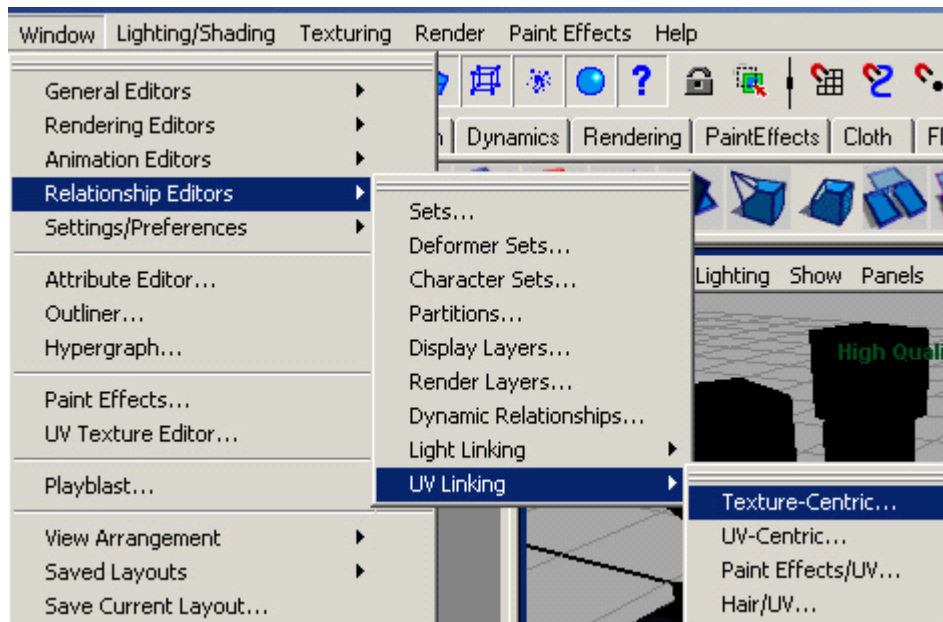
If you want to see lightmaps on your geometry in Maya set the following



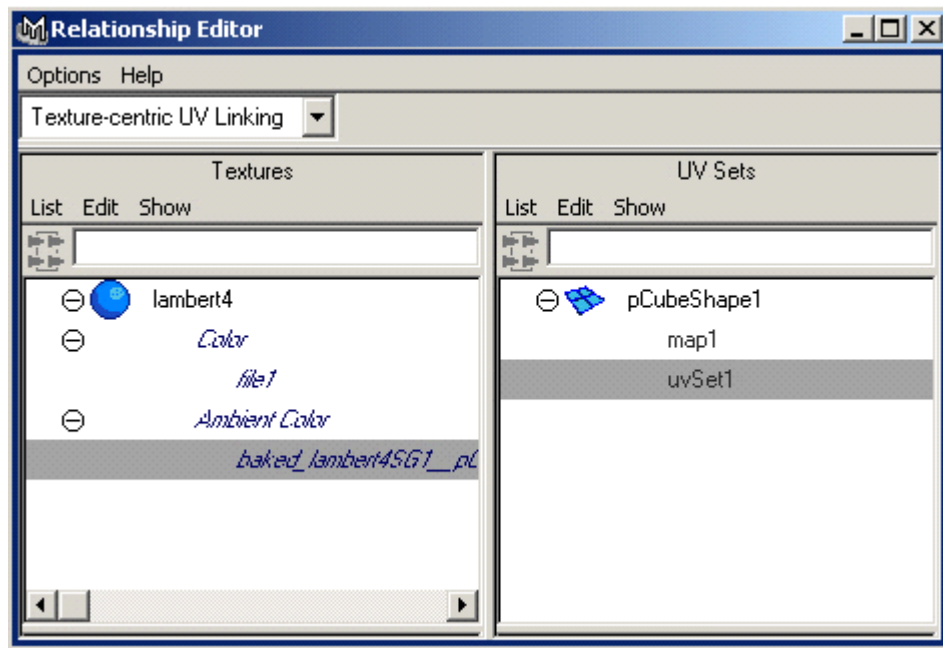
and



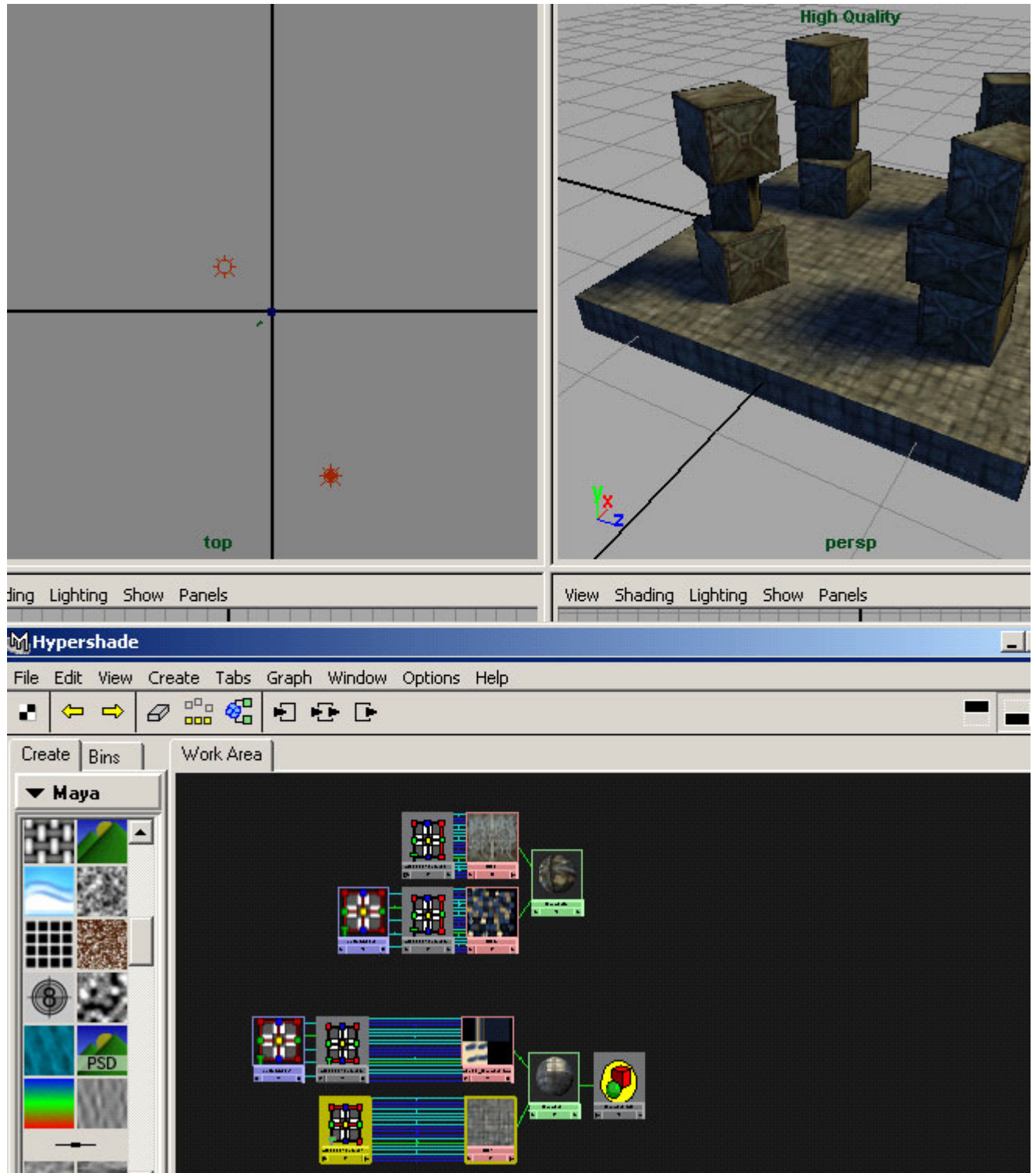
The result is strange, because we need to link the lightmaps to second UV set.



Select the object and in **Relationship Editor** click on the lightmap name in left window and on uvSet1 in the right one.



This is how the final scene and object materials look like:



The scene is ready to be exported to PainEd!