

Editing a Blender OSG File for: Adding a Lightmap, making a 2nd set of UV's, using a 32 bit Alpha, and making a multi texture mesh.

By Mr. D at Delta 3D 4-2007

Lightmap

This isn't that hard for a simple object where you can use the same texture UV's for both your diffuse and lightmap.

Simply it works like this you copy and paste within the Blender .OSG file for an object two sections, the first being your 'Texture Unit'.

```
}
textureUnit 0 {
  GL_TEXTURE_2D ON
  Texture2D {
    DataVariance STATIC
    file "C:\3d\Blender2.43\Blender\images\blt_sng.tga"
    wrap_s REPEAT
    wrap_t REPEAT
    wrap_r CLAMP
    min_filter LINEAR_MIPMAP_LINEAR
    mag_filter LINEAR
    maxAnisotropy 1
    internalFormatMode USE_IMAGE_DATA_FORMAT
  }
}
```

Ok, now '*textureUnit 0*' means this texture will override the **Diffuse** color settings and instead display the texture listed.

But a '*textureUnit 1*' will blend with *textureUnit 0*, which with a grey scale map acts as your **Lightmap**. So you just copy this section of code one under the other and **change 0 to 1**, and edit the *file line* to point to your Lightmap texture.



Should end up looking like this.

```
}
textureUnit 0 {
  GL_TEXTURE_2D ON
  Texture2D {
    DataVariance STATIC
    file "C:\3d\Blender2.43\Blender\images\bllt_sng.tga"
    wrap_s REPEAT
    wrap_t REPEAT
    wrap_r CLAMP
    min_filter LINEAR_MIPMAP_LINEAR
    mag_filter LINEAR
    maxAnisotropy 1
    internalFormatMode USE_IMAGE_DATA_FORMAT
  }
}
textureUnit 1 {
  GL_TEXTURE_2D ON
  Texture2D {
    DataVariance STATIC
    file "C:\3d\Blender2.43\Blender\images\bllt_Lightmap.tga"
    wrap_s REPEAT
    wrap_t REPEAT
    wrap_r CLAMP
    min_filter LINEAR_MIPMAP_LINEAR
    mag_filter LINEAR
    maxAnisotropy 1
    internalFormatMode USE_IMAGE_DATA_FORMAT
  }
}
```

Code Piece 2

The second piece of code you need to copy is the Texture Co-ordinate Array, which is located near the bottom of the file and will usually have a lot more numbers than this.

```
}
TexCoordArray 0 Vec2Array 96 {
  0.0 0.00191556173377
  0.268827676773 0.00109621684533
  0.278604596853 0.277011543512
  0.000824791379273 0.268991440535
  0.537655353546 0.000276895763818
  0.529582679272 0.276246547699
```


may notice a second texture unit needs its own set of UV co-ordinates. So if you switch to displaying a different UV setup it will be the one exported.

Now the problem: It's shared UV's. If I map a two faces as a quad it is considered to have 4 UV's as the vertices where they share an edge meet can be treated in the UV mapping as a just 2 and not 4 UV's.

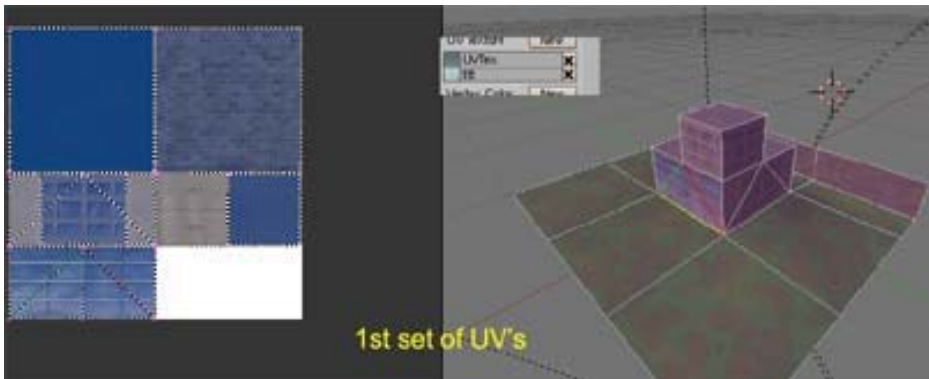
If you make a cube in Blender, cubic UV map and osg export you will find it has 24 UV co-ordinates; or 6 quads time 4 UV's. But if you count tri's it has 12 tri's time 3 vertices each means 36 UV co-ordinates.

So depending on how you map you can end up with two widely varied co-ordinate UV counts for the same object. How does the osg exporter for example in 3ds Max get around this? By not concerning itself with how you mapped the mesh, but by simply breaking it down into tri's on export for every UV co-ordinate output. So no matter how you map it always comes out the same count.

Fortunately you can do this in Blender, but unfortunately it is a bit complicated.

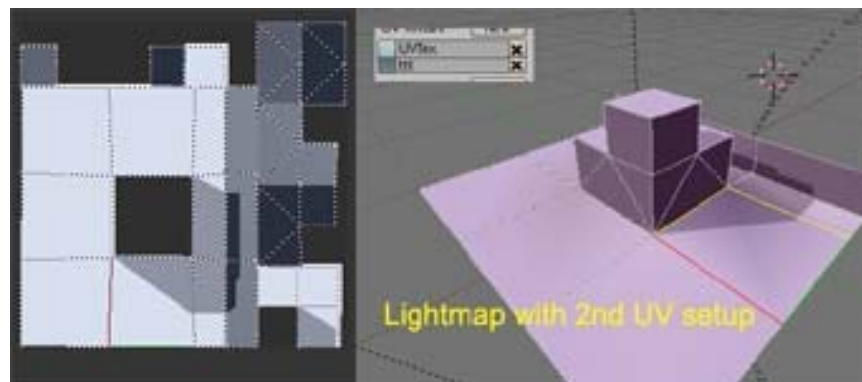


- 1) Same mesh but notice how the texture used differs from above, since my UV's for diffuse and Lightmap don't need to be the same; I was able to make larger textures mapped to faces where UV's overlapped.



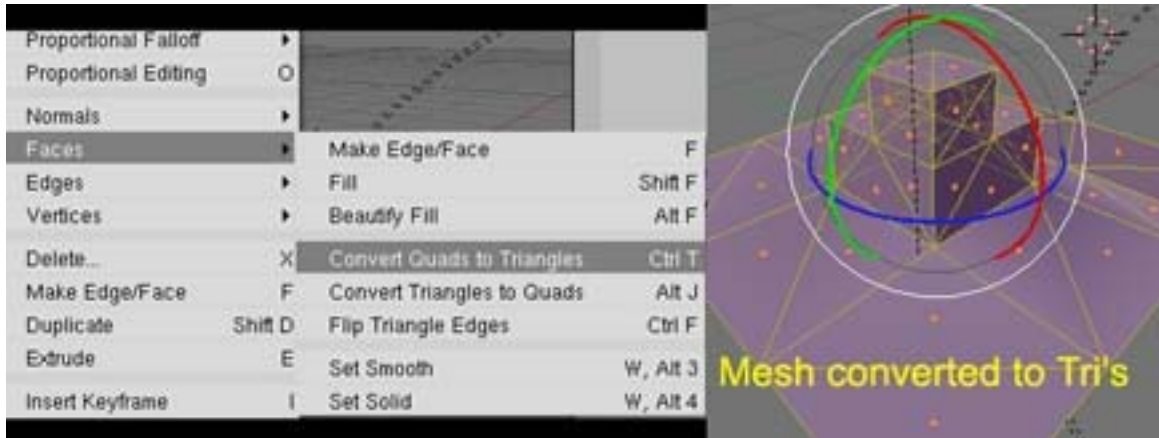
UV layout for this texture were all hand mapped to account for tiling.

UV layout for the lightmap texture, layout done using Unwrap – Smart Projection as for this lightmap I wanted no overlapping UV's.



STEP 1

First you'll need to convert your Quads over to Tri's, so in **EDIT MODE – Faces** select



all the mesh's faces. Then **EDIT> FACES> CONVERT QUADS TO TRIANGLES** remember you need counts to match so you want UV's output by 3 per Tri.



Now I had been told all I should need to do now is export out my UV's with 'Stick Local UV's to Mesh Vertex'. But I tried this several times and still got different UV counts for the two mappings.

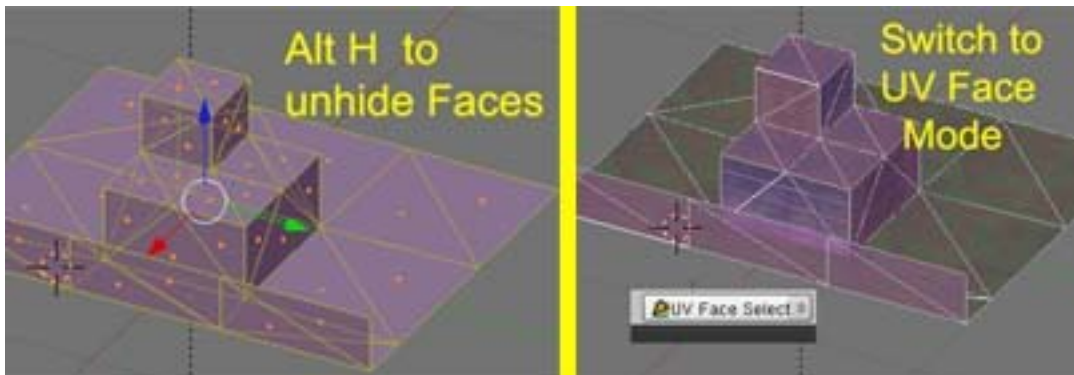
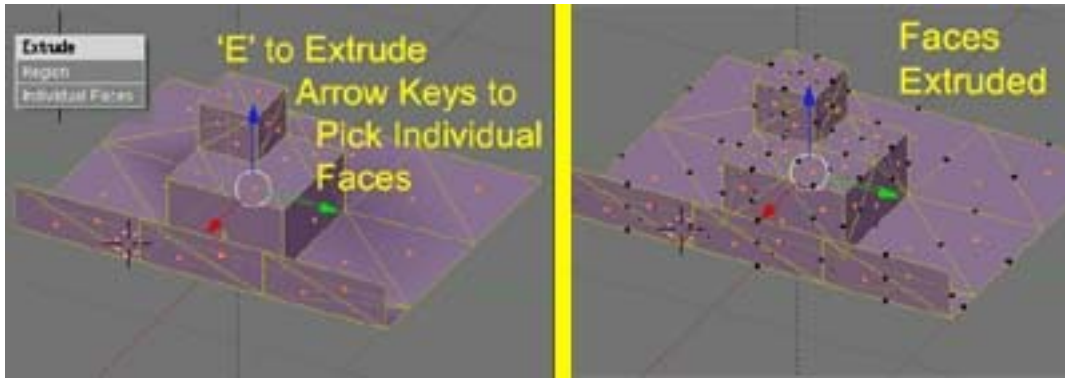
So this is how I devised to get even counts, and you will need to use your keyboard and key-short cuts, as using the mouse can cause small UV position changes if not careful.

In Edit Mode select all Faces and hit '**e**' for **extrude**, for the next part **use your keyboard** because even slightly moving your mouse will cause the faces to extrude out a bit; so when the extrude popup shows use the **Arrow Keys** to select '**Individual Faces**' then '**Enter**' twice (choose and confirm).

Now you want to hide these new faces so hit '**h**' to **hide**, then '**a**' to select all remaining faces, followed by '**Delete**' key using **Arrows to select 'Faces'** and '**Enter**'. Next to unhide your faces press **Alt 'H'**, and from here you can go back to your mouse.

Switch back to **UV Face Select Mode** and pick all your faces, recall you need a window open and setup for **UV/Image Editor** as this is where information is coming from. Now do an OSG export with one UV channel selected, change to the other channel and then export it.

VERY IMPORTANT! Remember to change the file name so you don't overwrite your 1st file.



Lastly check your two files to see if you've got the same number of Vertex Arrays.

```

TexCoordArray 0 Vec2Array 150 {
  0.752524375916 0.499283969402
  0.99880951643 0.252647697926
  0.998824059963 0.498647481203

```

```

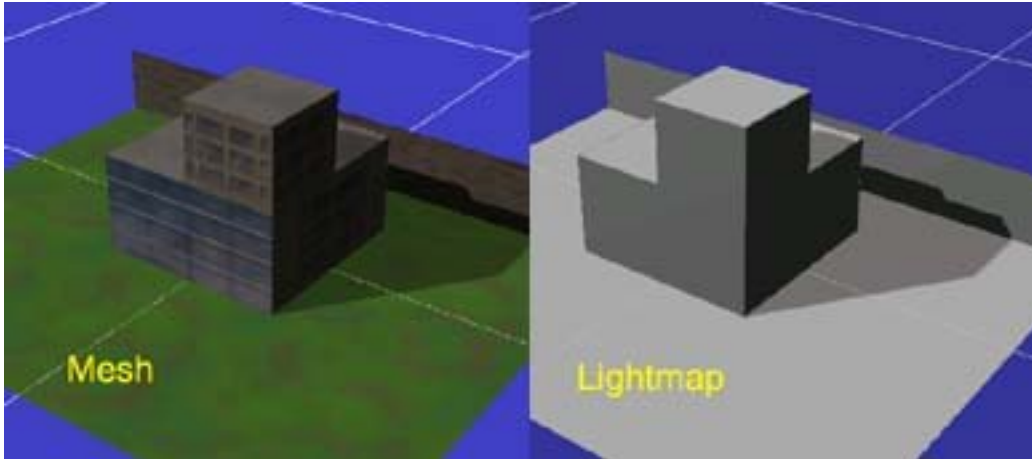
TexCoordArray 0 Vec2Array 150 {
  0.855641245842 0.000382498663384
  0.981515228748 0.124671287835
  0.856026172638 0.125053778291

```

These are my two, so you can tell things went OK here. I find with a little practice you should get them almost every time, also if different UV's you should see some number differences.

As to both being '*TexCoordArray 0*' as from above the second one will be for my Lightmap so I'll change it to a 1.

Finally from your Diffuse export do as above and copy and paste the Texture Unit 0 info, renumber to 1, and add your Lightmap's name and location. Then from your Lightmap Osg file copy and paste in the 'Tex CoordArray' section and renumber to 1. Check those brackets, save, and see if it works.



A Few Points:

A lightmap is only a Bitmap so you can edit it just like you would any other bitmap in a paint program. For say if you needed it brighter or darker, or wanted more contrast.

A lightmap is not lighting, but a bitmap that blends with the diffuse color, take a look at the beginning of this paper and notice how much brighter the lit building is in Blender. Because that is lighting and not blending.

Though I used a grey scale bitmap here, you can make it whatever color you want it to be, and you don't need to use one single light to make your map. If this was inside a building I could have setup a number of lights around the inside of rooms and rendered out a map to produce that type of effect.

Always load results in your engine to get a better idea of how it will look in your game.

For those who have not produce burned in lighting in Blender look for my paper on doing simple light and color rendering.

Some programmers can use a lightmap to calculate vertex lighting, or they can use the map to help develop crude character lighting. Comparing a lightmap's value in regards to the face location and using it to alter vertex light values on a character.

MORE FUN - GETTING A 32BIT TEXTURE TO SHOW UP IN THE ALPHA

Now since a 32 bit texture's Alpha will not show up in the Blender OsgEport, I'm borrowing some code from an .osg file I created in 3dsMax whose exporter does manage to do that.

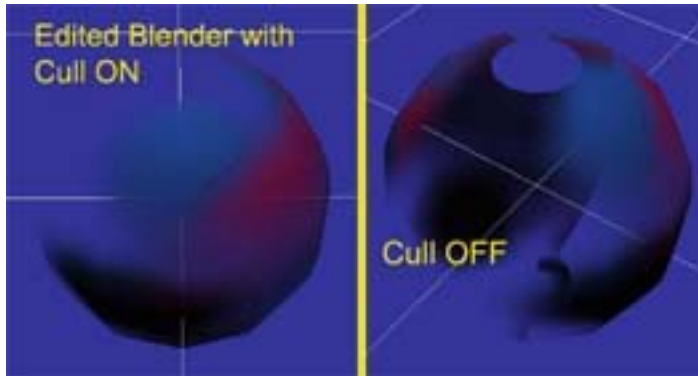
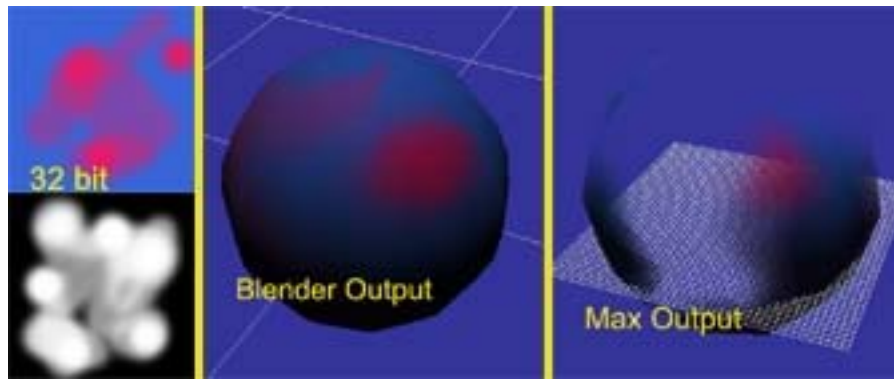
(3dsMax Code)

```
GL_CULL_FACE ON
GL_LIGHTING ON
0xba1 OVERRIDE/OFF
GL_BLEND ON
  Material {
    DataVariance STATIC
    ColorMode OFF
    ambientColor 0.588235 0.588235 0.588235 1
    diffuseColor 1 1 1 1
    specularColor 0 0 0 1
    emissionColor 0 0 0 1
    shininess 0
  }
  BlendFunc {
    DataVariance STATIC
    source SRC_ALPHA
    destination ONE_MINUS_SRC_ALPHA
  }
  textureUnit 0 {
    GL_TEXTURE_2D ON
```

(Blender Code)

```
GL_BLEND OFF ----- Make This 'ON'
  Material {
    DataVariance STATIC
    ColorMode OFF
    diffuseColor 0.800000011921 0.800000011921 0.800000011921 1.0
    specularColor 1.0 1.0 1.0 1
    emissionColor 0.0 0.0 0.0 1
    shininess 24.5960784314
  } ----- Take and paste in 'BlendFunc' section
  textureUnit 0 {
    GL_TEXTURE_2D ON
```

If you want your faces culled add the line *GL_CULL_FACE ON* above *GL_BLEND*.



Edited results for a 32 bit texture

BREAKING UP AN OBJECT TO MAKE ONE MULTI TEXTURED OBJECT

From my forum post: for those that missed it

Was comparing OSG files output between 3dsMax and Blender. Mainly looking at how Max can do a multi-material object while Blender can not.

Something interesting I found is that when I split off the faces I wanted a different texture on thus making a separate object; if I then created a material for that new part, in the end transforming my one object into several objects. Next I exported an .OSG file I could produce what seems to be a single multi-textured object by editing the file.

Make your own multi material OSg object from Blender multi object put

```

StateSet { Oxbal ON }
Matrix {
  DataVariance DYNAMIC
  1.0 0.0 0.0 0.0
  0.0 1.0 0.0 0.0
  0.0 0.0 1.0 0.0
  0.0 0.0 0.0 1.0
}
UniqueID "Cube.001"
DataVariance DYNAMIC
name "Cube.001"
cullingActive TRUE
num_children 1
Geode {
  UniqueID "Cube.001_geode"
  DataVariance DYNAMIC
  name "Cube.001_geode"
  cullingActive TRUE
  num_drawables 1
  Geometry {
    StateSet {
      UniqueID Cube.001_stateset
      DataVariance STATIC
      rendering_hint OPAQUE_BIN
      GL_BLEND OFF
      Material {
        DataVariance STATIC

```

```

1.0 0.0
0.0 0.0
0.0 -1.0
-1.0 -0.999999880791
-0.999999821186 2.38418579102e-007
}
}
MatrixTransform {
  StateSet { Oxbal ON }
  Matrix {
    DataVariance DYNAMIC
    1.0 0.0 0.0 0.0
    0.0 1.0 0.0 0.0
    0.0 0.0 1.0 0.0
    0.0 0.0 0.0 1.0
  }
  UniqueID "Cube"
  DataVariance DYNAMIC
  name "Cube"
  cullingActive TRUE
  num_children 1
}
Geode {
  UniqueID "Cube_geode"
  DataVariance DYNAMIC
  name "Cube_geode"
  cullingActive TRUE
  num_drawables 1
  Geometry {
    StateSet {
      UniqueID Cube_stateset
      DataVariance STATIC
      rendering_hint OPAQUE_BIN
      GL_BLEND OFF

```

from all objects after the first one remove these lines of code.
 should be one for each object output.

1st object in file
 set 'num_drawables #' to number of object-parts

First you must make sure you've got just the pieces that make up your object selected, each with a single material on it, then .OSG export selected from in Blender.

Now open the .OSG file created and look for the *"Geode" section* on the first object listed, and change the **number of Drawables equal to the number of parts** you have.

Then for all other objects **remove the lines shown, basically Maxtrix Transform through Geode section,** (Keep and eye on the brackets) and save the file back off.

I got it to show and act OK in both the Delta Viewer and Stage.

REMEMBER ALWAYS WORK FROM A BACKUP!