BF2 Statics Advanced Lightmap Tutorial

Introduction

This is a short tutorial on how to achieve nicer and more efficient lightmaps for BF2. We use a new feature in BfMeshView to generate the samples required to render lightmaps in BF2Editor. The most important thing to note is that you will have to do the lightmap UVs manually, rather than let the POE BF2 exporter generate them for you.

A big advantage is that you'll have more control over the lightmapping process, you will be able to make small changes (such as changing texture UVs or remove polygons, stitch seam and other small errors) without having to re-light the statics on your maps. Re-lighting can be a tedious process, especially if the static you changed is on a number of levels.

Another advantage is that you can pack the lightmap UVs tighter and cleaner than the POE StaticMesh exporter can (example). Typically, you can pack the lightmap UVs tight enough that you can halve the lightmap resolution, for many statics, which will reduce the lightmap texture load by 400% per static used in the level, which can bring down the overall texture load for a level considerably.

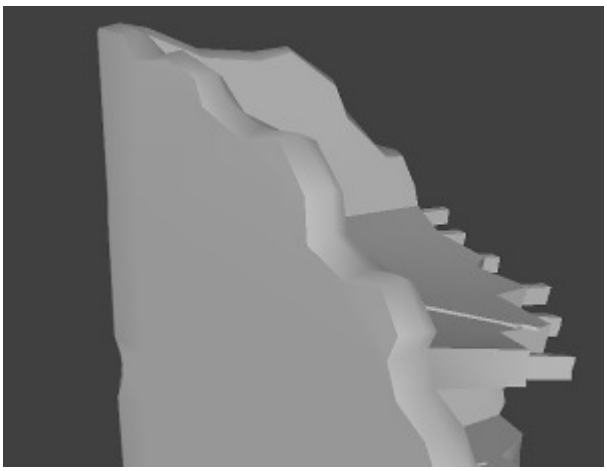
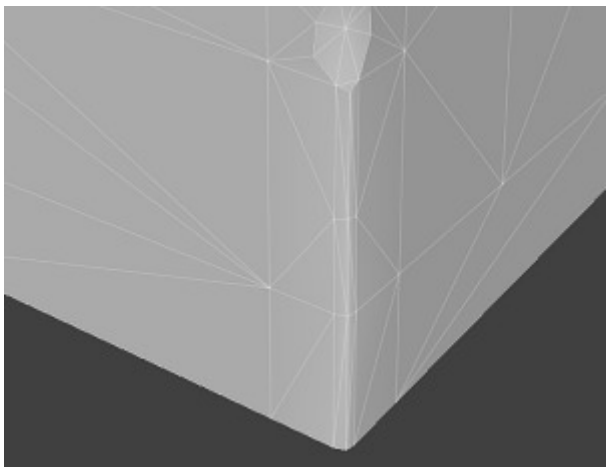
Unfortunately, this will not immediately work for existing statics exported with the POE generated lightmap UVs, changing the lightmap UVs implies re-lighting the static in the levels it's placed in (though you can always clone and rename an existing statics, and use it on future levels and leave the old one as it is).

Step 1 - Clean up your static

In this tutorial I will be using a static of a ruin like one you can find in Forgotten Hope 2. It has large flat surfaces, rounded corners, irregular broken stone and thin edges; ideal to highlight some of the problematic features on a mesh that can give problems with lightmaps.

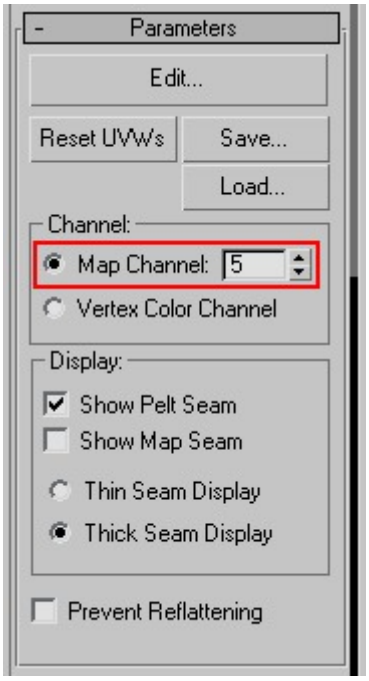
We start by inspecting our static and cleaning up the smooth groups. First off, make sure there are no thin degenerate (very thin) triangles or other issues that the POE static mesh exporter has problems with. The sample generator in BfMeshView can handle them without issue, but it is best to keep your mesh as clean as possible (tip: use the STL modifier in 3dsmax to pinpoint errors).

Getting the smoothing groups right is important because the samples will inherit smoothing errors. For this mesh I decided to smoothen the broken edges of the walls to keep the edges clean and orderly. The corner of the outer wall is rounded, with sufficient tessellation so that the large flat parts of the wall keep their flat appearance.

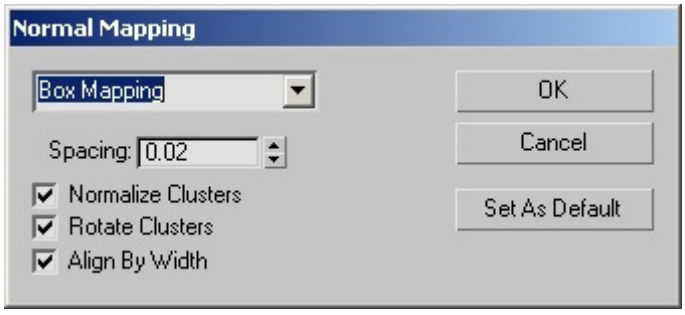



Step 2 - Automatic UV generation

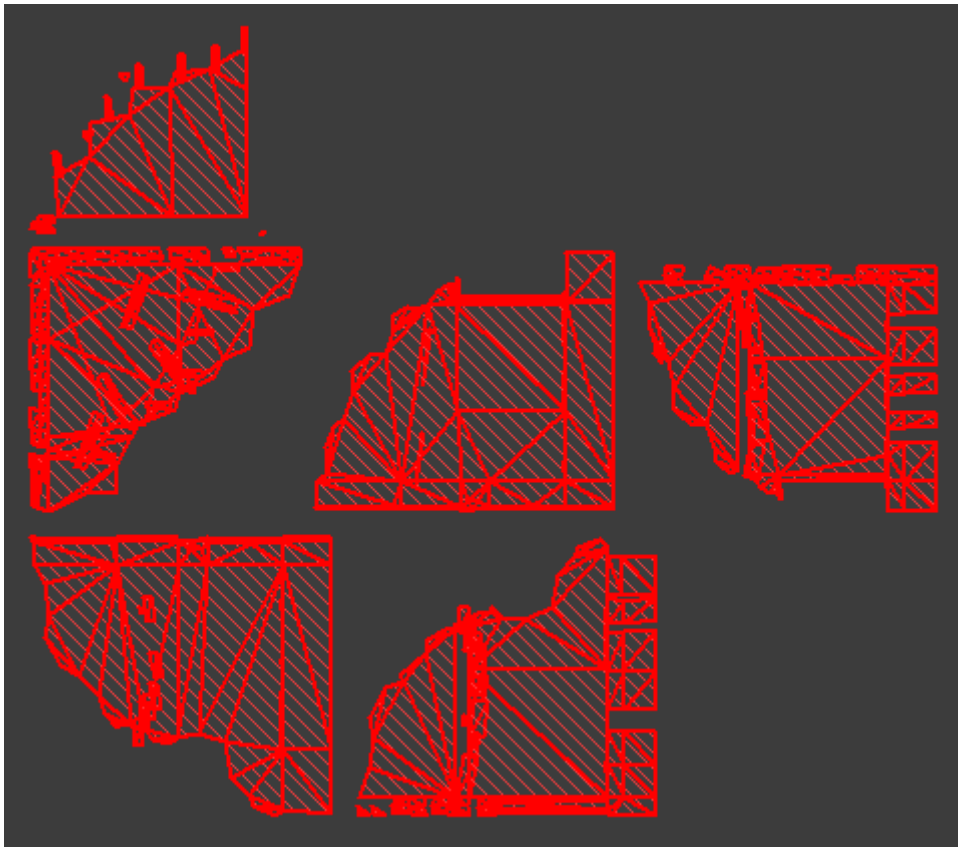
The lightmap UVs go in channel 5. We will automatically generate the UVs with the various UV Unwrap tools in 3dsmax, but you can also re-use the UVs from the base/detail channel, it can save you some time if you copy those and skip the next few steps, but I only recommend it if you know what you're doing.



We generate a quick mapping with UV Unwrap modifier, via the menu: Mapping > Normal Mapping, with these settings:

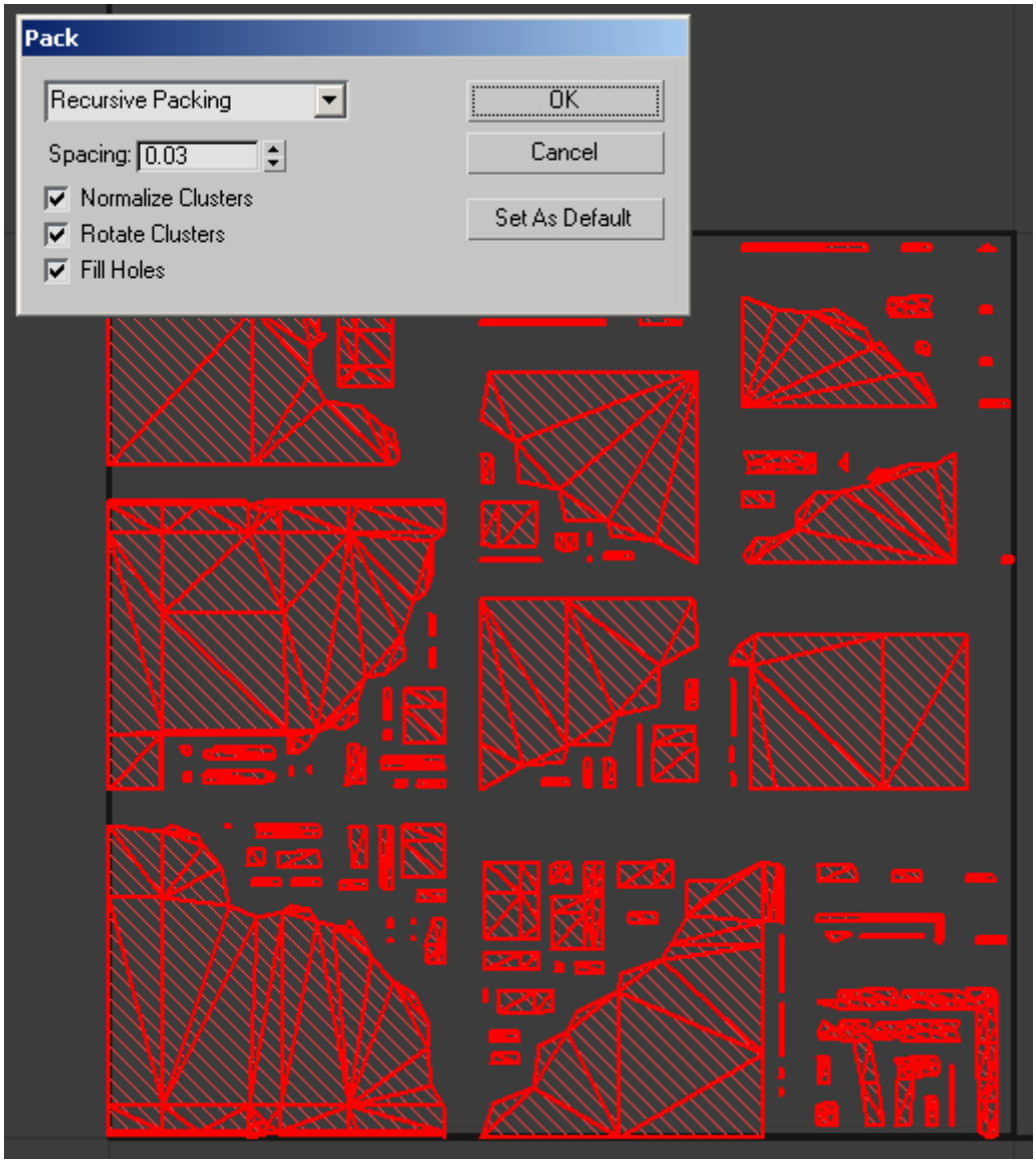


We use the box mapping, so all UV clusters on all sides of the static will be detached from each other. Which will give you something like this:



At first sight, not very useful as everything overlaps. What we do want however, is an uniform texel ratio. That means that the pixel mapping on the polygons will be (mostly) square, and the pixels have the same size on all polygons.

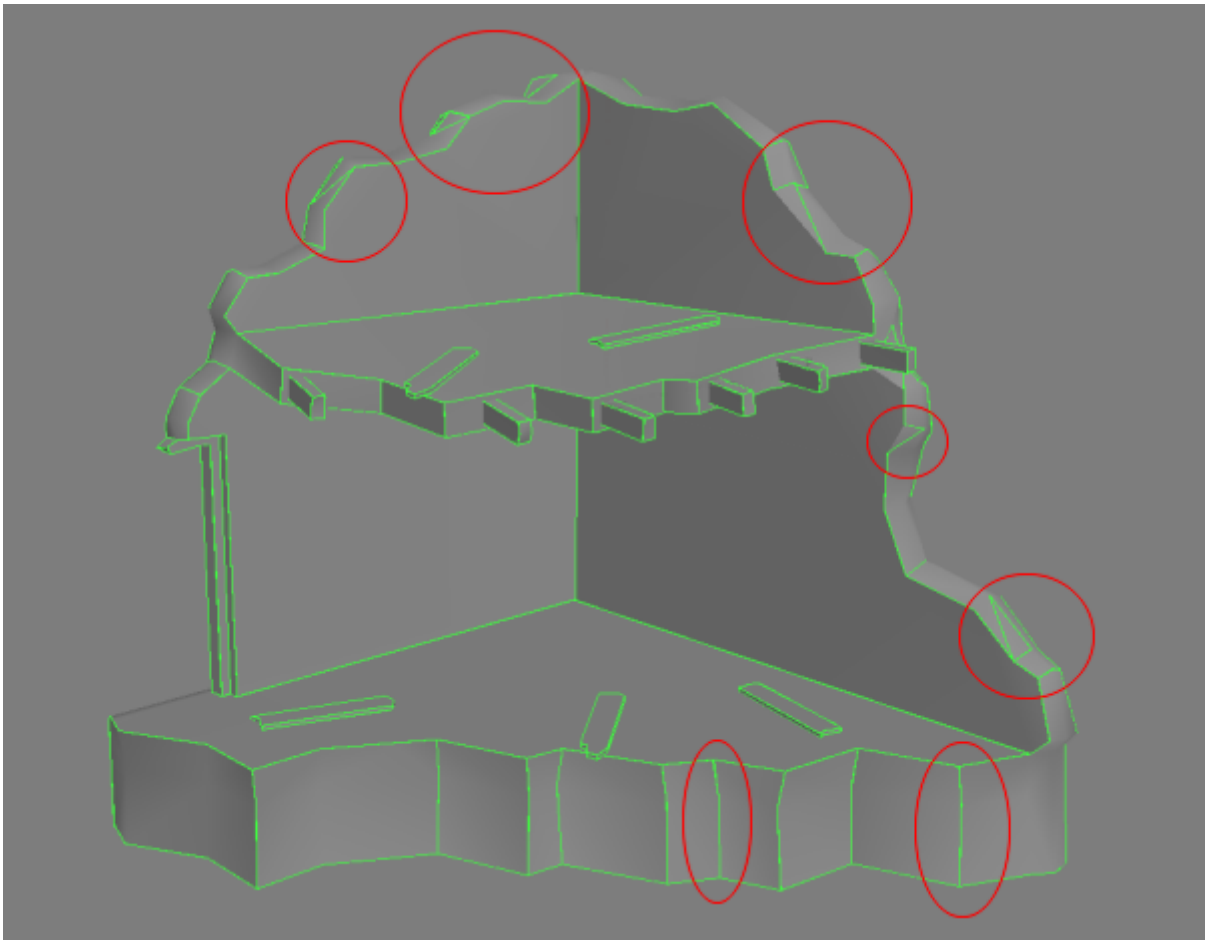
We then pack it, in the menu: Tools > Pack UVs, with the following settings (you may want to uncheck "Rotate Clusters" if you plan to edit the UVs after this, they may be easier to recognize when they are not rotated).



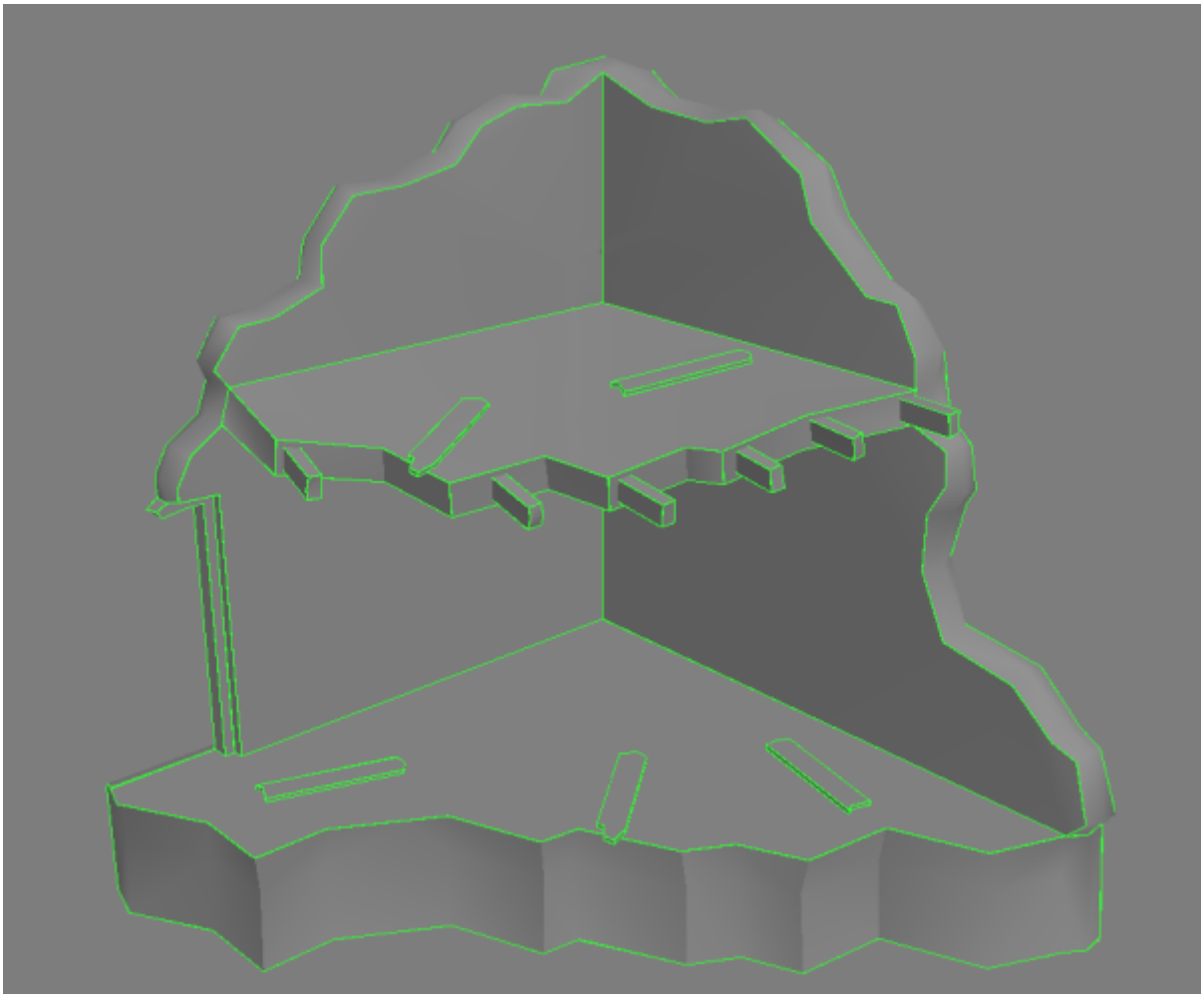
There, much better. Don't worry if it is not packed very tightly, we will first tweak some things and run the packer again later.

Step 3 - Tweaking the lightmap UVs

Unfortunately the automatic UV generation is rarely good enough, so we'll have to manually fix things. Often there are many UV seams in places where they are undesirable.

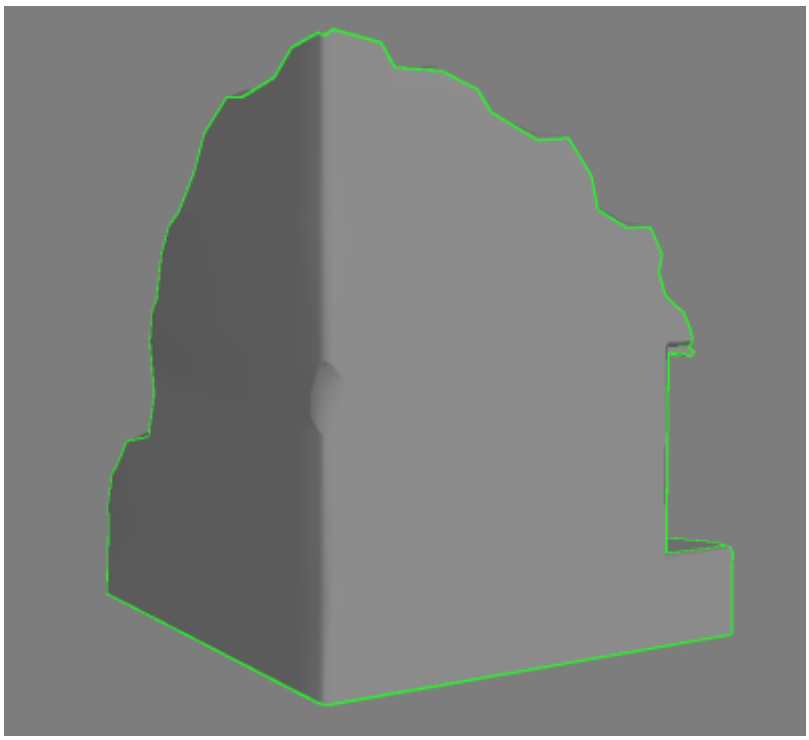


We'll have to fix that by hand. Generally the most ideal mapping is when the UV seams lie on the same edges as unsmoothened triangle edges.

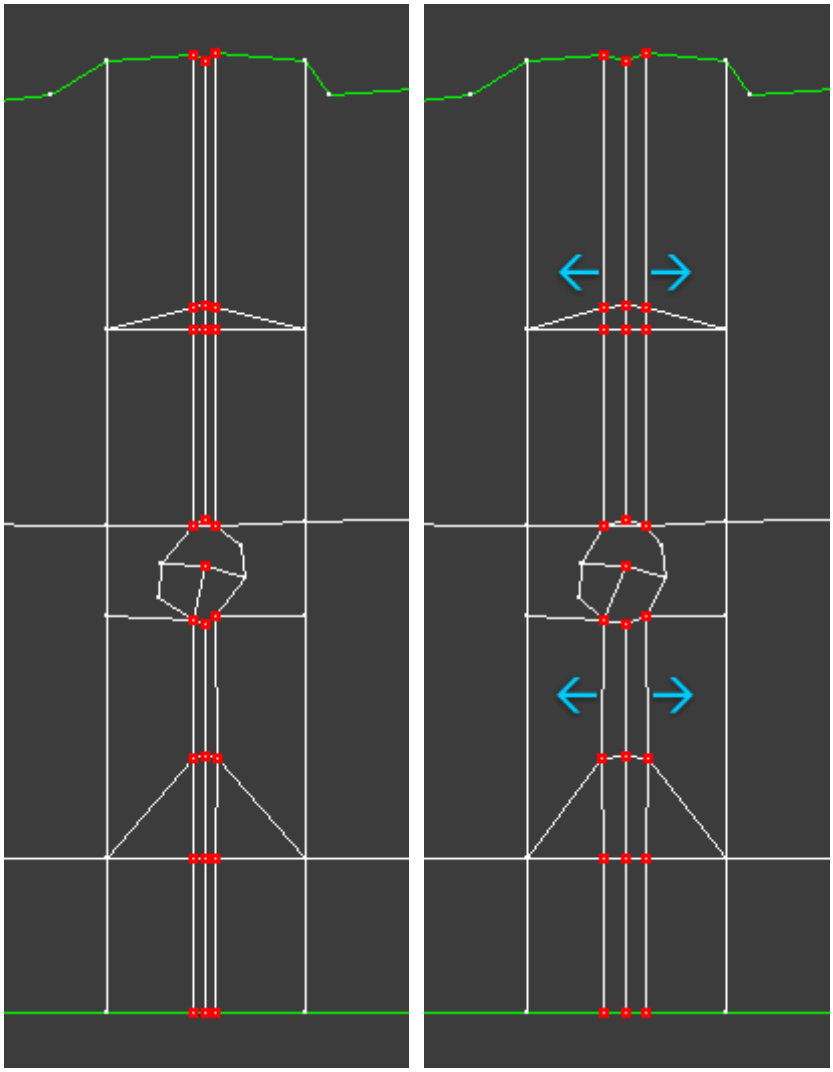


Much better. You can stitch up across unsmoothened edges if that helps to reduce UV fragmentation (as done here on the foundation of the ruin). There will still be sharp edges in BF2, but the lightmap can be cleaner. You probably don't want to do that often however.

Notice we also stitched the outer wall around the rounded corner. This ensures that there will not be an ugly seam in the lightmap.

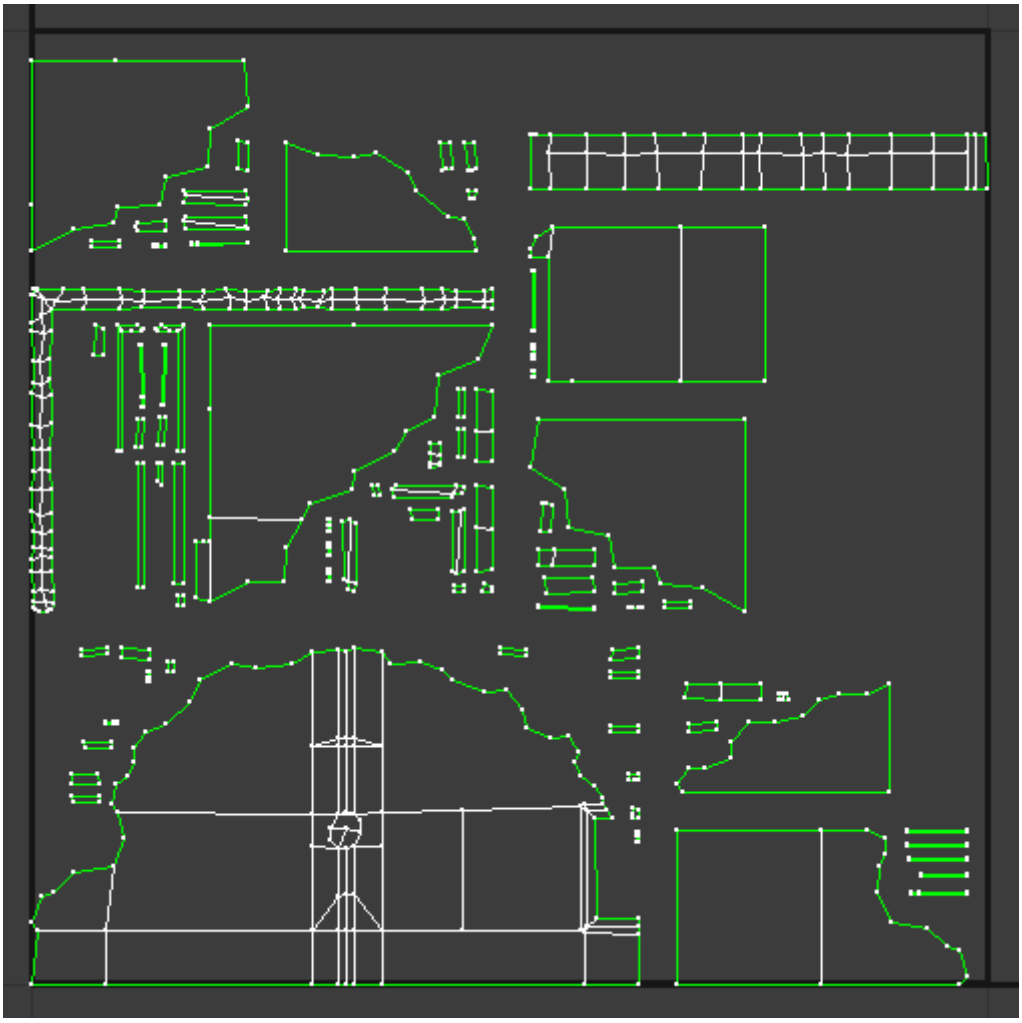


Another trick we apply: we scale the UVs on the rounded corner up a bit, so that more lightmap pixels will lie on the triangles, with the result that the corner remains smooth once lightmapped. The lightmap texels will be stretched here, but this is rarely noticeable.



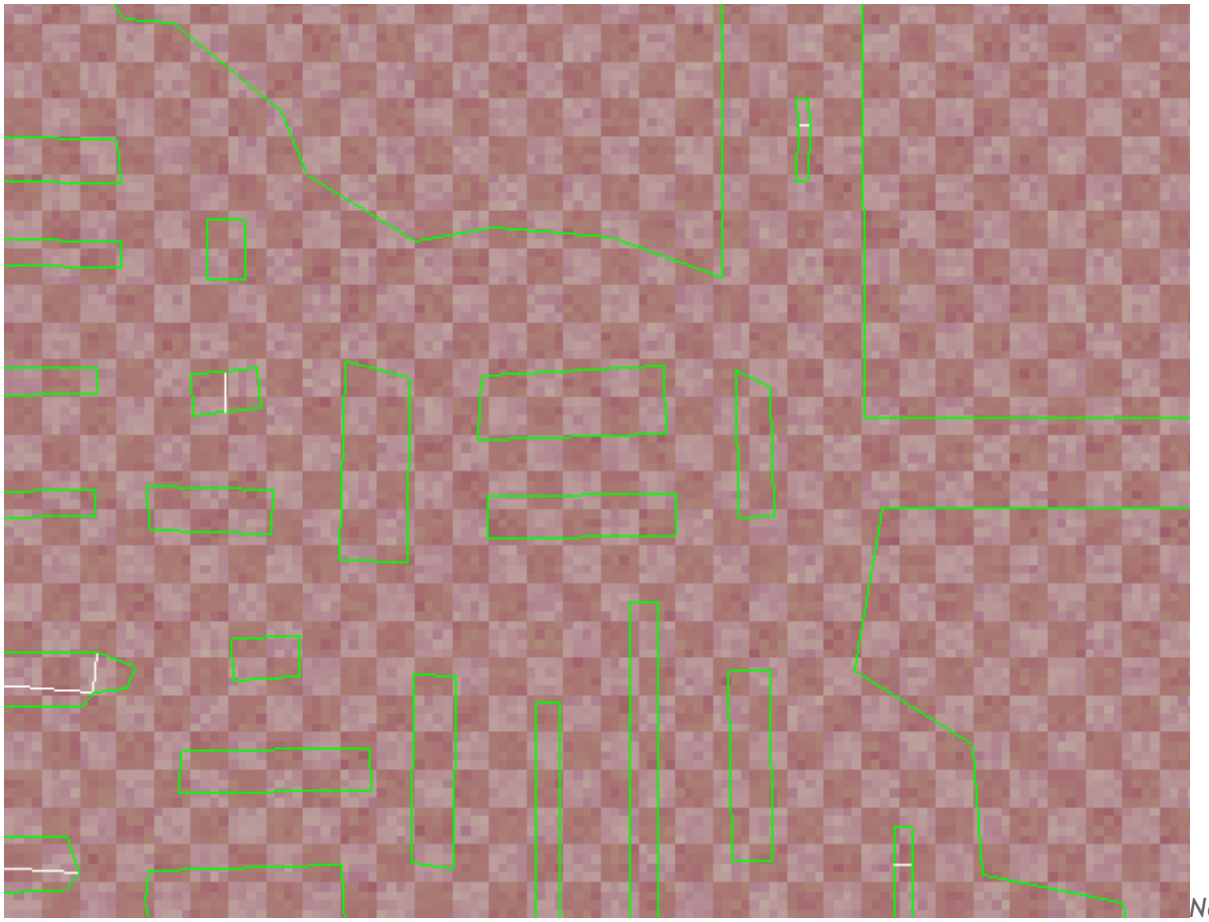
Step 4 - Final UV packing

With everything fixed up, we'll try to pack everything as tightly as possible, while keeping enough padding space between clusters to avoid lightmap pixels leaking onto neighboring clusters. We run the UV packer again.



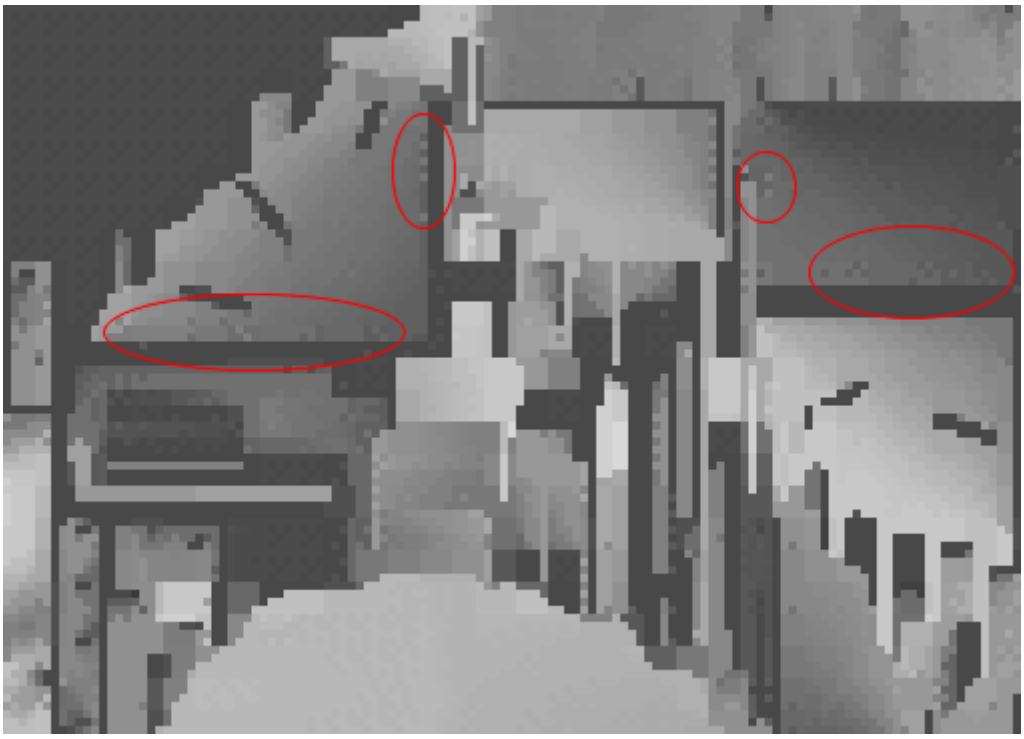
Not very good. We will improve it manually, and along the way, we scale up some of the small or thin clusters. Try to keep the triangles no thinner than the width of a row of pixels. This requires you to judge the final resolution of the lightmap. In this case I aimed for 128x128.
This is also important for padding. Put two UV clusters to close together, and a pixel will overlap both, and you will notice something we call 'light bleeding'. To much spacing, and you waste valuable pixels that could give the lightmap some extra detail.

To help with judging the scale of the lightmap pixels, generate a noise image at the desired lightmap resolution, and put it as background in UV Unwrap.



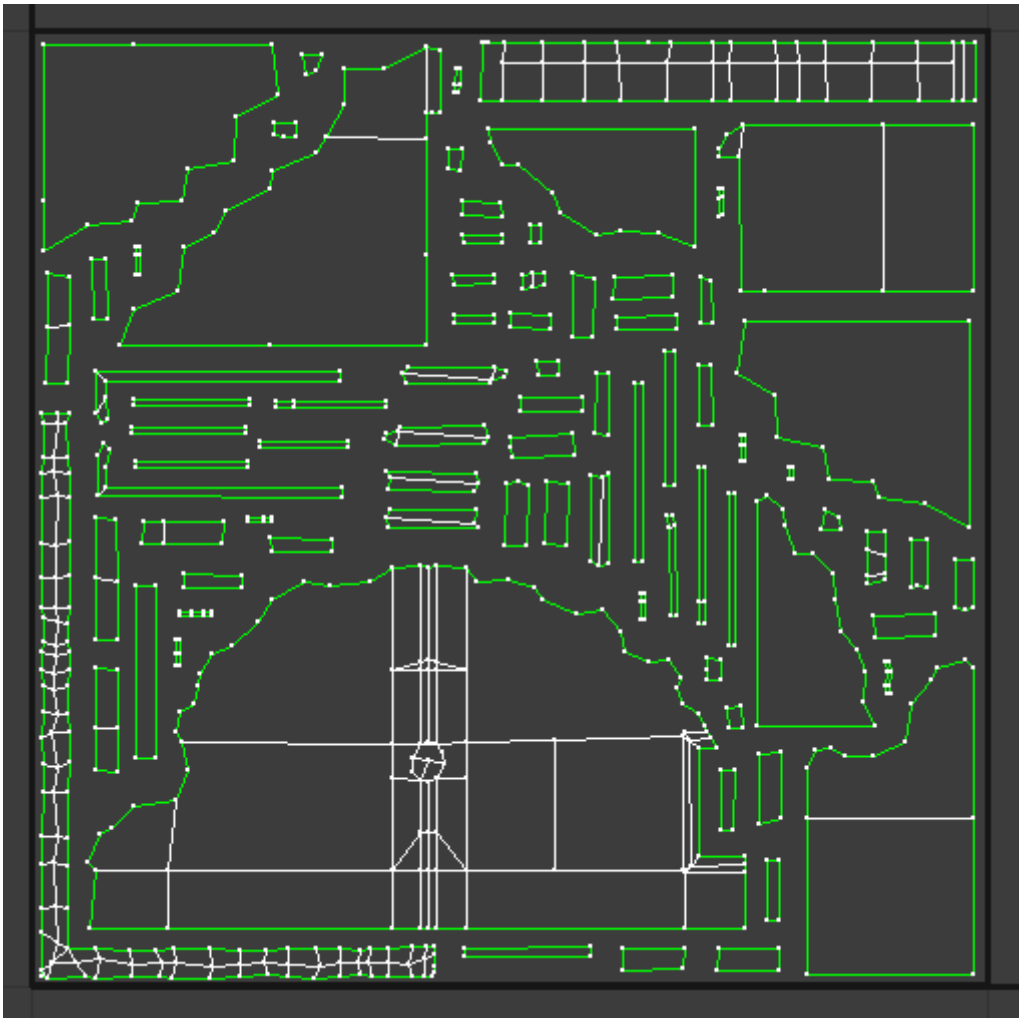
Noise-grid helper background image.

Keep in mind that lightmaps in BF2 are DXT compressed. DXT compression works on blocks of 4x4 pixels. When there is great contrast in the pixel colors within this 4x4 block, some color bleeding may be the result after compression. With that in mind, try to keep at least 4 to 8 pixels between UV clusters for best results. I added a 4x4 checker pattern to the noise texture (see above) to help with that.



Typical DXT compression artifacts on lightmaps.

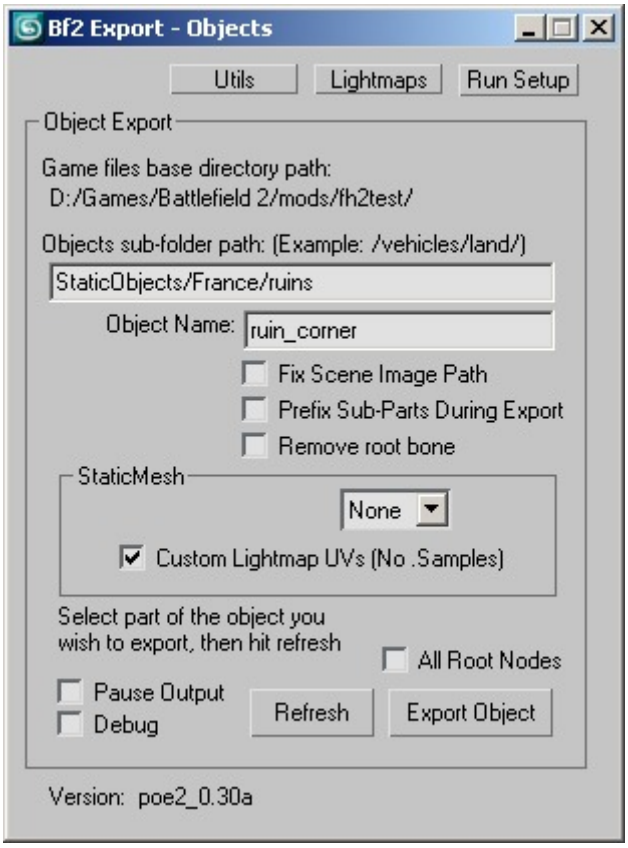
Here's our final, manually packed UVs:



Of course, it is absolutely crucial that there are no overlapping UVs. So before we call things done, we verify we didn't make any mistakes. In Unwrap UVW, click "Tools" > "Render UVW Template" and render a template, with the checkbox "Show Overlap" checked.

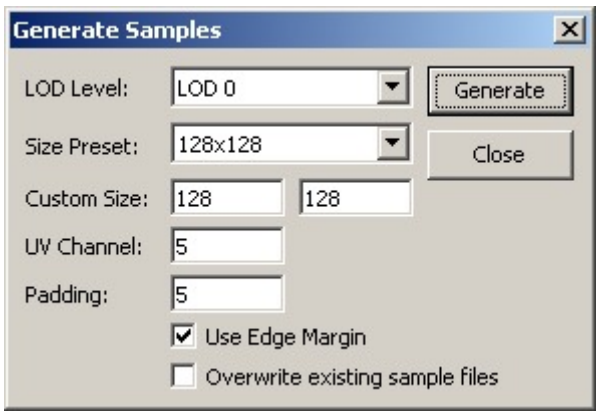
Step 5 - Exporting

All the hard work is done now, time to export the mesh. We use the POE exporter as usual, but check the "Custom Lightmap UVs" checkbox.



Step 6 - Generating samples

Next up, we run BfMeshView to generate the samples. Click in the menu "Tools" > "Generate Samples".



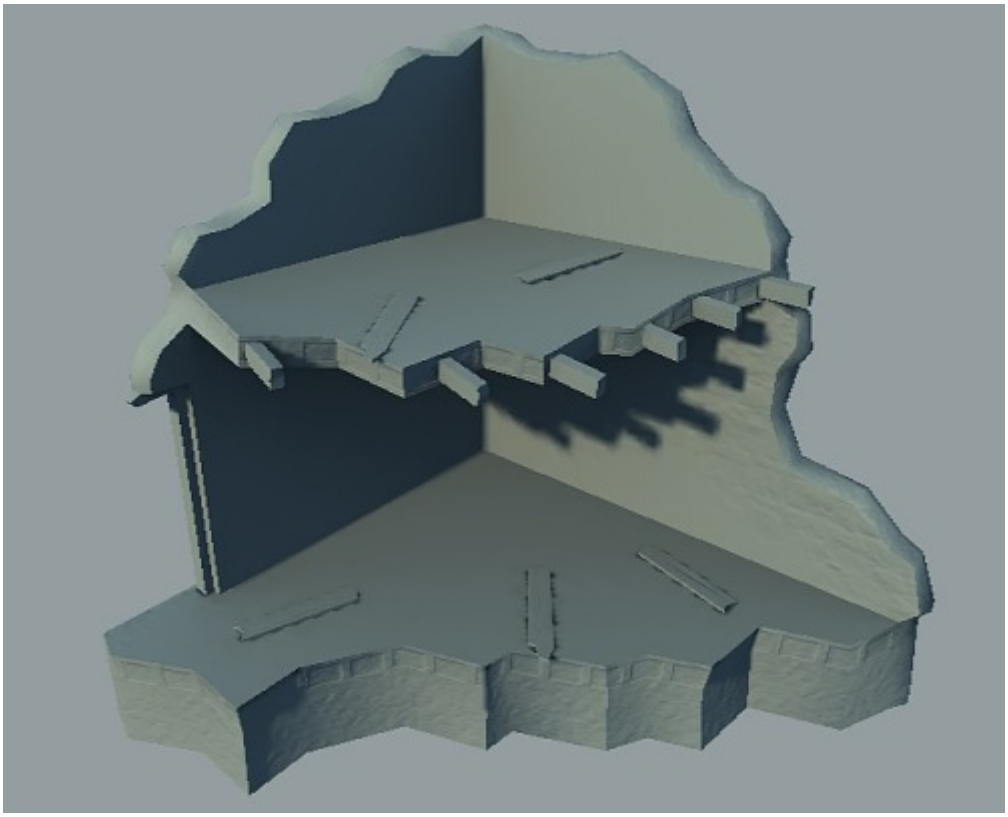
We select the resolution that we planned for earlier (128x128) and use the default padding, and select the "Use Edge Margin" option.

The "padding" parameter expands rasterized samples near UV seams outwards, so that neighboring un-rendered pixels pixels outside the triangle will have the same color as the pixel inside the triangle in the final lightmap. This is identical to the padding option on most texture baking tools, such as the one in 3dsmax. The default value is carefully tweaked, you'll rarely have to change it. A padding of 6 will rule out the chance that the unrendered black pixels will affect the pixel color due to DXT compression, because it spreads out the sample over the neighboring 4x4 pixels.

"Edge Margin" does something similar, but offsets the sample position outside the triangle. This often fixes light bleeding through walls, and avoids contrast near UV seams on rounded shapes shapes (typically a problem for cylindrical shapes where there's an UV seam where it wraps around).

Step 7 - Inspect lightmaps

Now run BF2Editor and put your static in a level, and render a lightmap at final resolution. Then switch to "Object Lighting Mode" via the "Render" menu.



Look for light bleeding on all sides and in all corners, especially small details. Rotate the static around and light it from various angles until you are satisfied there are no errors. Keep in mind that small lightmap errors are often not noticeable with full textured and normal mapped shading.

Tip: If you have disabled automatic reloading of textures, you can reload lightmaps by entering the command "texturemanager.reloadTextures lightmap". If you're smart, you add it to the "custom command" menu via the menu "Tool" > "Options" > Miscellaneous, so you can reload lightmaps with just one click.

If you want to increase or decrease the samples resolution, you can run BfMeshView alongside BF2Editor and re-generate the samples without restarting the editor, it just works. Keep in mind that you may need to adjust the lightmap UVs if you reduce the samples resolution, as the padding may need to be increased to avoid light bleeding.

Conclusion

Now of course, you also have to repeat this for LODs. Usually, you can copy the LOD0, and base the lower LODs from it, delete small details, collapse rounded/beveled edges like you normally do, and stitch up newly created seams in UV channel 5, when you're done. You can introduce seams for corners that are no longer beveled/rounded, and don't forget to correct the smoothing groups right as well. Overall, it is the same process, but you increase the UV padding space if the sample resolution is halved with each LOD.

Now stick your statics on a map, pick some nice light settings and make something cool of it!

Email: martijn AT bytehazard DOT com
Page created: 2010-01-28