How to make lightmap UVs

Creating UV maps for objects is tough. And doing it again for lightmaps is even more work. But it makes a big difference to visual quality. Unity's automatic lightmap UV generation doesn't work too well with complex objects, and for simple ones, it can still place seams in totally obvious and ugly places. The solution? Make them ourselves!

A lightmap UV must sit entirely within the 0-1 range, and no parts of it can overlap. If parts of the lightmap UV overlap, the baking result will be broken because the rays from the overlapping regions will fight with each other. If you see black splotches on your lightmap, it can be caused by broken lightmap UV. (Bakery will warn you about overlapping UVs in the log.)

Try and give each surface a consistent texel density. In other words, make sure each part of the object gets about the same amount of lightmapping space, such that if the model's texture was a checkerboard the squares would look a consistent size. Bakery will automatically smoothen out seams between your lightmaps where it can, but if the sides don't line up, it will end up looking a bit broke anyway.

With that in mind, let's go over some key points to keep in mind.

1. Don't overlap UVs.
As noted before. The lightmap can't be generated correctly if the UVs overlap.

2. Keep the texture coordinates within 0-1 space.
As noted before. The lightmap can't be generated correctly if the UVs are outside this range. The actual resolution on lightmapped objects is set inside Unity.

3. Don't pack individual UV islands together too tightly.
Shadows from other parts of the model will leak if they're too close. To avoid light bleeding, there must be a sufficient amount of space between each island.

4. Hide seams as much as you can.
Make sure seams are inconspicuous, because depending on the lightmap resolution they can be pretty obvious.

5. Don't place UVs diagonally when you could align them to an axis.
Diagonal UVs will stand out badly, and they're wasteful compared to axis-aligned rectangles. Avoid them as much as possible.

6. Connecting UV islands isn't always a good idea.
On the one hand, connecting islands together makes sense. It reduces seams and ugly gaps. But on the other hand, if the resolution for an area isn't high enough, the texels for one area of the mesh will bleed across to the other parts, leading to incorrect shading.

The disadvantage tends to be especially noticeable in buildings, since they are generally flat planes.

7. Put more texel resolution into corners where the angle of the surface changes.
If your model has bevelled edges, giving the bevelled regions extra space on the lightmap avoids both bleeding and seams.

Actually, for VR especially, it's a good idea to give the edges of hard-surface models a bevelled edge. This removes the obviously fake CG-look that super sharp edges have... but it also makes the lightmapping look better.

If you have hard edges on a model, splitting the lightmap there looks right, but most props and objects in the real world won't have a super hard edge. If you connect the edge there, though, the lightmap bleeds across it in an ugly way. So smooth that edge out with a bevel! The bevel will let the light fall across the model smoothly, adding in more perceived detail and a natural appearance.

Notes:
* http://tsumikiseisaku.com/blog/unity-japan-office-2/
This info was mainly based on this blog post, which is written in Japanese. Check out the original for some nice demonstrative images and insight into the modelling process for the Unity Japan Office project.

* https://docs.unity3d.com/Manual/LightingGiUvs-GeneratingLightmappingUVs.html
The Unity documentation on generating lightmap UVs, which goes into great detail about how the auto generation process can fail.

* https://docs.unity3d.com/Manual/ProgressiveLightmapper-UVOverlap.html
Unity's page on how to debug overlapping UVs, and the kinds of issues they cause.

* https://docs.unity3d.com/Manual/GIVis.html
Unity's page on the various visualisation modes that can display issues with your lightmap.

Revision #2
Created 26 January 2022 19:52:56 by Silent
Updated 26 January 2022 20:03:36 by Silent