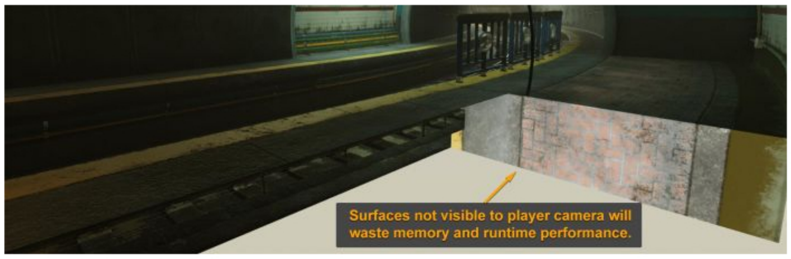
Modeling

Checked with version: 2017.3 - Difficulty: Intermediate

One of the mistakes that content creators usually make is not planning for what's ahead before modelling. It's ok to do fast and loose modelling during pre production or for roughing out a space, but the moment there's a need for the asset to be somewhat finalized, there are a few things that need to be thought out ahead of time. Here are few things to keep in mind when modeling a proper Scene that might be overlooked even by seasoned content creators.

- **Make every polygon count.** Much like everything else in project development, simplicity is your friend. Despite modern hardware being more capable than ever, simple geometry can go a long way in creating a Scene. Unnecessary tessellation and complex geometry can backfire as it will be difficult to manage for realtime setup, cost performance and can burn memory unnecessarily.


Surfaces not visible to player camera will waste memory and runtime performance.

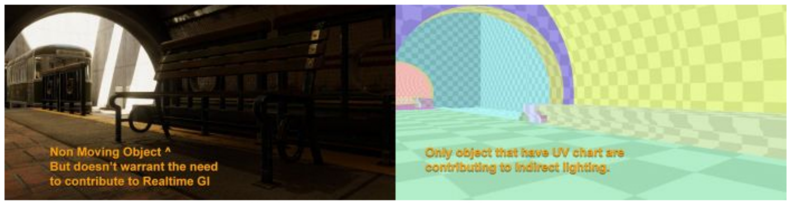
  In the example above geometry that is never seen by the players will only ended up wasting resources such as lightmap, overdraw and at worst it can cause light leakage.

- **Modeling rules based on Lighting mode of a geometry.** If you are using baked lighting or realtime Global Illumination with Light Probes you need to decide whether a model contribute or only receive the indirect/baked lighting in the Scene.

  – For objects that you want to contribute to lighting
  (Lightmap Static checked in the inspector) Simpler and smoother surface area produce better indirect bounces/baked lighting because of its efficiency of space usage in lightmaps textures. UV2 for the geometry might need to be authored when doing a light bake if the auto lightmap UV provides inefficient chart or generate undesirable seams. UV3 for the geometry might need to be authored for efficient results in realtime GI. Sometimes in Realtime GI, a UV of a mesh can be simplified to make the geometry uses significantly less resources and produce the best result with fewer artifacts.

  – For objects that only receive lighting from dynamic lights and Light Probes, the geometry don't have lightmap UV restriction. The geometry still needs special attention if its large as it might not be lit properly with a single probe and might require a light probe proxy volume to stitch multiple probes light definition. Check Unity manual for LPPV guide for further information.

Just because an object is not moving, doesn't mean it has to be lit by a lightmap or contribute to Realtime GI. It is very easy to fall in the habit of marking all static objects Static Lightmap. If an object is small, or it doesn't have surfaces that will bounce much light, it probably doesn't need to be included in the lightmap. The bench and railings below are a good example:


Non Moving Object ^
But doesn't warrant the need
to contribute to Realtime GI
Only object that have UV chart are contributing to indirect lighting.

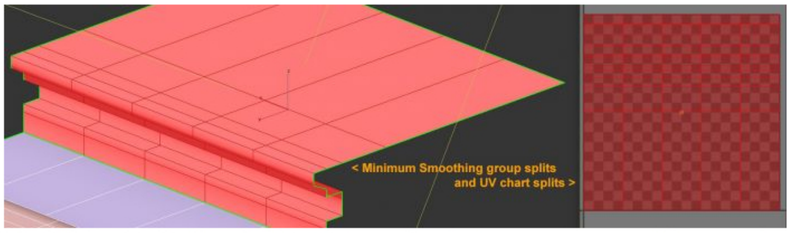
- **Model UV layout strategy.** UV layout can help improve visual quality while paying for the same memory footprints for normal map baking (typically UV1), lightmaps baking (UV2) and real-time lightmaps (UV3), especially for geometry with non-tileable textures.

Here are few tips for UV layout:

- For UV1 charts, split only as necessary and try to layout the UV chart as efficient as possible to not waste texture

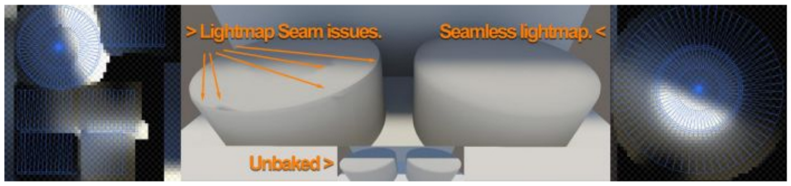
Creating Believable Visuals

- For UV1 charts, split only as necessary and try to layout the UV chart as efficient as possible to not waste texture space for normal map baking. To put it into perspective, a 1024 square texture will use the same amount of memory whether you place details or not in the texture.

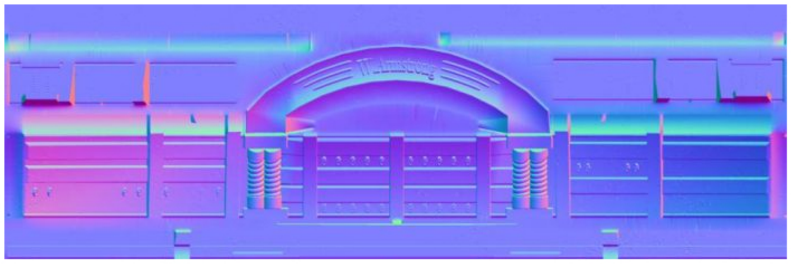


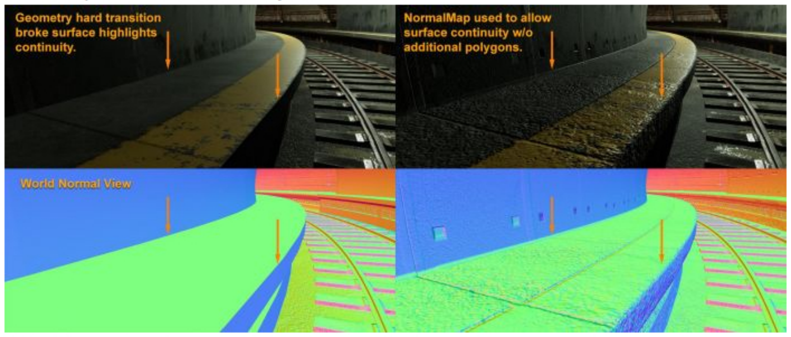
Above is an example how the pieces occupy the whole texture space avoiding wasted space.

- For lightmaps (UV2), try to make the charts a contiguous unbroken chart to avoid seams in the light baker. Lightmap UV charts should not overlap to avoid bleeding. Keeping a consistent scale between UV charts/shells is important for even distribution of lightmap texels across your model.

  Above is an exaggerated lightmaps splitting on a simple geometry that showcase issues with lightmap seams.
- For realtime GI (UV3), prioritize UV space for large areas that represent big surfaces in your model as best as possible to reduce memory usage and avoid seams. In many cases, the automatic UV settings in the model can go a long way in optimizing the chart. In depth chart optimization for realtime GI can be found here.
- For objects that doesn't require lightmaps, don't waste memory and time by authoring those additional UVs unless required by custom shaders.
- **Details in geometry.** Real world objects are highly detailed. Identifying details to be placed in geometry vs the normal map and textures is a standard part of real-time geometry authoring. High polygon to Low polygon Normal map baking is the norm these days in developing assets for real-time Scenes.

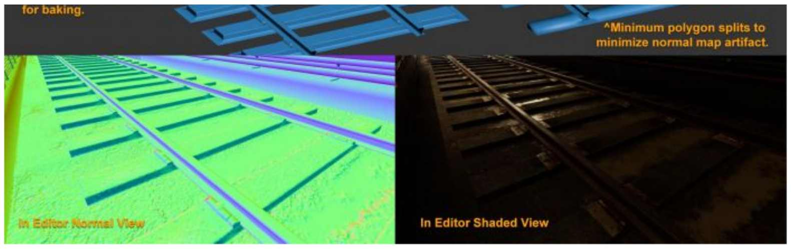
  

One important details that often gets missed is the way edges of an object catch highlights. It is unusual to find a real life object with very sharp edges, with non bevelled edges, or without details in edge definition. Replicating this effect improves the believability of the Scene.

Geometry hard transition broke surface highlights continuity.
NormalMap used to allow surface continuity w/o additional polygons.
World Normal View
This gives a pleasing smooth flowing highlight on surfaces where it meets other geometry.

- **Smoothing groups** (Hard/Soft edges of polygon). Efficiency of models and normal maps can be improved with proper smoothing groups.

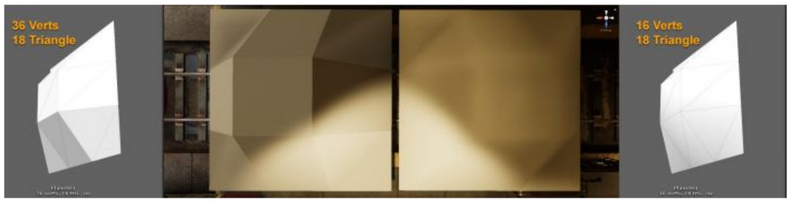
  – When dealing with Normal Map baking from high polygon to low polygon, a simple configuration for smoothing groups is preferred over multiple faceted polygons. This is due to tangent normal maps needing to bend the hard split of the surface normal from the low poly geometry.

  < Low polygon mesh
used in the scene.
High polygon mesh >



The above example showcase how the normal map does a great job in bending the low poly normal smoothly to mimic the high polygon mesh.

– A smooth polygon with a good normal map will also save on vertex count, which equates to more efficient geometry to render. Here's a simple example, a simple 18 triangle plane in 1 smoothing groups equals 16 vertices, compared to a single plane with split smoothing groups that is equals to 36 vertices.



– A smooth polygon will also save on chart splitting in Lightmap baking and Realtime GI that will, in turn, give a smoother visual result.

While this list doesn't cover the complete process of modeling a 3D prop or Scene, hopefully it gives content creators a good idea what to look out for.