Types of Textures

There's a popular saying among programmers; "garbage goes in, garbage comes out"! If you don't know what you're putting into the shader, you're only going to get garbage out of it. So, what are the different types of textures that Standard supports, and what do they do? Let's find out!

- Albedo/Diffuse
- Normal Maps
- Specular and Metallic Maps
- Occlusion, Height, and Emission Maps
- Detail Mask and Details Maps

Albedo/Diffuse

First, we have the albedo map. The albedo map is uninteresting, but very important. It's where the colours of your material come from! What you'd put here is the same kind of texture that other engines would call a colour or diffuse map. To go into extra detail: "Albedo (al-bee-doh) is a measure of how much light that hits a surface is reflected without being absorbed." Note the "without being absorbed" part. That'll be important later.

After that, we have the Color. Wait, this isn't a texture! But here's something important and often overlooked - when you export a model from Blender, Blender typically changes the material colour to *grey*. When Unity loads the model, it sets the Color in the material to grey too, *making the material duller*! So, whenever you load in a new model to Unity, you should make sure the material colour is white and fully opaque.

Normal Maps

Next, we have the normal map. The normal map defines details on the surface of the material seperate to those of the mesh.

If you're reading this guide, you might not know what a normal is. Let me explain it for you! You see, meshes are made up of vertices, points in space, which make up polygons. And a polygon in a mesh is more than just three points in space. Stored throughout it is information regarding which direction the polygon is facing - the normal. This is what decides the lighting of meshes. Bad normals mean weird seams and warping that don't match our expectations. Good normals mean things look soft and round in all the right places, and hard and sharp elsewhere!

Think about meshes themselves. You can put a texture on a flat simple plane and cover it in colours, but it'll still be a simple plane underneath. So, someone had the great idea to use a texture to shift the normals around, covering it in details. A texture can be more abstractly detailed than a mesh, right? And that's what a normal map is.

In Unity, these are stored in a special format once imported, which is why you need to mark new normal map textures imported *as* normal map textures. Unity will normally show a warning if a texture is assigned as a normal map without being imported as a normal map.

If you have a normal map from another game engine, it might be oriented differently. You can tell because the ridges on one axis will look like bumps, and vice versa. In particular, normal maps made for Unreal Engine will always be inverted. In the current version of Unity, you can fix this once the texture is set to a normal map by checking the "Flip G channel" option.

If you're looking at assets without normal maps, there's no need to panic, because a normal map isn't necessary if you don't have one. It only adds lighting detail. You can generate them yourself if you want, see below.

- More details on normal map orientation, and a handy table of which applications support and export what by default.

Specular and Metallic Maps

Specular maps are a tricky subject. Many write ups explaining specular maps focus on how a specific game or engine handles them, and the specific compromises and simplifications inherent to them. One thing is clear, though, specular maps focus on *shiny* stuff.

In Unity, that shininess is defined by two components, following the principles of PBR. There is the part that determines which sections are light absorbing, like metal, instead of light reflecting, like paper. That's stored in the red channels of the metallic map. And then there's the part that determines which sections have a smooth and reflective surface, like polished wood, instead of a rough surface like rusty metal. That's stored in the transparency of the metallic map. These attributes are called Metallic and Smoothness.

In proper terminology, non metallic surfaces like cloth are called dielectrics. The light that hits their surface bounces around inside and produces an even colour; albedo light. Metal, on the other hand, conducts electricity, and so reflects the photons that hit it and absorbs what doesn't reflect; producing specular light.

So that covers metallic maps. But wait, what about specular maps? Well, it turns out that it's a lot easier to have *one* map that has both the albedo and specular colours, and a second simpler map to define which parts are metal and which are not, instead of having two full colour maps that can take up more memory. However, a metallic map can't represent materials like cloth or skin well, which are neither metallic nor hard. So just consider a specular map as a way of defining a specific kind of shine, while a metalness map only defines that something is or isn't metal. This is the difference between Standard and Standard (Specular).

In a specular system, the metallic portions of the material are defined in the RGB channels of the specular map. The Alpha is still reserved for the smoothness. This takes more memory, but it's useful when a simple mask doesn't cover what you need. A good real world example are surfaces with shiny, thin metal lines. At angles, or far away, they'll lose their shine with a metallic map due to texture filtering. But with a specular map, they'll remain properly shiny.

However, if you add a metallic/specular map to an object, you'll notice the areas with specular become darker. This is because of *energy conservation*. One material can not be both fully dielectric and specular at the same time - only one, or the other, or something in between. So if dealing with specular maps made for another engine, they may need editing to fit.

To review:

- **Metallic** maps store an on/off value specifying whether part of the material is metallic or not, in the map's **red** channel. The material's albedo is used to specify how strong the specular reflectance is.

- **Specular** maps store a colour value representing the specular reflectance directly, in the **colour (RGB)** channels of the map.
- **Smoothness**, or glossiness, or the inverse of the surface's roughness is stored in the **alpha** channel, in both metallic and specular mode.

References:

- https://google.github.io/filament/Filament.md.html Google's guide to using PBR with their Filament engine. Written by their engineers, it covers the topic in a deeper way.
- https://academy.allegorithmic.com/courses/b6377358ad36c444f45e2deaa0626e65
- https://marmoset.co/posts/pbr-texture-conversion/
- https://www.sharetextures.com/textures/blog/physically-based_rendering/
- https://google.github.io/filament/Material Properties.pdf Google's reference chart to PBR materials. When making your textures, this is a useful thing to have on hand.

Later on, I'll explain how you can actually use these features.

Occlusion, Height, and Emission Maps

Occlusion Maps

Next up are occlusion maps. These maps contain ambient shadows - the cracks and crevices that are normally too small for light to reach unless it's shone directly into them.

For single models like props and characters, you can bake these using Blender, but the results depend on your model's scale and UV mapping. For repeating textures like brick walls, or floor tiles, or even just some mud, you can generate these from a normal or height map.

In Standard, the green channel is used for occlusion maps.

Height Maps

Height maps are used in Unity to provide a "parallax occlusion" effect. They shift the texture according to the height map texture and your perspective. Unity uses a fairly cheap implementation of this effect that can fall apart from the wrong angles, so be careful and don't turn it up too high.

In Standard, the red channel is used for height maps.

Emission Maps

Emission maps are for areas of a model that emit light, or glow. If you were making a material for a lamp, the light would be emissive. But an emission map has to be specially made - in particular, the unlit parts must be black! Setting up emission wrong can result in your models looking really weird. Emission maps are multiplied by the emission colour of the material. **Tip:** For a glow-in-the-dark effect, use an emission strength of 1.0. For a bright glow, use an emission strength of 5.0 or so.

Detail Mask and Details Maps

Detail Mask

Next up is the detail mask, which controls how the detail map in the next section is applied. Where the detail mask is black, the detail map won't be applied. This lets you have a detail map that affects some parts of your mesh, but not others.

In Standard, the blue channel is used for the detail mask.

Detail Maps

Detail map is a texture Unity combines with the base albedo. This is handy, because you can specify a different tiling factor for it, letting you overlay a pattern over a less detailed texture to make it look more detailed. This seems like a good idea in practise, but actually using it well is pretty hard.

By default, detail maps in Standard are applied by multiplying the albedo by the detail map and doubling the result. In simpler terms, a middle-grey detail map will have no effect, a white one will brighten the albedo, and a black one will darken it.

Detail normal maps work like detail maps, but for the normals. One example is to apply a pattern over a wall to make it look rougher at a different scale from the rest of the material to give it more variance. Or to add depth to a feature added through the detail map!

An important option to note for detail maps is the ability to set their UV channel. Just as meshes have UVs to tell them where the textures go, Standard supports using a second UV map specifically to put details on the mesh. This allows you to, for example, have a piece of clothing with a main texture consisting of just cloth, and then a detail texture with just a logo. Using the logo as a detail texture, and with the secondary UV channel mapped to the logo texture, you can combine the two textures and have a crisp logo over your model's material.

For more detail on detail maps, check the Unity documentation.