Section 1 - Introduction

Welcome to the Alloy Shader Framework Documentation! In this doc you will find a quick introduction to most relevant concepts of Physically-based Shading, and the Alloy Framework. Following this in section 2, there is a breakdown of the Alloy shader families and their variants. Finally, in section 3 you will find a detailed list of all of the parameters you will find in the various shader families, what they mean, with some tips for usage.

Alloy has been a passion project, nay an obsession for myself, and the exceptionally talented and dedicated Josh Ols, who two years ago approached me with the crazy idea of dragging Unity kicking and screaming into the world of Physically-based shading. Since then we here at RUST have made several games, a slew of prototypes, and the DX11 Contest winning Museum of the Microstar, all using various permutations of Alloy. We now gladly bring the fruit of our labors to the wider Unity community, and can't wait to see what incredible creations you all produce using Alloy.

-Anton Hand, Beard Warrior at RUST LTD.

Section 1.A - Core Concepts of Use

Energy Conservation: Alloy shaders use energy conserving equations. What this means in practice is that the total light output (excluding vfx elements like incandescence and added rim lighting) will never exceed the amount of incoming light energy from ambient and direct sources. There is no separate 'specular intensity' map or slider present here, which might take some getting used to. What this ensures is that regardless of your settings, the resulting material is likely to be physically-plausible, and will behave consistently in a range of lighting environments.

Metals: In the real world, metals interact with light in a very different manner than other materials. Internally, they have an albedo that is completely Black, and instead have an f0 (reflective constant), that is very high. Nonmetal materials in the Alloy model instead have a constant f0 of 0.04, which is a value that others implementing PBS have found to be totally workable.

Instead of having the user swap black into the relevant texture channel based on a material being metal/nonmetal, Alloy shaders have a 'Metalness' parameter, both in global slider and per-pixel form. This slider basically feeds a Lerped value of your Base Color, and Black into the appropriate parts of the PBS equations, allowing you to easily make a material behave as a metal, or nonmetal object. Values between '0' and '1' of metalness are incredibly uncommon for real-world materials, but we've left the option open to you to do, in the case that you need some sort of weird exotic sci-fi material characteristics.

RSRMs: Alloy shaders use a Look-up-texture called a Radially Symmetric Reflection map, which is an in-house solution we've put together for providing generalized reflectance information, for when generating and managing cubemaps is impractical. This map has pre-computed values for our full range of specular powers in it, and creates the appearance of a generic-horizon-line style range of luminance. What this ensures, is that light-mapped Alloy materials that are not being struck by _any_ real-time lighting will still appear reflective, and metal will still look like metals, without looking like mirrors.

Our cube-mapped variants currently Lerp between the output of this map, and whatever cubemap you have plugged into them based upon material smoothness, ensuring that only materials with maximum smoothness exhibit a perfect, bright cubemap reflection.

Ambience: For Alloy shaders to exhibit their full range of visual character (such as grazing-angle Fresnel), it is critical that all objects have some source of ambient lighting. For that reason, scenes using lightmapped objects should also have a corresponding set of Light-probes for all dynamic objects to reference. You could theoretically just use the constant ambience in the Render Settings, but we really don't recommend it.

Lightmapping: While Alloy shaders are fully compatible with all three forms of Lightmapping found in Unity, we heartily recommend that you use Directional lightmaps for best results. If you do not do so, you will find that many matte materials feel fairly flat. Dual lightmaps may be cheaper at runtime, but for about the same storage cost, directional lightmaps yield a dramatically higher visual quality.

Section 1.B Performance Considerations

Due to the BRDF that we've chosen for Alloy, the added performance cost of using this set should only be minimally above using the standard Unity shaders in principle. Where the cost lies instead is in the features and data-set sizes that you will find yourself using alongside Alloy:

- Deferred Rendering: which you pay a fixed cost for, which I of course feel is worth it 100% of the time for the ability to use dynamic lights _everywhere_. Plus being able to use spot-light shadows is a must.
- Per-pixel Operations: Normal mapping, per-pixel spec, rim lighting, etc. aren't free of course. While Alloy contains plenty of exotic variations, you should use these will discretion as with any complex shader.
- Texture Data: Between directional lightmaps, and fat texture sets, you're likely going to find yourself using tons of texture data (our Museum of the Microstar project was 80% texture data on disc, but we used 4096s on everything). Don't expect games using Alloy to be small downloads
- Translucent and Grab-Pass Variants: Be aware that these shaders don't benefit from a lot of engine optimizations, and can get heavy FAST.

Section 1.C Project Setup

Before using this shader set, there are a couple things you _must_ set up in your project. If you do not set up these options, you will get visual artifacts caused by broken math.

1. Open Edit->Project Settings->Player

2. Open the 'Other Settings' rollout

3. Set 'Color Space' to 'Linear'

4. If you want to use lots of dynamic lights, set 'Rendering Path' to 'Deferred Lighting'

5. Select your camera in your scene

6. Check the 'HDR' box

7. Now save a scene, and CLOSE UNITY COMPLETELY **This is necessary for the custom deferred shader to be loaded to override Unity's default deferred shader. This is what allows us to use a BRDF in deferred mode.**

8. Open Unity and your project back up.

Section 2 - Shaders

Alloy contains several shader families, each containing a set of variants for RSRM and Cube-mapped reflectance. This section of the documentation is designed to give you a conceptual overview of the framework's content, and where and when it is best to use various shaders in the family.

Section 2.A - Shader Families

Core: The shaders found in this family are the work-horses of the Alloy set. You will find the simplest shaders (the RSRM and Cube shaders) here, which you will likely use for a majority of your objects. You will find variants with glowy rim lighting, detail mapping, and incandescence in this family.

Nature: This category contains our custom Terrain shader, which supports up to 4 splats, applied in the usual way. The alpha channel of each Base color texture map should contain a Smoothness map in it if you need per-pixel variation. In the material itself, you will also find per-splat Metalness, Smoothness and Tinting parameters.

Particles: The shaders found here aren't Physically-based like the rest of the Alloy set, but are instead a couple useful HDR-compliant particle shaders that we've found ourselves using commonly alongside Alloy. These have intensity values that can go up to '8' for soooper glowy particles :-)

Self-Illum: The shaders found in this family have one core purpose. Namely they are configured specifically to allow for colored emission from the surface to be properly sent to Beast. If you do not need this functionality for a given object, the shaders in the Core family are generally a better choice due to their flexibility.

Transparent: The shaders in this family follow the combinatorial logic of the Core family, except that they are all translucent. Note that this means that they are Forward-only, and will not receive shadows in deferred mode. They will also have a limit of the number of pixel lights that can affect them at once, defined by your quality settings. The Alpha-cutout shaders can also be found in this set, which you can also use for shadow-casting particles!

Section 2.B - Individual Shader List

Core:

Core/Rsrm & Core/Cube

> The work-horse. The most stripped down Physical shader in the Alloy set. We recommend using this on all your common environment objects, simple props, etc. This will scale the best, and has everything you need for common realistic objects.

Core/Rsrm Detail & Core/Cube Detail

> The basic detail-mapped shader of the set. These use an extra Color, Occlusion and Normalmap texture, which can be mapped independently of the Base textures. This is useful for large surface areas and objects that you want to add extra microtexture to. Great for rocks, bark and other naturalist scene objects.

Core/Rsrm Detail Rim & Core/Cube Detail Rim

Just like the detail-mapped shader, but with ability to the add glowy rim lighting.

Core/Rsrm MaskedIncandescence Rim & Core/Cube MaskedIncandescence Rim

This shader allows you to add masked incandescence and rim lighting to your material. The Incandescence map has its own set of texture transforms, while the Incandescence mask stays locked to your base map. Super useful for having flowing energy moving through the crevices of a complex material, having burning objects glow from the inside, and other fun vfx tricks.

Core/Rsrm Rim & Core/Cube Rim

Just like the standard Rsrm and Cube shaders, but with the ability to add glowy rim lighting.

Nature:

Nature/Terrain/Rsrm & Nature/Terrain/Cube

While using Alloy, you must use these shaders if you have terrain in your scene. Two important notes on their use however are that you are limited to 4 splats, and you MUST set the terrain basemap distance to maximum when using them. The basemap LOD splatting auto-combine in Unity's terrain system breaks with Alloy. We're still trying to find a work-around.

Particles:

Particles/Alpha Intensity

This is a standard Additive particle shader, but with the added feature of being able to raise its intensity above 1.0. Super critical for getting the most out of your particles when using HDR

Particles/Cullback Alpha Intensity

Just like the shader above, but culls the backface from being drawn. For some special case I can't remember.

Particles/Independent Alpha Intensity

This is a particle shader we put together specifically for doing texture-scrolling/sprite sheet animation on Trail renderers. The alpha channel has a separate set of texture transforms from the Color map, allowing you to move them independently via script for vfx purposes. Also has an intensity parameter that goes above 1.0 for HDR.

Self-Illum:

Self-Illum/Rsrm & Self-Illum/Cube

The standard set of Alloy shader for use with baked static environment objects that you wish to have send their emissive information into the Beast lightmapped. If you don't need this functionality, use the core set instead.

Self-Illum/Rsrm Rim & Self-Illum/Cube Rim

Just like the shader above, but with the ability to add glowy rim lighting.

Self-Illum/Hdr

A purely emissive color shader with HDR values.

Self-Illum/Cutout Hdr

Like the main shader in this family, but with an alpha cutout mask, stored in the alpha of the base color.

Transparent:

Transparent/Rsrm & Transparent/Cube

The standard translucent shader in Alloy. Best used with an alpha value near '0'. Remember that this is a forward-only shader, and will only receive a number of real-time light influences as set in your quality settings. Additionally, note that the Metalness parameter will affect the opacity of this material, as higher reflectivity results in lower refractivity.

Transparent/Rsrm Distort & Transparent/Cube Distort

All the features of the shader above, but with grab-pass powered refractive distortion. This distortion occurs in screen-space and is influenced by distortion power slider and the normal map on this material. Note that grab-pass shaders are very costly, and should be used sparingly.

Transparent/Rsrm Distort MaskedIncandescence & Transparent/Cube Distort MaskedIncandescence

All the features of the shader above but with the Masked Incandescence setup described in the Core family.

Transparent/Rsrm Distort MaskedIncandescence Rim & Transparent/Cube Distort MaskedIncandescence Rim

All the features of the shader above, but with the ability to add glowy rim lighting.

Transparent/Rsrm MaskedIncandescence Rim & Transparent/Cube MaskedIncandescence Rim

All the features of the shader above, but without the grab-pass distortion.

Transparent/MaskedIncandescence Rim

A translucent shader that does not respond to lighting at all. Purely gets its appearance from Masked Incandescence and Rim lighting. By being purely the product of additive passes, this shader effectively exhibits order-independent transparency in most use-cases, so it perfect for environments that need lots of overlapping transparent-looking surfaces.

Transparent/Cutout/Rsrm & Transparent/Cutout/Cube

A standard Alpha-cutout shader with the parameter set of the base Core shader. Use this for vegetation!

Section 3 - Shader Parameter Details

Alloy Shaders have a set of parameters and maps that you will see recur throughout the set. This section will cover those parameters, what their values mean in abstract, and some workflow advice for ranges that will produce good results.

Section 3.A - Common Parameters

1. Main Color: This color picker will be multiplied against the Base texture below. Note that it's alpha value will only be utilized on transparent variants. This will affect the bounce and emission color for lightmap generation.

2. Base (RGB): This texture represents the base tone of the material. As it is the only per-pixel color map for the core shaders, this map plays 'double duty' when it comes to reflective surfaces. Essentially, it will inform both the 'diffuse' and 'specular' coloration of the material based upon the settings below. Note that the texture repeats/offsets values set here will determine the mapping for ALL of the other common texture maps. None of the other texture repeat/offset boxes will function. This map will determine the bounce and emission colors for lightmap generation.

3. Metalness: This slider, which goes from 0-1, determines whether the material is physically calculated as being a standard material, or a metalloid. Behind the scenes, what is happening here is that the Base texture/tint above is being pushed to the f0 (specular constant) of the material, while the albedo of the material is being pushed towards black. In practice, a majority of materials (that are physically plausible) will have this slider all the way at '0' or all the way at '1'. The only exception to this is exotic or composite materials such as glass, glazed surfaces, exotic plastics, and car paint, which exhibit some visual characteristics of matte and metalloid. These types of materials might have this Metalness value somewhere in between.

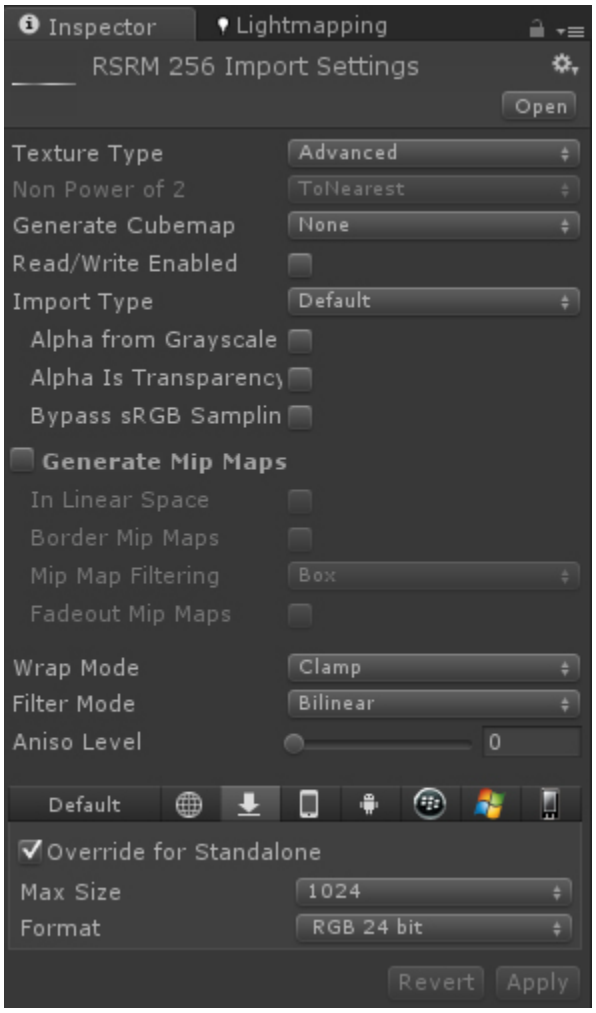
4. Smoothness: This slider, which goes from 0-1, determines the materials micro-texture, moving from fully-rough (or matte) at '0', to fully smooth at '1'. This will result in both a brighter and tighter specular highlight as smoothness increases. As Alloy shaders are energy conserving, no combination of Metalness and Smoothness will make your material globally brighter than the combined incoming light energy, but will instead focus that energy differently. Note that in common use, most materials will never crest past .7-.8 in smoothness, as even most common metal objects have some degree of micro-texture. 1.0 smoothness should only be used for polished mirror surfaces.

5. Metalness (R) Smoothness (A): The texture allows per-pixel control of the Metalness and Smoothness parameters above. Note that this is multiplied against the sliders above, so if you are determining these values per-pixel, you will likely want to the two sliders above set to 1.0. These channels are placed in the Red and Alpha channel respectively so they can be 'channel packed' with the Ambient Occlusion texture below (which uses the Green channel), once you've got your maps finalized, to lower unique texture usage.

6. Ambient Occlusion (G): Fairly self explanatory. Through you can place an Black and White RGB texture in here, only the Green channel will be referenced, allowing you to pack your AO with the Metalness and Smoothness per-pixel maps above.

7. Normalmap: Nothing special here, just your run-of-the-mill normal map.

8. Radially-Symmetric Reflection Map: RUST's homebrew solution for generalized horizon-style reflectivity. This is a pre-computed Look-Up-Texture, that MUST be set to the included RSRM 256 texture found in the Shader directory. Please ensure the import settings for this texture are setup to match the following image or you will have problems.



9. Reflection Cube Map: When using Alloy 'Cube' variants, you will also be able to plug a regular Unity Cubemap into this channel. Note that as this system was designed around sharp cubemaps, this map will be cross-faded with the output of the RSRM based upon the Smoothness of the material (as only mirror-smooth surfaces should perfectly reflect their environment).

Section 3.B - Special Opaque Parameters

1. Rim Tint: This color picker will determine the color of the rim lighting. The alpha value does nothing.

2. Rim Intensity: This float value will determine the intensity of rim lighting. This parameter is HDR-compliant, so if you want your rim lighting to 'glow' ensure that this value exceeds the Bloom Threshold of your post-fx.

3. Rim Power: This float value will determine the breadth/width of the rim lighting, with lower values spreading the rim lighting out further towards the view direction.

4. Detail (RGB): This texture will be multiplicatively blended over the base color above but can be mapped differently using the texture repeat/offset boxes directly beneath it. Note that this will make materials using this map appear MUCH darker, so it is recommended that you use a very bright/subtle texture here.

5. Detail Ambient Occlusion (G): This texture will be multiplicatively blended over the Ambient Occlusion texture above but can be mapped differently using the texture repeat/offset boxes of the Detail texture.

6. Detail Intensity: This slider, which goes from '0' to '1', determines the intensity of the detail normal map. This can be used for either fading in cracks/damage, or just pulling back on blended-in detail without changing the map.

7. Detail Normalmap: Nothing special here, just another normal map. IMPORTANTLY, unity will not complain if you plug a texture in here that hasn't yet been set to 'normal map' in its import settings. Always check that you have done this before putting a map in here, or you'll get straaange results.

8. Incandescence Tint: This color picker will tint/multiply against the values of the Incandescence map below. The alpha value does nothing.

9. Incandescence Intensity: This float (which defaults to '0'), will multiply the contribution of the Incandescence map below. While fully HDR compliant (you can set this to up to '8'), it WILL shift the hue of your map, so you may have to compensate by using the tint above to get the final bloomed color to be at the hue you want.

10. Incandescence (RGB): This texture allows for full-color incandescence, which is multiplied by the Intensity parameter above. Note that the texture repeat/offset parameters fully work and are independent of the rest of the material's. Note that this form of Incandescence will NOT be sent to the Beast lightmapper.

11. Incandescence Mask Scale: This slider, which goes from '0' to '1', controls the intensity of the Incandescence Mask below.

12. Incandescence Mask (A): This texture (which I commonly store in the alpha channel of my incan maps), allows you to mask off parts of your Incandescence map, as it uses the Base Texture's texture repeats/offsets. Conceptually, what this allows if for you to have 'zones' on your model that are Incandescent that line up with the Base map at all times, but then scroll the Incandescence Map (which would in this case be a seamless texture with a nice color range), giving the impression of moving energy/shields/fire, or other effects. Several examples of this can be found in the Demo Scene.

Section 3.C - Self-Illum Parameters

Note that the variants in this family are here specifically for objects that you want to be emissive AND have that emission be sent to the Beast Lightmapper. If you simply want glowy bits on characters/vfx/etc. you likely want to use the shaders found in the Core family.

Note also that if you want colored emission to be sent to Beast, that color MUST be represented on the appropriate pixels of the Base texture.

1. Emission Tint: This color picker will tint the emissive contribution of the material. The alpha value does nothing. Note that this tint will NOT affect the values sent to Beast.

2. Emission Intensity: This float value will determine the intensity of the emissive contribution of the material. Note that this will NOT affect the values sent to Beast.

3. Emission (Lightmapper): This float value will determine the intensity of the emissive contribution that will be sent to the Beast Lightmapper.

4. Illum (A): This texture mask will set what pixels of the Base texture will be counted as emissive, both for on-screen effects, and for what is sent to the Beast Lightmapper.

Section 3.D - Transparent Parameters

To start with, there are a couple things you should understand about translucent objects in a Physically-based model. The first of these is that the energy-conserving nature extends to the way translucence is handled. Merely setting the Alpha value of the Base map/tint will not be enough to make a material fully transparent. The Metalness of the material will also affect how transparent the material will appear, as it controls the relative balance of _reflection_ and _refraction_.

The easiest way to think about this is that alpha darkens albedo, but doesn't affect f0. f0 darkens albebo, and raises alpha to opaque. Thus diffuse + refraction + reflection <= 1.0. If this is confusing to you, please play with the sliders on the sample glass materials found in the Demo Scene to see what effect they have.

The second important thing is that all translucent shaders in Alloy are Forward-rendering only. This means you will only see dynamic specular highlights from a number of lights equal to the 'max pixel lights' parameter found in your quality settings. I tend to keep this value at 5-6.

1. Distortion: This slider will determine the amount by which pixels behind the material will be distorted in screen-space by the normal map on this material. NOTE that materials will this parameter are using a grab-pass for their effect, so use them judiciously, as multiple grab-passes can obliterate the performance of complex scenes.

2. Alpha cutout: This slider, found on the two cutout variants, is used to control the interpretation of the Alpha channel of the Base texture, in determining which pixels of the texture are made 100% transparent. Note that cutout shaders ARE deferred-compatible, and will cast dynamic shaders in-game and when Lightmapped. Delicious.