Smooth Fresnel Shader Short Documentation

The Smooth Fresnel Shader (SFS) pack is a compilation of Unity surface shaders based on a unified light model combining Fresnel, Phong and Lambert lighting with customizable light wrapping.

It contains the following shaders in 2 versions - standard and vertex colored:

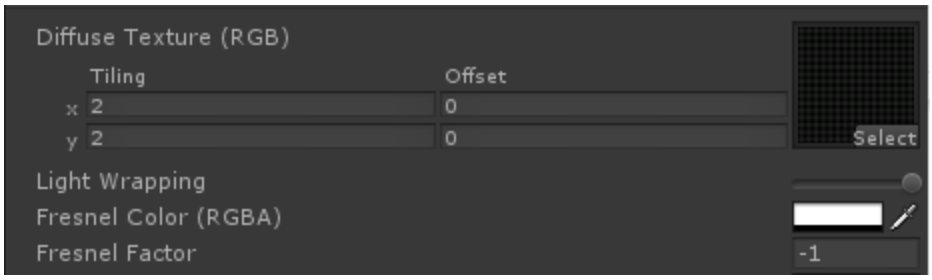
- Diffuse
- Diffuse & Dissolve
- Diffuse, Dissolve & Self-Illumination
- Diffuse & Self-Illumination
- Diffuse, Self-Illumination & Transparency
- Diffuse & Specular
- Diffuse, Specular & Dissolve
- Diffuse, Specular, Dissolve & Self-Illumination
- Diffuse, Specular & Self-Illumination
- Diffuse, Specular, Self-Illumination & Transparency
- Diffuse, Specular & Translucent
- Diffuse, Specular & Transparency
- Diffuse & Translucent
- Diffuse & Transparency
- Normal Mapped
- Normal Mapped & Dissolve
- Normal Mapped, Dissolve & Self-Illumination
- Normal Mapped & Self-Illumination
- Normal Mapped, Self-Illumination & Transparency
- Normal Mapped & Specular
- Normal Mapped, Specular & Dissolve
- Normal Mapped, Specular, Dissolve & Self-Illumination
- Normal Mapped, Specular & Self-Illumination
- Normal Mapped, Specular, Self-Illumination & Transparency
- Normal Mapped, Specular & Translucent
- Normal Mapped, Specular & Transparency
- Normal Mapped & Translucent
- Normal Mapped & Transparency

totalling up to 56 shaders. Additionally, the pack contains 6 shaders for use with Unity terrains: Diffuse, Diffuse Translucent, Normal Mapped, Normal Mapped Translucent, Specular Normal Mapped & Specular Normal Mapped Translucent.

This results in 62 shaders you can use in your games.

Lighting options

There are a number of options that are available across all shaders in the SFS pack:



Diffuse Texture `[_Diffuse]`
This is your objects main texture.

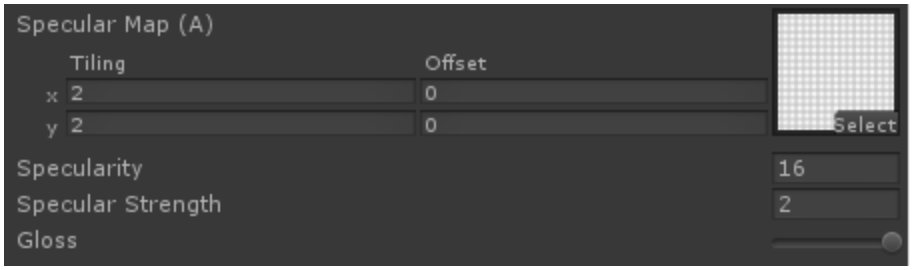
Light Wrapping `[_Wrapping]`
This slider lets you set the light wrapping for the object where left is standard Lambert lighting and right is view based lighting creating an effect of broad light distribution.

Fresnel Color `[_Color]`
This is the color of the fresnel effect. Most cheap fresnel based effects use white but you can use colors like red or green for interesting effects as well. The alpha value lets you set the interpolation between the fresnel color and the diffuse color. An alpha value of 0 means the object will fresnel with its respective diffuse color; this creates a more subtle and natural fresnel effect.
With dissolve shaders, the value is named `[_FColor]`.

Fresnel Factor `[_Factor]`
How strong do you want your fresnel to be? ;-) Use negative values for stuff like creepy light falloff effects!

Apart from those, every special shader has it's own additional options that we'll discuss now.

Specularity



Specular Map `[_SpecMap]`
This is an alpha map used for selective specularity. The most basic usage would be to take the diffuse texture and erase all parts that you don't want to have specularity; you can also leave it empty to have specularity on the whole object.

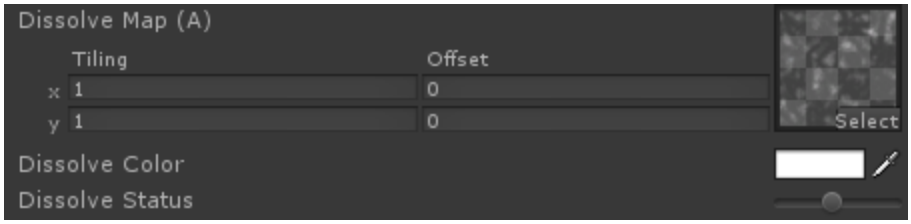
Specularity `[_SpecK]`
The lower this value, the rougher your specular distribution will be; in other words: high values will give you more sophisticated specular details.

Specular Strength `[_Strength]`
How much light should be reflected by the specular details? Higher values mean brighter specular details. Negative values will give you an inverted specular effect meaning instead of high light reflections you will get points that appear to "suck away" the lighting.

Gloss `[_Glossyness]`
Technically, this interpolates the color of the specularity between diffuse and light color. Left means the specularity color will be based on the diffuse texture while right means it will be based on the light's color. For most materials, basing specularity on the diffuse color is fine and will deliver a very smooth specular effect (e.g. a golden armor will look great) but for darker diffuse materials the specularity will appear to be weak and even vanish for black materials; use higher gloss values in those cases.

Dissolve

The dissolve effect lets an object disappear based on an alpha map and a dissolve value.



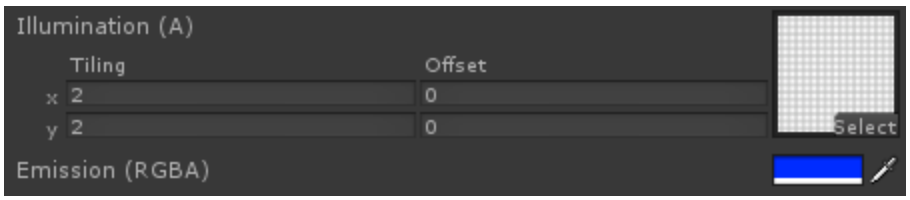
Dissolve Map `[_MainTex]`
An alpha map used for dissolve distribution. The lower the alpha value in this texture, the faster the object will be dissolved there. Cloud or noise maps are your best bets here unless you need a very specific dissolve effect.

Dissolve Color `[_DisColor]`
This color will determine how the model will be colored while being dissolved, e.g. a red/orange value will result in a fiery effect. When setting the alpha value of the color to 0 the dissolve effect will kick in without color changing. This will be slightly faster because the color effect is based around a color distribution calculation that will not be made when ignoring the color.

Dissolve Status `[_Cutoff]`
The current status of dissolving where left means the object will be completely visible and right means complete dissolve. This value can be accessed via script by the name of _Cutoff and uses a range from 0.0 to 1.0 so you can animate the dissolve on runtime.

Self-Illumination

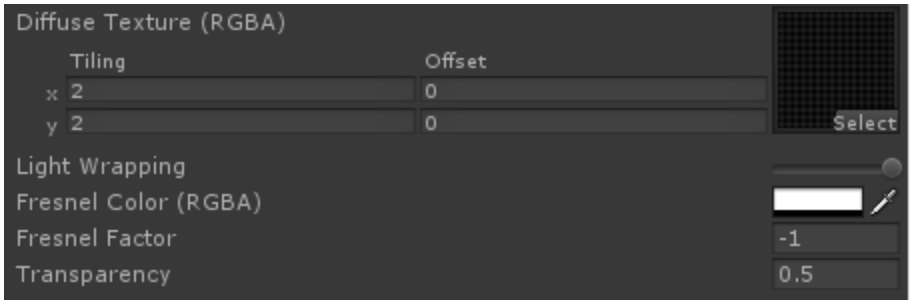
Self-Illumination is a great effect with a wide variety of uses. You can for example light up windows or lanterns. You can also use them for neon signs although you might want to additionally use a full screen glow effect to get a fake environmental illumination effect.



Illumination `[_Illumination]`
This is you alpha map for specifying which parts of the object should “glow”.

Emission `[_Emission]`
The color of your self-illumination. The alpha value is used to determine the strength of the illumination. This value can be accessed via script by the name of _Emission and its alpha value uses a range from 0.0 to 1.0 so you can animate the illumination on runtime.

Transparency



Diffuse Texture `[_Diffuse]`
The transparency shaders of SFS use the diffuse texture's alpha channel for texture based transparency. That way you can use the transparency shader not only to let an object be semi-transparent or vanish but also for cheap cut-out effects.

Transparency `[_Transparency]`
A value between 0.0 and 1.0 to determine how solid the object should be. 0.0 means full transparent while 1.0 is 100% solid. This value can be accessed via script by the name of _Transparency so you can animate the transparency on runtime.

Translucent

Translucent material lets light information pass through. This is usefull for material like skin, wax and other cool effects.

SubSurf Dis. `[_Distr]`
The subsurface distribution factor. The higher the factor, the broader the light will be scattered through and around the object. When animating this factor by scripts, keep in mind that the range of the slider is from left = 1.0 to right= -2.0.

SubSurf Pow. `[Power]`
The subsurface power. The higher the power, the more light will be let through. When animating this factor by scripts, keep in mind that the range of the slider is from left = 0.5 to right = 3.5.

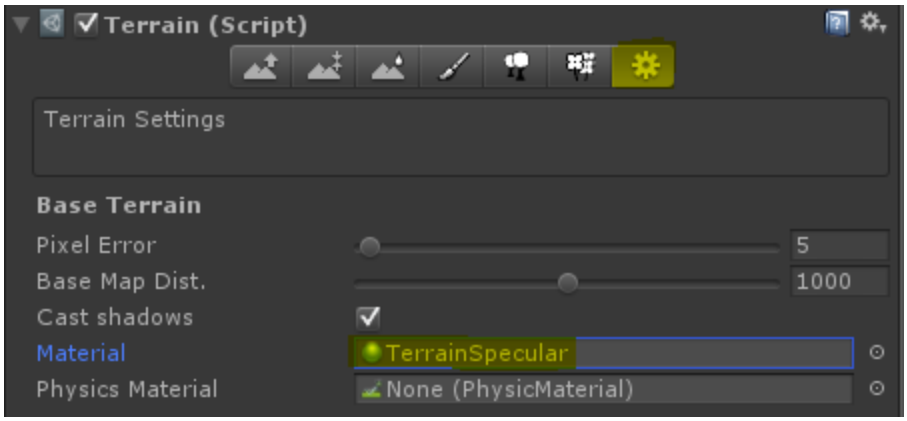
Standard vs. Prototype

While the standard shaders should be fine for most usecases, some of you may want to use the vertex colors of their models or - like us - use ProCore's fabulous Prototype tool which allows you to build geometry inside the Unity editor and apply vertex colors.

This is what the Prototype versions of the shaders are for; those will factor in the vertex colors of your objects. Since this means an additional multiplication those shaders require slightly more GPU power. However, the Prototype shaders also use correct viewing angles instead of approximate ones because approximate viewing angles are giving weird results on extreme low polygon objects; therefore, it's best to use standard shaders for performance reasons.

Terrain shaders

The SFS pack also features 3 shaders with which you can create materials to use with your terrains.



Those materials will feature the same options as the standard SFS pack shaders.

Shader Requirements

Almost all diffuse shaders of SFS run on Shader Model 2.0 hardware with the exception of dissolve shaders which require Shader Model 3.0.
As soon as you start using normal maps, things get a little bit more complicated:

A rule of thumb for nomral mapping is: does it use specular mapping, translucency and/or dissolve? Then you will need Shader Model 3.0 hardware. Otherwise, Shader Model 2.0 should be fine.

Terrain shaders always require Shader Model 3.0.

A word on mobile
The SFS pack was not created with mobile platforms in mind. However, all shaders should at least run on mobile if the hardware supports the respective needed shader model. A great range of mobile GPUs doesn't support Shader Model 3.0 (like the Mali-400MP found in smartphones like the Galaxy SIII) but it's getting more and more adopted.

So if you want to aim for maximum compatibility, you will want to stay away from shaders that require Shader Model 3.0; you should also avoid transparency shaders.