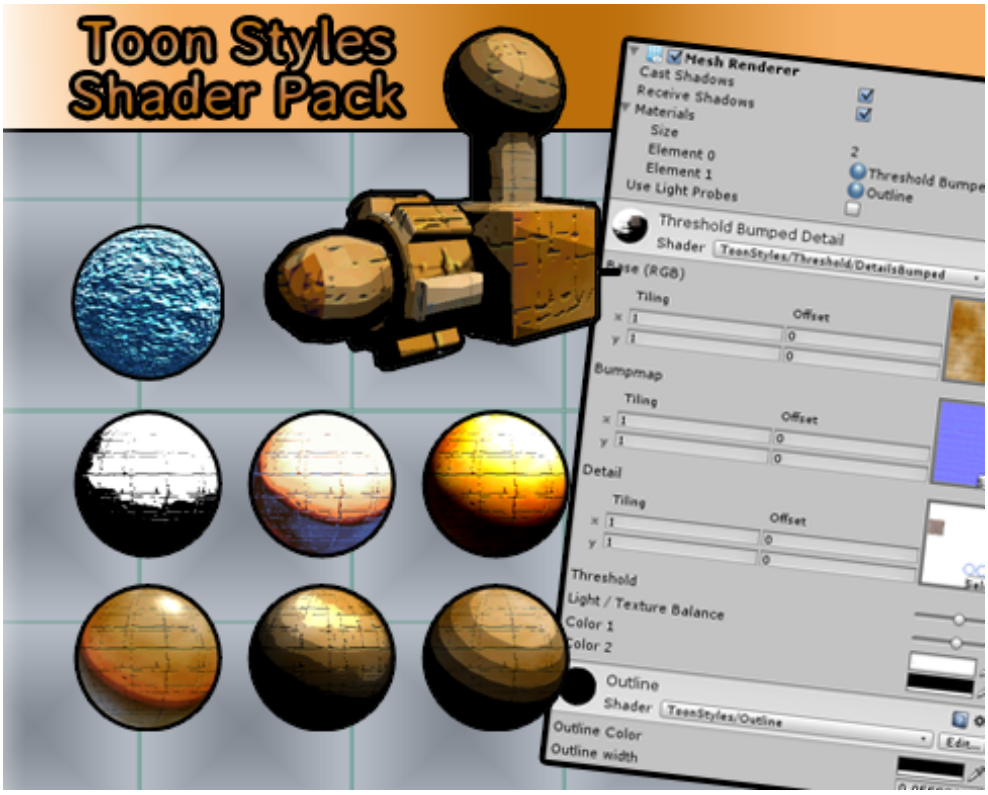
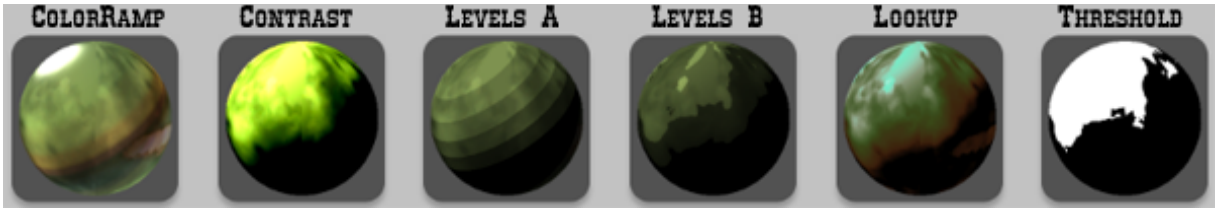
Toon Styles Shader Pack



1 Overview

The Toon Styles Shader Pack contains an assortment of surface shaders intended to produce various lighted cartoon-type visuals. The shaders are divided into **styles**, such as *Threshold* and *Lookup*, and each shader style has a number of variants, such as *Diffuse* or *Bump*.

In addition to shaders the pack contains premade toon ramps and a demo scene with its assosiated assets.

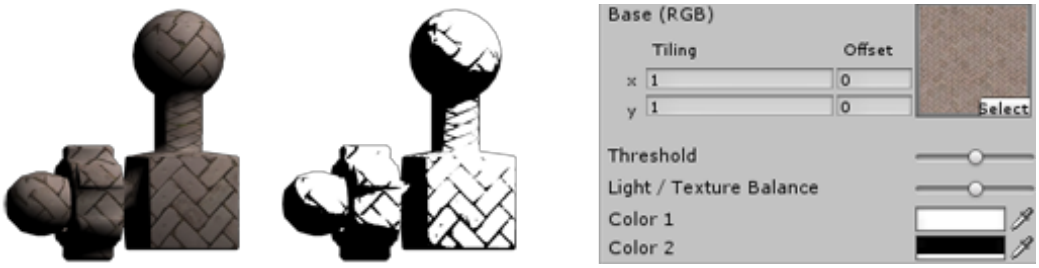


Version history:
1.0
- Initial release
1.1.1
- Added new ramp textures
- Added Outline B
- Fixed some shaders being darker on iOS device than on the editor

2 The Shaders

2.1 Threshold Style

Threshold shaders apply one of two colors to the final color of the object based on the luminance of the pixels after the base texture and lighting has been applied.



An object with diffuse shader and threshold shader with colors set to white and black.

Shader-specific controls:

- Threshold (_Threshold) : Range(0, 1)
    - The cut-off point for the two colors
- Light / Texture Balance (_LightBalance) : Range(0, 1)
    - How much the final color is affected by lights and by the base texture
- Color 1 (_Color) : Color
    - The color in the light areas of the object
- Color 2 (_Color1) : Color
    - The shadows color

2.2 Lookup Style

Lookup shaders replace the objects final color with a color picked from a lookup texture. The picked color depends on the luminance of the pixels after the base texture and lighting has been applied. The brighter the pixel, the further to the right on the lookup texture the replacement color is.

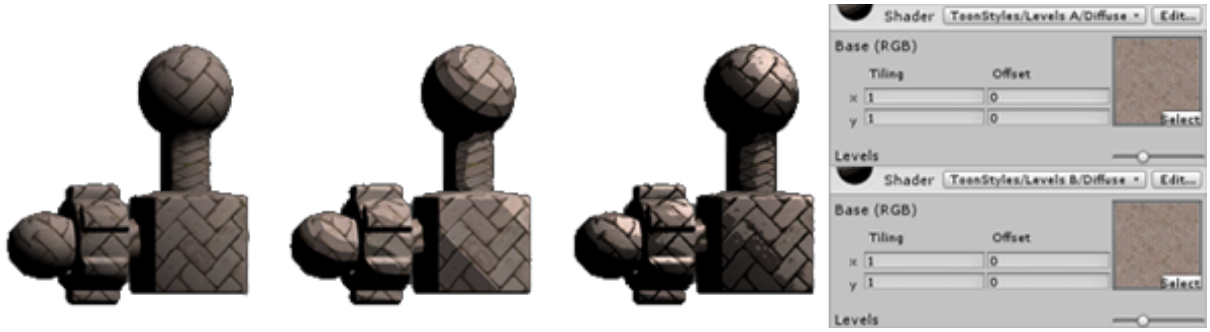
An object with diffuse shader and lookup shader with a colorful lookup texture.

Shader-specific controls:

- Lookup Offset (_Offset) : Float
    - Offsets the lookup position to the right (or left if negative)
- Lookup Scale (_Scale) : Float
    - Multiplies the calculated luminance. In effect enhances contrast
- Lookup Texture (_Lookup) : Texture
    - The texture whose colors will be used to replace to original

2.3 Levels Style

Levels shaders apply a quantized coloring effect on the object. There are two variants of the Levels Shaders: Variant A is applied *during* the lighting calculations, and B is applied *after*. Variant A produces somewhat cleaner results with clear lines, wheras variant B has a more grittier effect on the object's visuals.



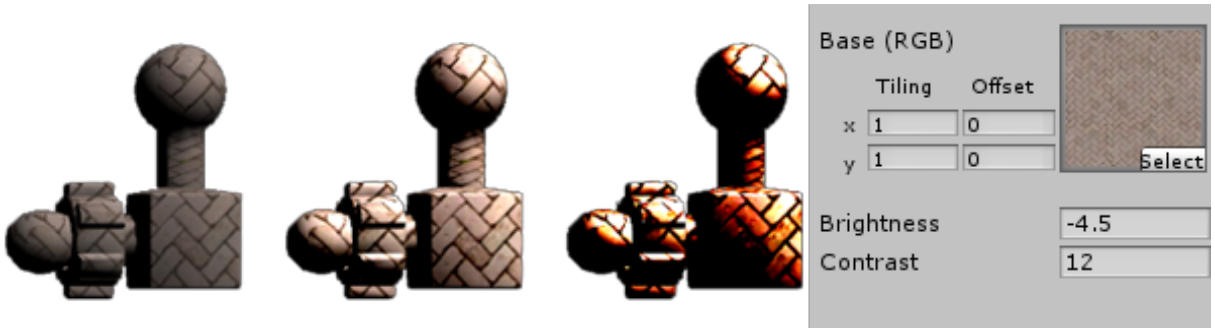
An object with diffuse shader and Levels A and Levels B shaders with identical settings.

Shader-specific controls:

- Levels (_Levels) : Range(2, 8)
    - The amount of coloring levels

2.4 Contrast Style

Contrast shaders add brightness and contrast controls on the material. Though this is not a toon effect as such, it can be used to achieve quite stylistic (and unreal) effects.



An object with diffuse shader (left) and contrast shaders with subtle (center) and radical (right) values.

Shader-specific controls:

- Brightness (_Brightness) : Float
    - The amount of toning levels
- Contrast (_Contrast) : Float
    - The amount of toning levels

2.5 ColorRamp Style

ColorRamp Shaders behave very much like Unity's standard lit toon shader, but have specular highlights. Another difference to Unity's toon shaders is the availability of shader variants, such as bump or transparent.

An object with diffuse, Unity's lighted toon and ColorRamp shaders.

Shader-specific controls:

- Coloring Ramp (_Ramp) : Texture
    - The toning ramp texture
- Shine (_Highlight) : Range (0, 64)
    - The size of the highlight. Note that the higher the value, the smaller the highlight

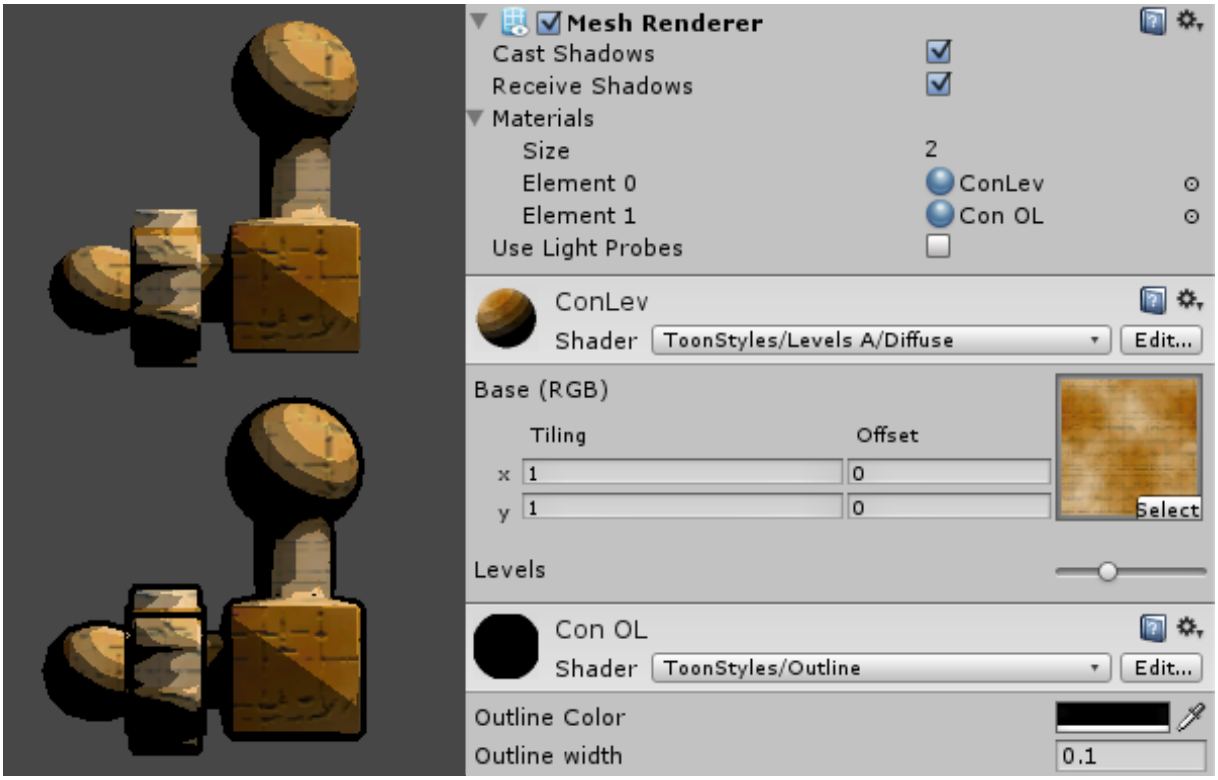
2.6 Outlines

The *ToonStyles/Outline* shader is used to draw outlines around objects. It is used by adding an outline material to a renderer as an additional material. Outlines work with any non-transparent material, not just the ones that come with this package. You can also add the outline to your own custom shader with the ShaderLab command

```
UsePass "ToonStyles/Outline/OUTLINE"
```

if you'd rather have the outline integrated in the shader rather than always adding an additional material.

Outline B variant tries to maintain the outline size regardless of distance to camera.



Levels A/Diffuse shader with and without outlines. Note how the Mesh Renderer has two materials.

Unlike Unity's own toon outline, the Toon Styles outline works in ortographic cameras, and its relative size to the object does not change with distance to camera.

Shader-specific controls:

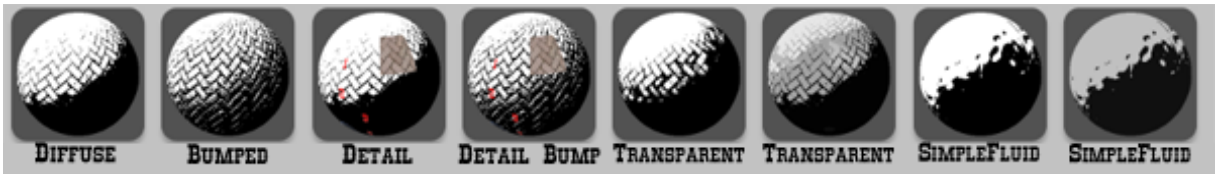
- Outline Color (_OutlineColor) : Color
    - The color of the outline. Can have transparency.
- Outline width (_Outline) : Float
    - The size of the outline.

A note on models with multiple materials on a single gameobject:

If the model is mapped to multiple materials, added extra materials are only applied to the last material mapping. When this is a problem, there are couple of different ways to get the outlines show properly.

- In your modelling program,
    - map one material specifically for the outline
    - *or* only use one material for one object
- In Unity, create a duplicate of your object and set its materials to a material using an outline shader

2.7 Shader Variants



All shader styles have several variants available, though not all styles have all variants. Variants can also appeat in different combinations.

2.7.1 Diffuse

The most basic lighted shader. The fastest of the bunch, so you should default to the Diffuse variant when you don't really need any of the others.

2.7.2 Bumped

Applies a normal map to model giving it a bumpier and more detailed look.

2.7.3 Detail

A detail texture is added on top of the model *after* the shader-specific effects have been applied. The detail textures alpha channel is used to determine where it is applied, so you can have other parts of your model use shader's effect and other parts show the details using basically a diffuse effect. The detail texture needs to be supplied.

2.7.4 Transparent

Single and double-sided shader variants that allow for transparent textures, and transparent colors in the case of threshold shaders.

2.7.5 SimpleFluid

Nice, simple animated fluid, primarily intended to be used as water surface. Quite similar to Unity's simple water, but works better with lighting, doesnt't require a script and has transparency support.

2.8 Unlit Background Shader

The *ToonStyles/Unlit-Background* shader is used to draw unlit textures so that they always appear behind other geometry, regardless of world position. It is used in the ShaderGallery example scene.

3 Tips

3.1 Performance

The **outlines** create an additional draw call and double the number of polygons on screen, so they should be used with care, especially on mobile platforms. As always, use smart texture atlasing to keep draw calls to a minimum and try to keep your models as low poly as you can get away with. In some cases, outlines can be replaces by drawing them directly on the model textures. For high-poly models it might prove to be beneficial to create a separate, low-poly version of the model for use with the outline material.

For **static** objects with static lighting an unlit material will sometimes suffice; you can then draw the desired effect directly on the texture and save a lot in terms of performance. Unity's own Mobily/Unlit (Supports Lightmap) is especially handy.

Transparent shaders should be used sparingly on mobile platforms.

Use **diffuse** variants of the shaders when more complex shaders aren't absolutely necessary.

3.2 Stylistic

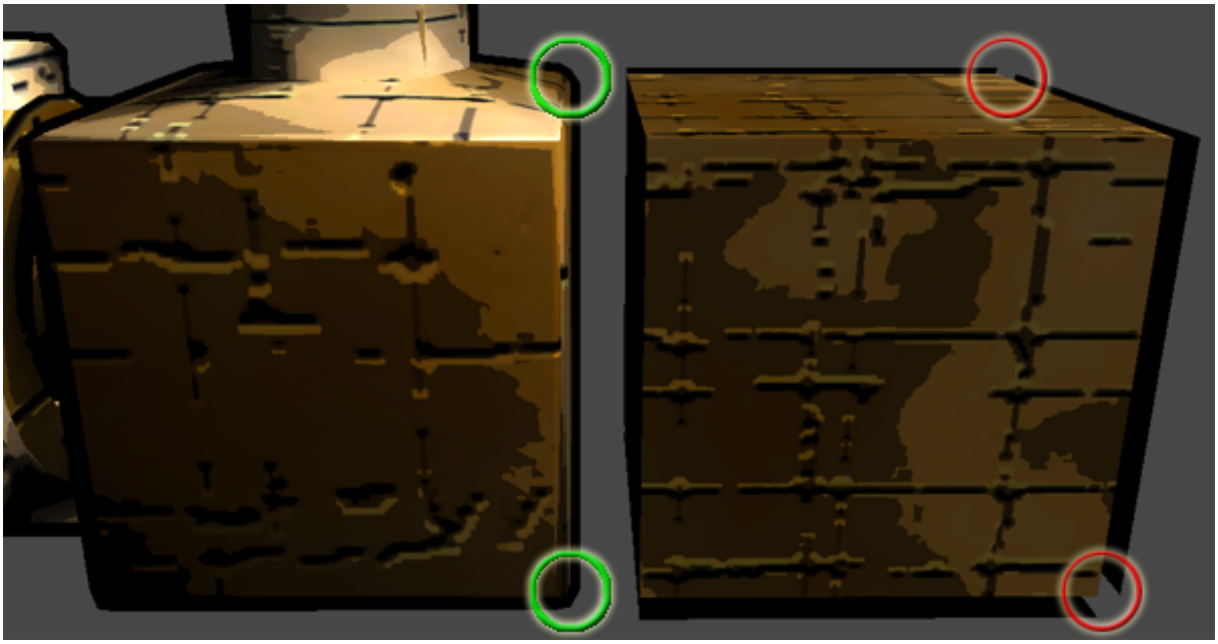
For cleaner, more toon-like effect the **diffuse** variants are often a better choice than the bumped variants, which tend to produce grittier and more realistic look.

For **Lookup** and **ColorRamp** shaders, experiment with different ramp textures. Using a single ramp, or a set of similar ramps, helps in keeping the visual style of the game consistent.

Outlines can help to make your models 'pop', especially if you use them sparingly (i.e. only for those objects who *need* to stand out). The ouline doesn't need to be black, but can be colored either for visual effect or to inform the player (red for enemies, blue for pickups etc.). The outline can also be colored in code, even flashed to make something *really* stand out. The outlines also support transparency.

```
myOutlineMaterial.SetColor("_OutlineColor", myColor);
```

Slightly **rounded** shapes usually react to lighting a bit more nicely, and very sharp angles on models can break the outlines. Use slightly convex shapes for a nicer effect.



These models share the same materials, but the model on the left has rounded corners, and the top surface is bulging slightly as opposed to the completely flat face on the right.

The **lighting** of the object is really important in all 3D games, but in toon games it can be especially tricky to get it 'just right'. Here's a couple of tips:

- Use ambient light to your advantage! (Edit->Render Settings->Ambient Light) Ambient light basically sets the color of your shadows.
    - Set it to black to get high contrasts and to make your lights more impactful
    - Set it to dull grey to make the scene seem washed out and flat
    - Set it to a (dark) color to help give your scene some character
- Lights have a color property, use it
- Simpler is usually better
- Lights can have their color and intensity animated in code
    - Flickering point lights for fire (torches, fireplaces, etc.)
    - Smoothly interpolating intensity for directional lights in outdoor scenes can create a nice illusion of rolling clouds overhead

3.3 Customizing the Shaders

The main functionality of the shaders is defined in the */Shaders/toonstyles.cginc* file.

A good practise in making your own, customized versions of the shaders is not to edit the shaders or the cginc file directly, but to make a duplicate of the cginc file. Also make a duplicate of the shader you wish to modify and then include your modified cginc file instead of the original. This way, if the shader pack is updated, you can download the updated version without overwriting your modifications.

```
#include “../toonstyle.cginc”
```

to

```
#include “path/to/myfile/mytoonstyle.cginc”
```

Remember to give a new name to the duplicated shaders!

```
Shader "ToonStyles/ColorRamp/Bumped"
```

to

```
Shader "MyToonStyles/ColorRamp/Bumped2"
```

Never move the shaders or the cginc files relative to each other. The shaders include the cginc file from their parent folder, and if they can’t find it there, the shaders won’t work.

4 Contact

Toon tyles Shader Pack is released for Unity Asset Store and maintained by Ifelse Media Ltd. We are happy to provide support for our products and to answer any questions.

Support email: support@ifelsemedia.com

Website: http://ifelsemedia.com/
Development blog: http://ifelsemedia.tumblr.com/
Facebook: https://www.facebook.com/IfelseMedia
Twitter: https://twitter.com/IfelseMedia
YouTube Channel: http://www.youtube.com/user/IfelseMedia

The developer responsible for this product is Timo Moisio, and can be contacted directly at timo.moisio@ifelsemedia.com

Have specific needs for your game, an idea for a new toon style or feedback on your mind? Don't hesitate to send us email, we'd love to hear from you on how we could improve this package!