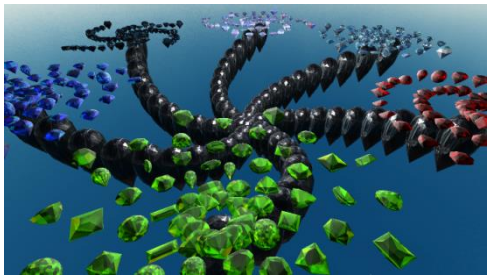
Advanced Gemstone Shaders

Introduction:



I am fascinated from crystals and minerals and gemstones. As a game developer, I always look for solutions to simulate such precious stones. When most people hear the word gem, they automatically think about diamonds, sapphires or rubies as they are most known to the mayor population. However there are way more different kinds of stones like opals or moonstones which are usually cabochons and are very difficult to simulate with shaders. I have checked already existing gemstone packages but was not satisfied with what I got.

About this Package:

Occasionally I have the need to work on shaders and suddenly 12 hours have passed. This package contains shaders I have made to simulate gemstones and will be updated whenever new shaders for gems are implemented. It also contains 50+ gemstone models of common shapes found in the jewelry business and a simple pattern generator to arrange these objects.

The Shaders:

Shaders of this package sometimes work different than standard shaders you know to archive such effects. However one common thing you will notice is the high usage of cube maps. They are indeed important to simulate the reflections and effects which occur in gemstones.

The aim for these shaders is to make it possible to create many different materials with a few textures as in my home environment, artists are very rare. Also keep in mind that some shaders require a reflection probe in order to look nice.

The best thing is really to just play around with the values and parameters. Often fancy looking material were created accidentally by just experimenting with parameters.

Another thing is that there are so many different kinds of gemstones that this package will never be completed and there is always something new to discover and improve. Also when new technologies emerge which may allow the simulation of inclusions.

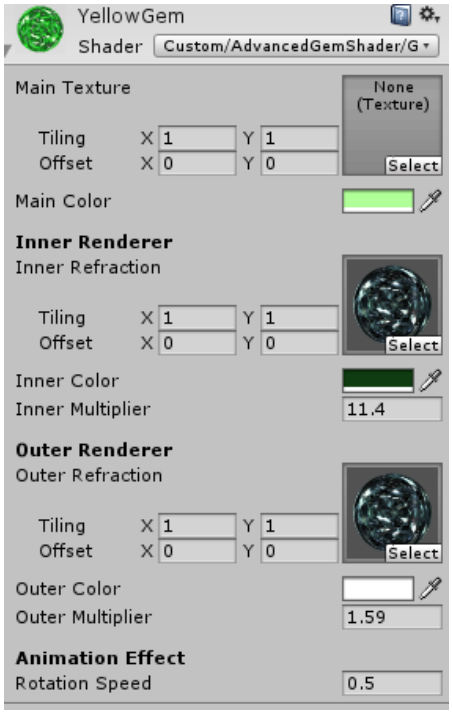
Shader 1: Gemstone Shader Simple

This is the first shader and the most simple one. You have full control over the visual appearance of the gemstone. However it does not look fully realistic but it does not need reflection probes.

This effect is acquired by rendering the interior and the outer parts separate. The most important parameters are the refraction and reflection textures which are simple cube maps. In the hints section are some tips about how to create them fast.

Parameters:

- Main Texture: standard main Texture
- Main Color: standard main Color



Inner Renderer:

- Inner Refraction: Inner reflection Cubemap
- Inner Color: Cubemap Color Multiplier
- Inner Multiplier: Power Multiplier

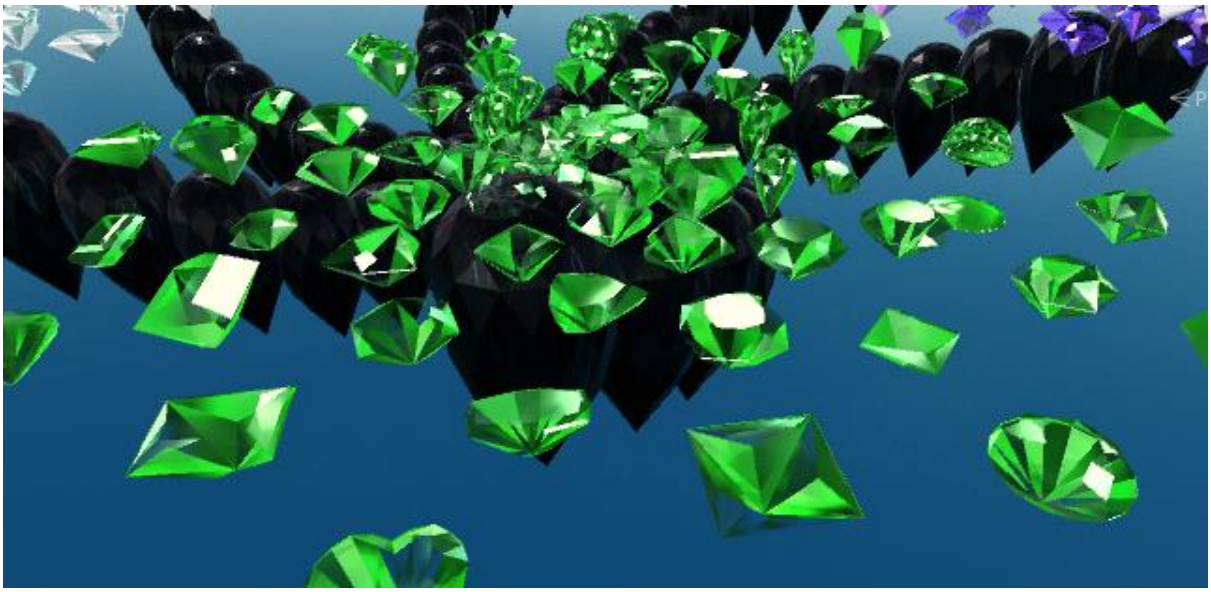
Outer Renderer:

- Same parameters as Inner renderer. Renders the outside of the gemstone

Animation Effect:

- Rotation Speed: Lets the cubemap rotate to give those gemstones magic like animations.

Usage: This shader is good for mobile applications or if the gemstone should stay unaffected from external influences. It is also possible to create unnatural gemstones where the sparkle color is black which is impossible in real life. Check out the “Curse” Material



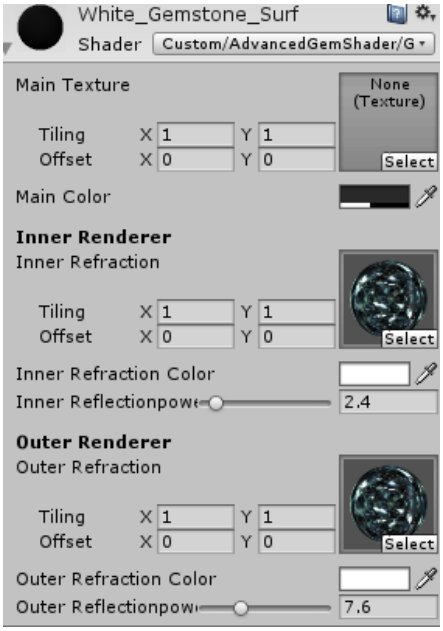
Shader 2: Gemstone Shader Realistic

This is the second shader and attempts to simulate gemstones as realistic as possible**. This shader requires a reflection probe.** If you don't want to use it, use the simple shader instead as It looks terrible without reflections.

This effect is also acquired by rendering the interior and the outer parts separate. The refraction and reflection textures are just multipliers for the probe reflection.

Parameters:

- Main Texture: The Main Texture.
- Main Color: Color Multiplier



Inner Renderer:

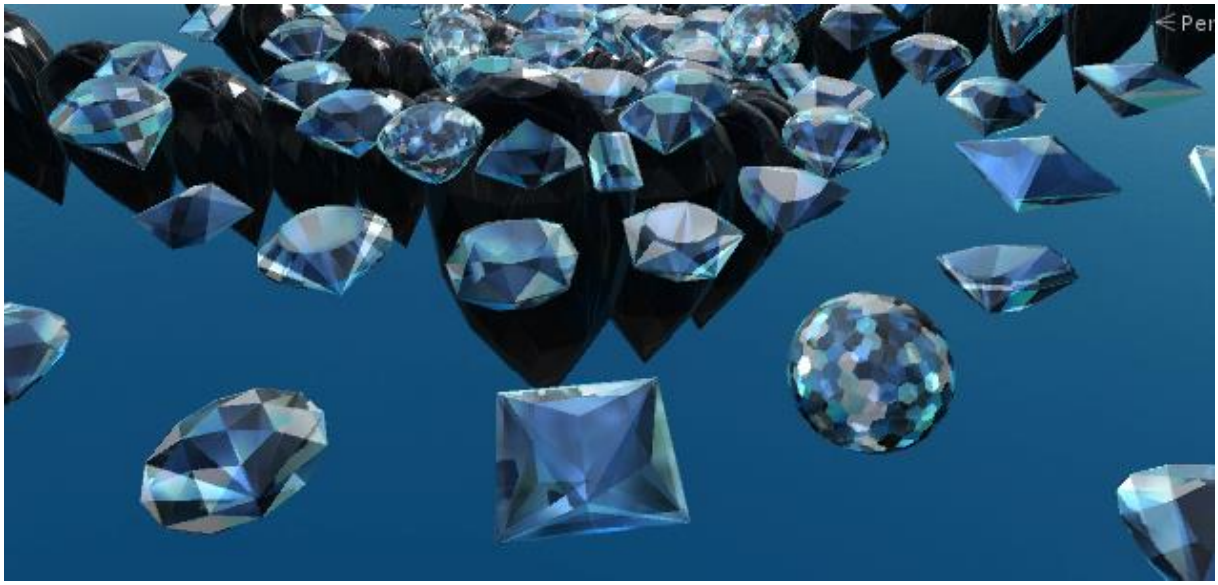
- Inner Refraction: Inner reflection Cubemap
- Inner Color: Cubemap Color Multiplier
- Inner Reflection Power: Influence Factor

Outer Renderer:

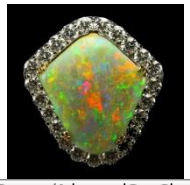
- Same parameters as Inner renderer. Renders the outside of the gemstone

Usage: This shader is good for realistic applications. Keep in mind that the gemstone will be just dark in environments without light and reflection like in real life.

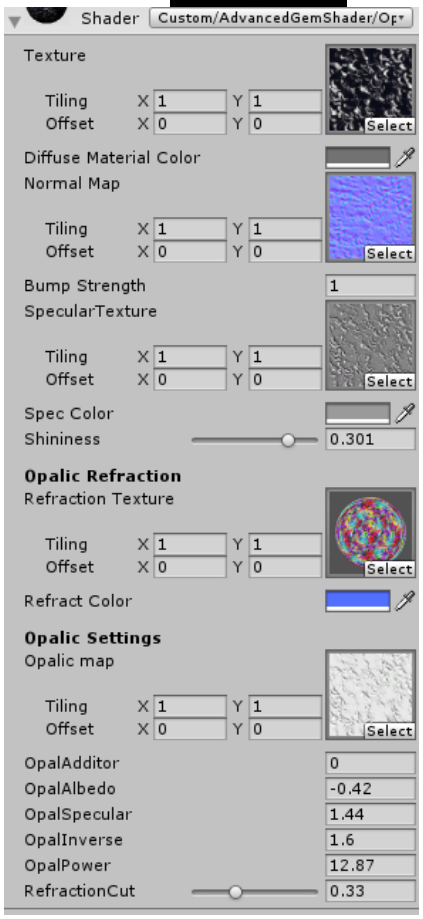
These 2 Shaders handle normal gemstones like diamonds with all their colors. Although they are not 100% realistic as we would need ray tracing which doesn't work in real time at the moment. However with the upcoming OpenGL update Vulcan, better results could be possible.



Shader 3: Opalic Normal/Refractive



In real life a special kind of gemstone exists which is completely different than common gemstones which are called opals. Opals are a special form of silica which is amorphous. Due to water inclusions, opals have sparkles in all colors. The right image is a real example of such stone. Currently rendering one would require some kind of 3D-Texture map.(image from Wikipedia)



This shader maybe comes close to 30-40% of a real opal as graphic engines have to be able to render inclusion without altering the actual geometry. This may be possible with procedural 3D Textures in the future. This shader uses the standardspecular shader as base

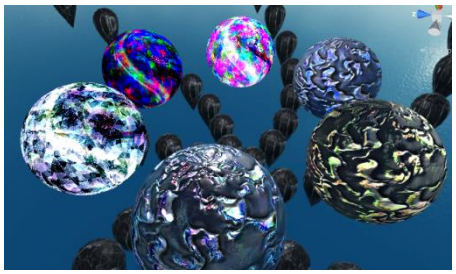
Parameters:

- Texture: The Main Texture.
- Diffuse Mat Color: Color Multiplier
- Normal Map: Standard normal map
- Bump Strength: Normal power
- Specular Texture: Normal specular texture
- Spec Color: Color Multiplier
- Shininess: Shininess for Reflection Probe

Opalic Refraction:

- Refraction Texture: Reflection texture of “opalic” Sections
- Refraction Color: Color Multiplier

Opalic Settings



This is a texture with an alpha channel. Alpha describes where “opalic” spots are. Color of this texture does not matter.

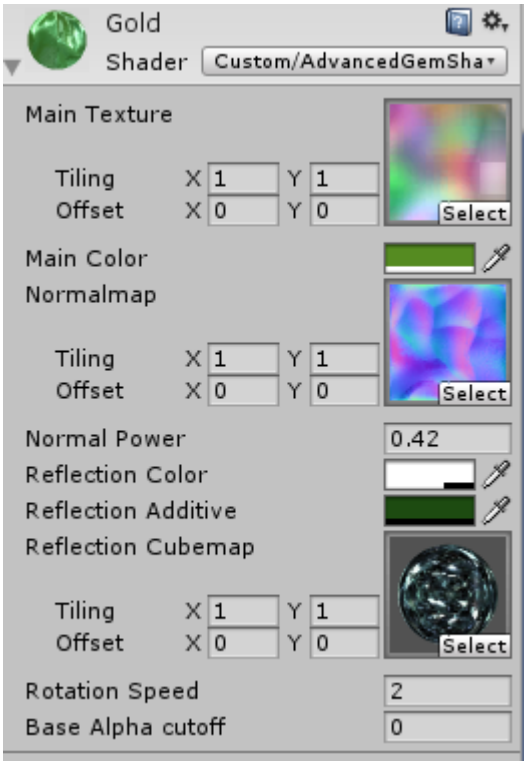
- OpalAdditor: Add value to the opalic value
- OpalicAlbedo: Influence to the albedo map
- OpalicSpecular: Influence to specular map
- OpalInverse: Multiplicator of the opalic map.
- OpalicPower: Power for the opalic map. Values with low alpha stay low while values with high get higher
- RefractionCut: Final multiplicator to the opalic value

Hint: Opalinverse can be used to invert the alpha of the map. Set inverse to -1 so alpha of 0.75 becomes -0.75 and set Additor to 1 resulting in -0.75 + 1 = 0.25

Shader 5: Metallic:

Sometimes you face the problem with simulating metallic material in 2D games. Usually you can use the standard shader and set the metallic value to maximum and throw in a reflection probe. However in 2D games, the metallic properties are not noticeable because the camera angle does not change.

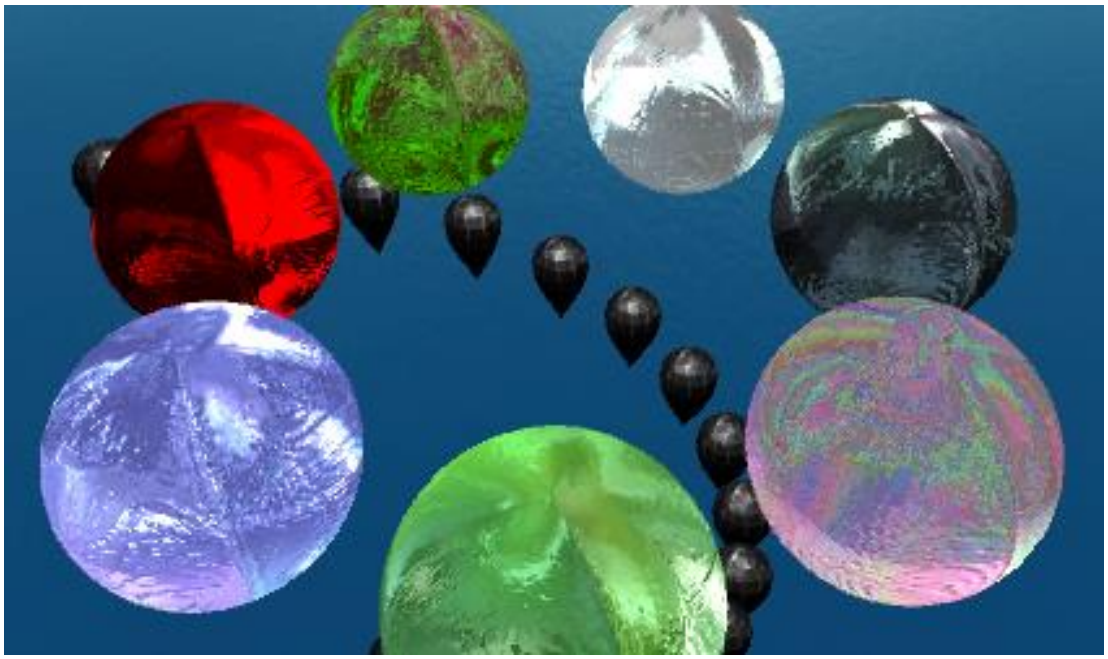
This shader aims to simulate metallic structures without reflection probes and perspective cameras.



Parameters:

- Main Texture: Standard Main Texture
- Main Color: The Main Color.
- Normal Map: standard normal map
- Normal Power: normal map power
- Reflection Color: reflection color
- Reflection Additive: fixed color added to the reflection
- Reflection Cubemap: Reflection map
- Rotation Speed: Reflection rotates so it can have animations.
- Alpha cutoff: supports cutoff to simulate fences, grids etc.

This shader has 2 variants, one with cutoff and one without. If your stuff don't need transparency, use the opaque version as opaque shaders always have a better performance than transparent shader.



Shader 6: Alexandrite

Alexandrite is an gemstone with very special properties. Its appearance depends on the color of the light. For example such gem is yellow in sunlight and blue in red light. Their internal structure changes the wavelength of incoming light. It uses the same parameters of the realistic gemstone shader and requires an reflection probe.

Parameters:

- **Red/Green/Blue Color:**

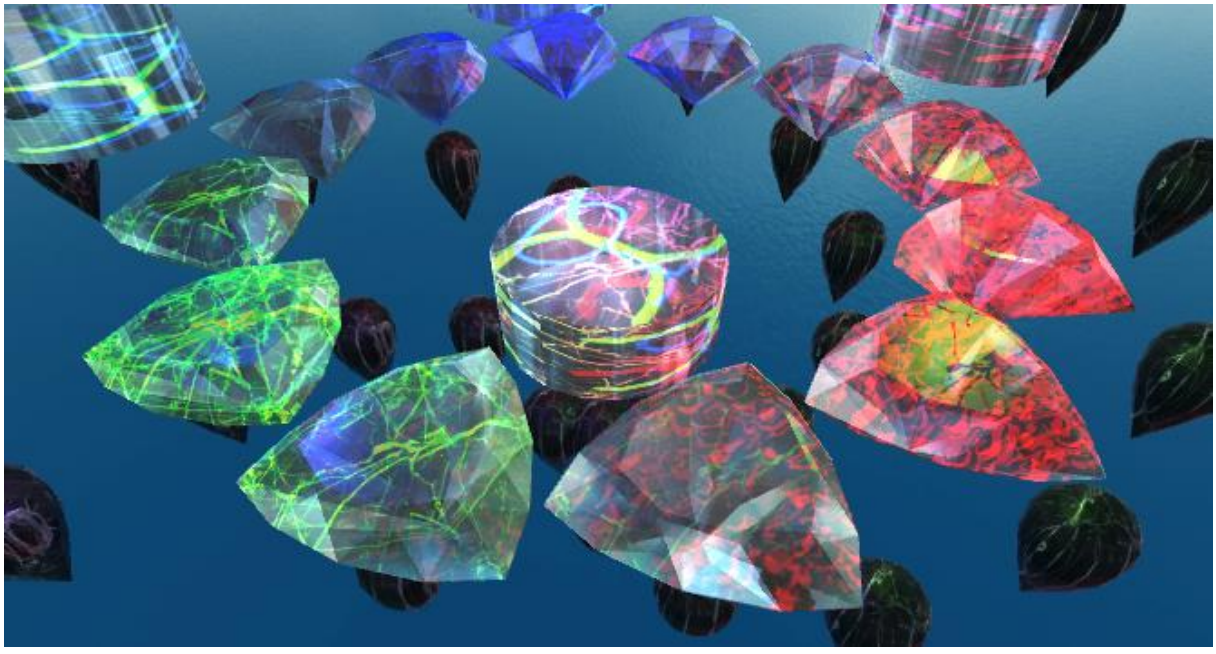
The color it should emit when colored light hits this material.

- **Red/Green/Blue Influence:**

Power of influence: 0 means that colored light of the specific color has no influence.

- **Red/Green/Blue Texture**

Texture influences the specific color emitted from the gemstone. It is also possible to make material which reveals hidden stuff or information when exposed to colored light. The sample shows a material where runes are visible under certain light. This effect is similar to fluorescence where materials emit yellow light when exposed to UV radiation

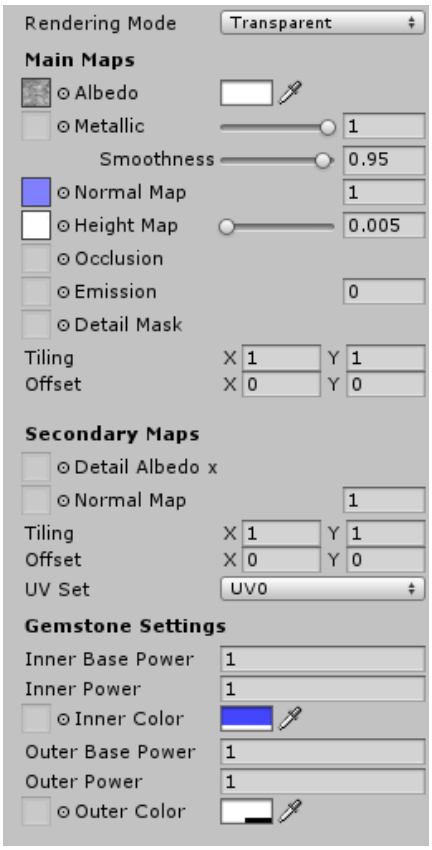


Shader 7/8: Standard Versions:

The standard gemstone shaders come in two variants which are the normal standard and standard specular versions. They are extended versions of the original standard shaders so they become gemstone like properties.

The main difference is only the addition of the Gemstone Settings. Parameters are also split into inner section and outer sections and are simple color multipliers.

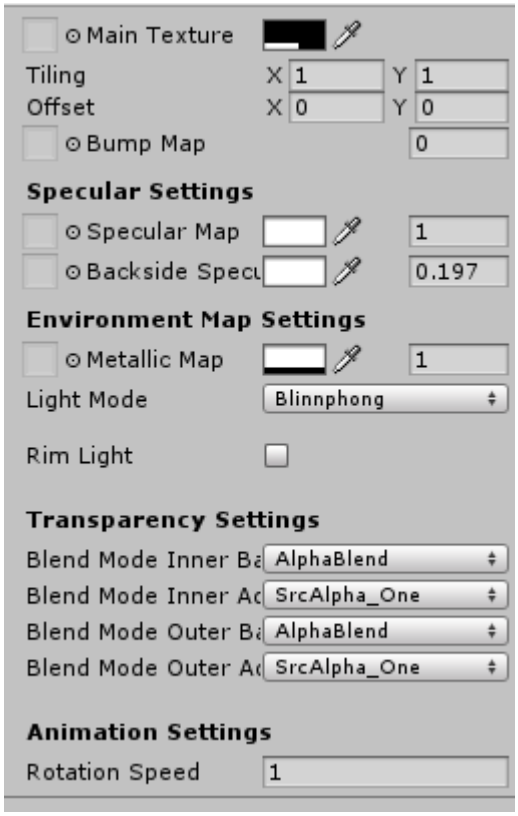
Keep in mind that this shader only makes sense if the rendering mode is not set to opaque.

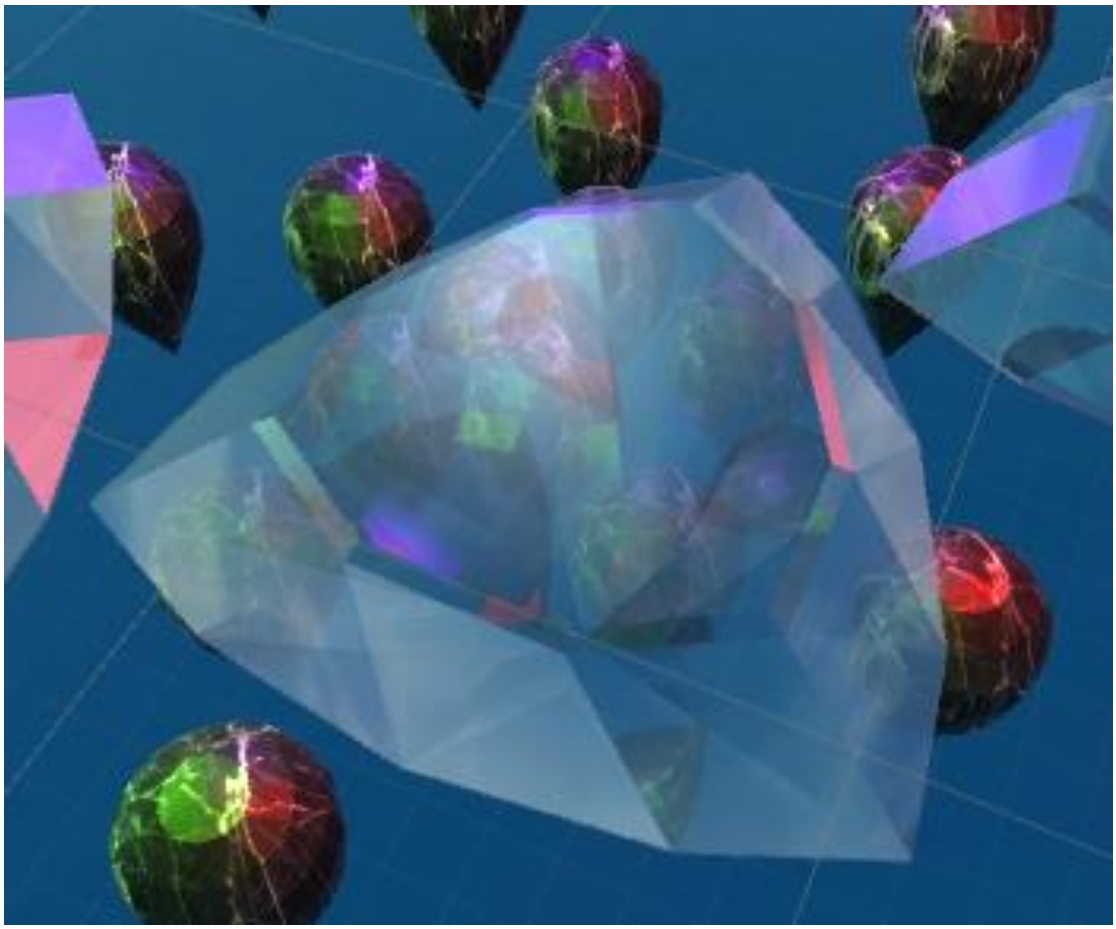


Shader 9: Complex

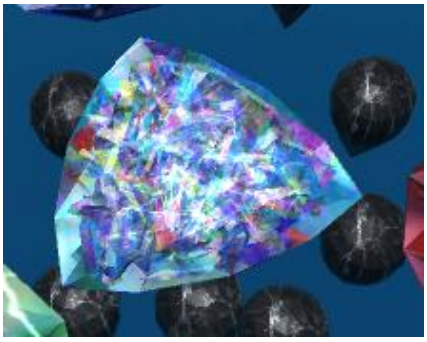
This is currently a highly experimental shader as it allows full control over the rendering calculations unlike surface shaders which are somewhat restrictive. It is a custom implementation with a custom lighting calculation. It is possible to change the blend modes of the individual render passes and it also supports rim light. The rim light also includes a simple gradient generator for the required ramp textures.

Available light modes are the normal lambert, phong and blinn-phong. It also is a preparation for the upcoming Unity v 2018.1





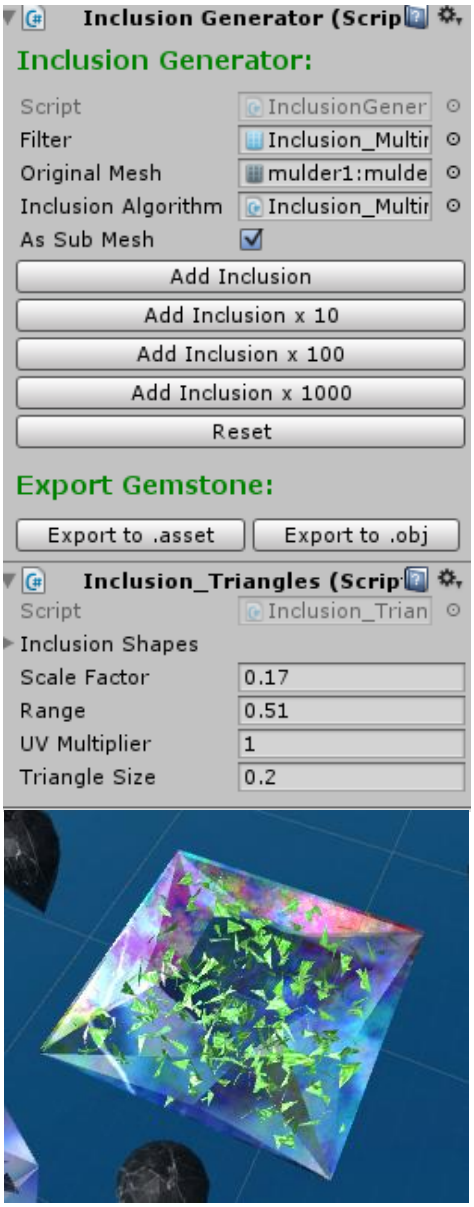
Inclusion Generator:



The gemstones in this package are actually absolute perfect gemstones. However such gemstones are very unnatural because most gemstones have errors in their crystalline structure. The inclusion generator attempts to create such “errors” by injecting meshes into the original gemstone mesh. It is almost possible to create real opalic behavior if the density of such inclusions is high enough.

How to use:

The Inclusion Generator has its own subfolder within the Advanced Gem Shader folder (Could be an own asset). Either simply drag in one of those prefabs or attach the Inclusion Generator to a game object.



The generator itself requires a MeshFilter component attached to it and an original mesh which can be any convex shape which is saved in the asset database. This shape is used to check if the injected inclusion is really inside the gemstone and not outside.

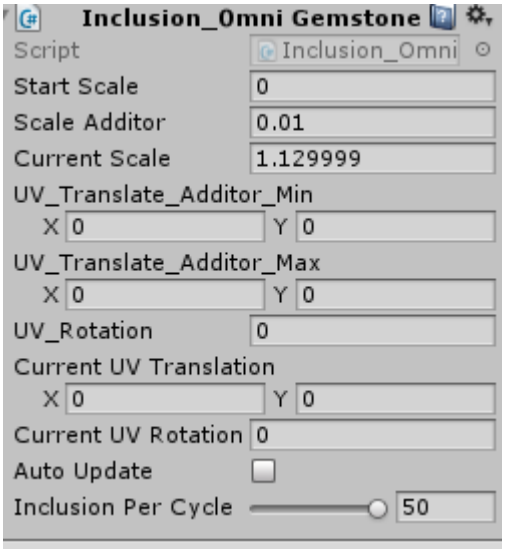
The generator also needs an inclusion algorithm which is also a component attached to the game object. It is recommended to keep the parameters very small so the inclusion is really inside the gemstone.

Whenever you hit one of the “Add Inclusion” buttons, the inclusion algorithm creates a mesh and injects it into the gemstone. However this process can fail if any vertex of the inclusion is outside of the gemstone or the maximum vertex count is reached.

Now it is also possible to separate the Inclusions from the main Gemstone. It will become a sub mesh so the final gemstone consists of two sub meshes. The resulting gemstone can therefore consist of multiple materials: One for the gemstone and one for the inclusions.

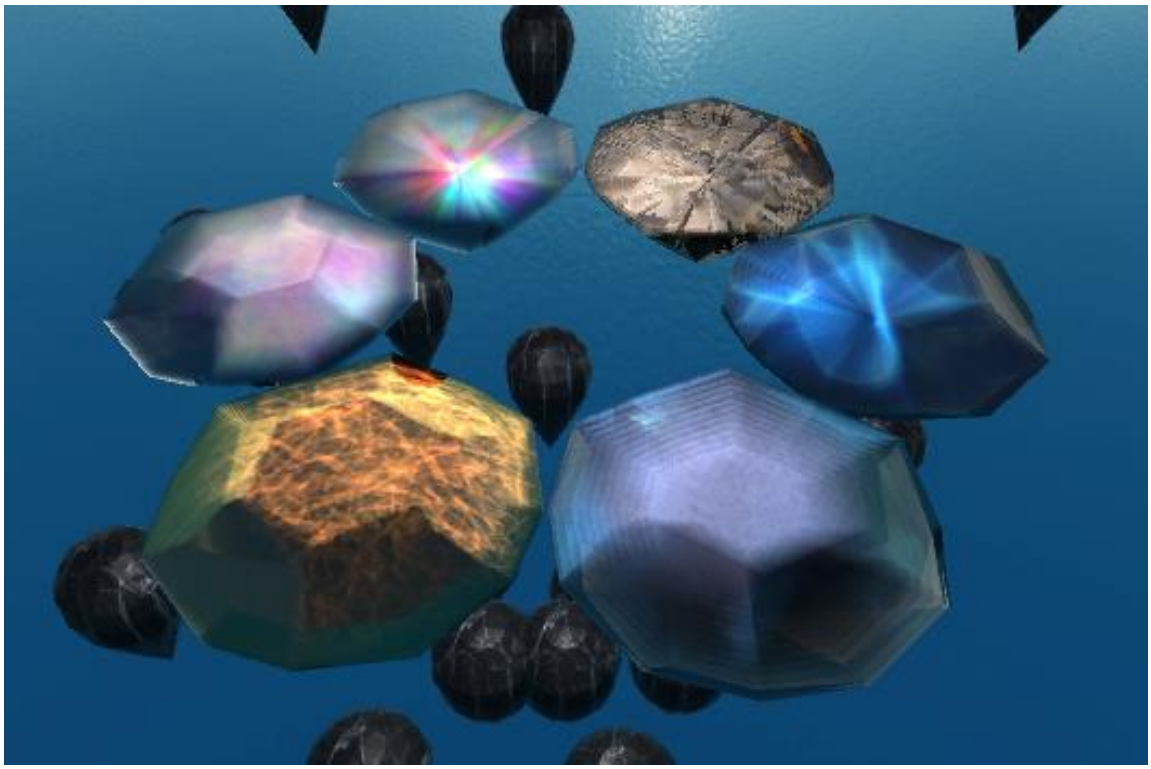
OmniGemstones:

OmniGemstones can be created using the Inclusion Generator. What it does is simply creating inclusions of the same shape as the gemstone itself but with a smaller scale than the previous inclusion. If combined with transparent shaders, gemstones with very unusual properties are possible.



The start scale is the initial scale of the inclusion and the scale additor determines the growth of each layer. The current scale is simply to display the current scale. It is also possible to manipulate the UV values of each layer.

You can also see the results immediately when auto update is checked but the maximum amount of layers are 50 to prevent slowdowns of unity. It also ignores inclusion check to increase speed. If you want more than 50, hit the normal create inclusion buttons to add layers.



Extending:

It is possible to create your own Inclusion algorithms by simply deriving from the **Inclusion_Base** class. The class itself has 4 virtual functions which are:

- **Preview():** For drawing debug stuff and Gizmos
- **CreateInclusion():** Is called when an inclusion should be created. Returns a mesh.
- **InclusionInvalid():** Is called when the mesh is not completely inside the gemstone.
- **Reset():** Called when hitting the “Reset Button”.

Pattern Generator:

This is a very simplistic version of a possibly future asset for the asset store. It simple arranges child game objects in a pattern described by an algorithm.

It includes 3 Patterns which are circle, box and spiral patterns.

Hints:

As you may notice, these shaders are working with cube maps. With different cubemaps, you can get a lot of interesting results. There is a very easy way to make one fast:

Select your texture, set texture type to advanced and change mapping to 6 Frame Layout or Mirrored Ball. Don’t forget to hit apply afterwards.

Last Notes:

If you have any questions, suggestions, bug reporting don’t hesitate to contact me. If you are going to sell a game which uses this asset, inform me because I may buy your game and play it ☺

Contact Information:

E-Mail: mhartl.mmt-b2013@fh-salzburg.ac.at

Homepage: http://lostinpixels.org/