Table of Contents
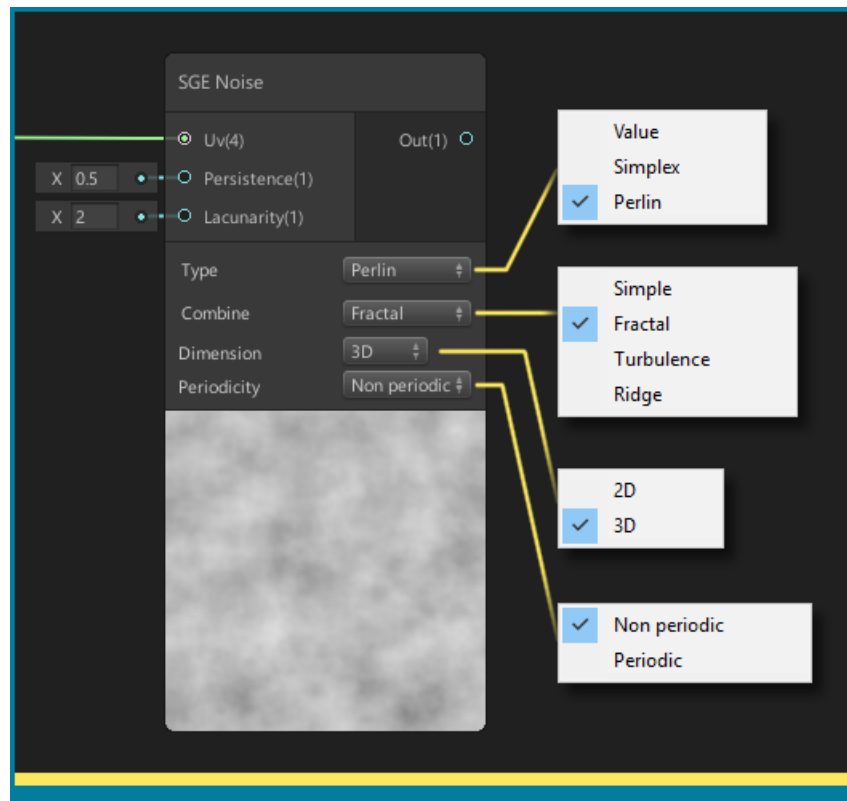
ShaderGraph Essentials features

ShaderGraph Essentials (SGE) is a unique plugin that adds several very important features to ShaderGraph.

Noises

A one-in-all noise node, with a powerful UI that let you iterates quickly in ShaderGraph. Features Simple value noise, Simplex, Perlin2D and Perlin3D. In addition every noise can be tillable (periodic) or not; and you can enable Fractal, Turbulence or Ridge combination in one clic !


Bake texture

A node that let you bake the output of any other node in a texture. This is extremely useful when you've got a group of node that is static according to the UV mapping of your mesh. You can optimize your graph by baking part of it in a texture, then using it instead of your baked graph. This node gives you the performance of a texture AND the iteration time of ShaderGraph, as you'll continue to work and generate your texture in ShaderGraph. Please note that this works in editor-only, not at runtime. If you want to bake textures at runtime, you might want to look into Unity's render texture (require a bit of C# code).

Bake texture from any node

Optimize noises

Optimize generated normals
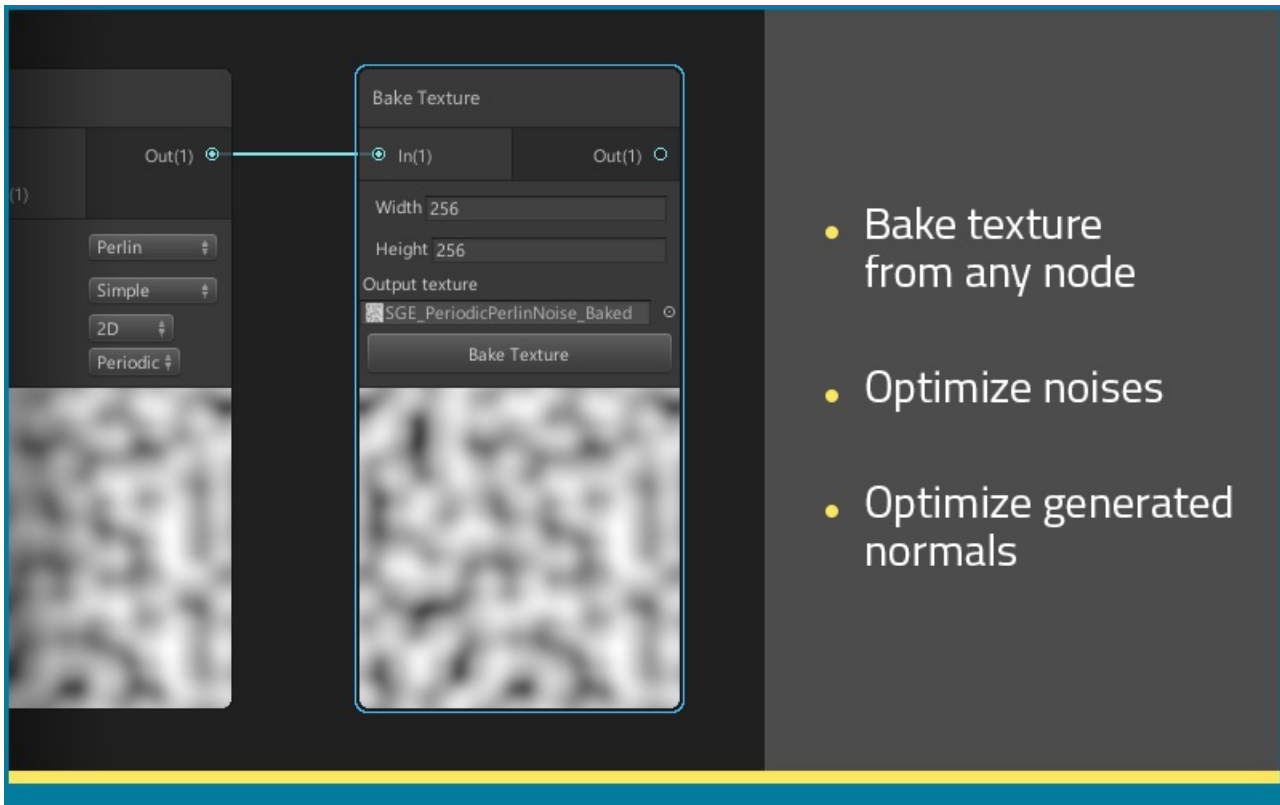
Extended master nodes settings

New master nodes that include extended settings over your shader graph behavior, including:

- RenderType / RenderQueue tags
- Blend mode
- Cull mode
- ZWrite / ZTest tags
- Custom editor
- Correct vertex position when using vertex displacement

Simple Lit master node

Custom blend mode

Custom ZWrite / ZTest

Custom editor

LWRP
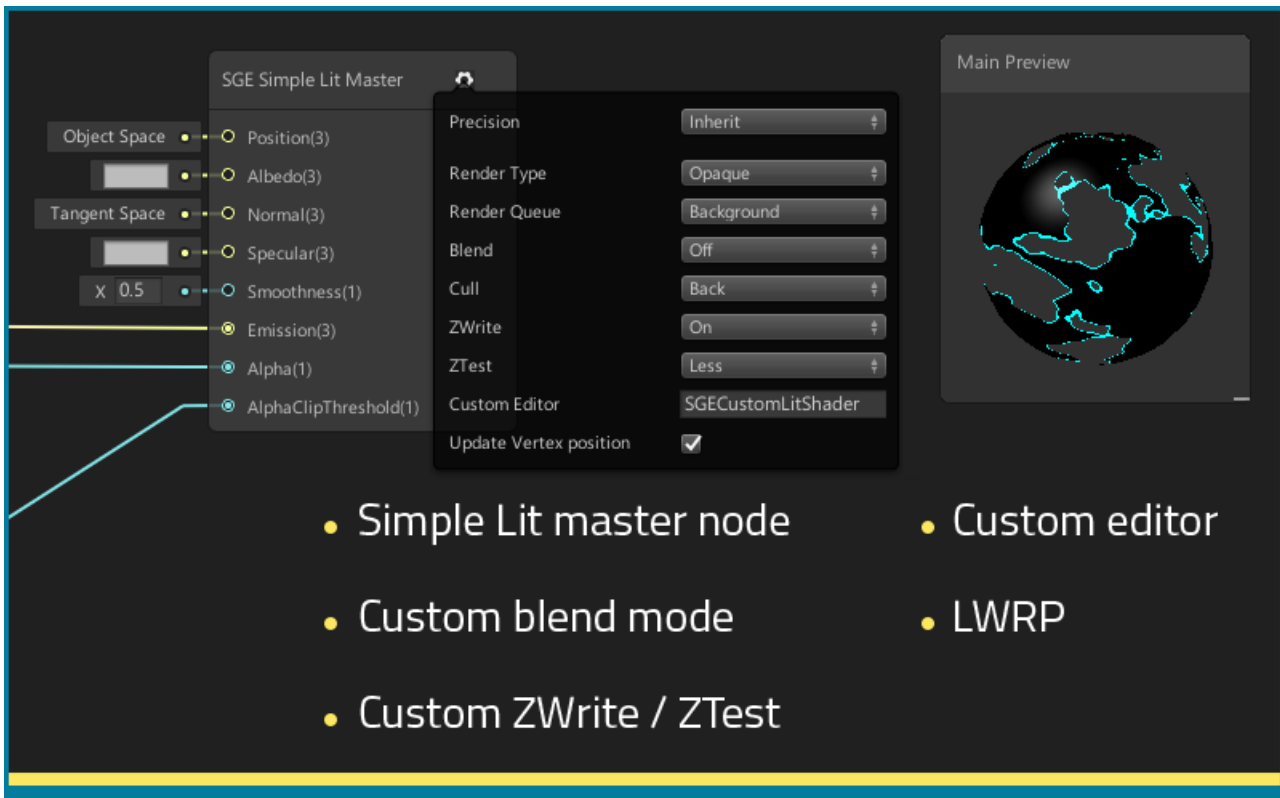
SimpleLit master node

An optimized lit shader that have better performance than the PBR master node (especially on mobile). It's the equivalent of the URP Simple Lit shader. Many assets or scenes don't need to support and execute the full PBR shader and this node gives you the power to use a much faster lit master node.

Custom and toon lit master node

A new master node that let you coe yourn own diffuse and specular lighting functions using HLSL. Unlike what you can do in vanilla ShaderGraph, this one reacts correctly to multiple lights and shadows. As an example I implemented a toon lighting node (available in demo scenes), that you're free to modify to suits your needs.
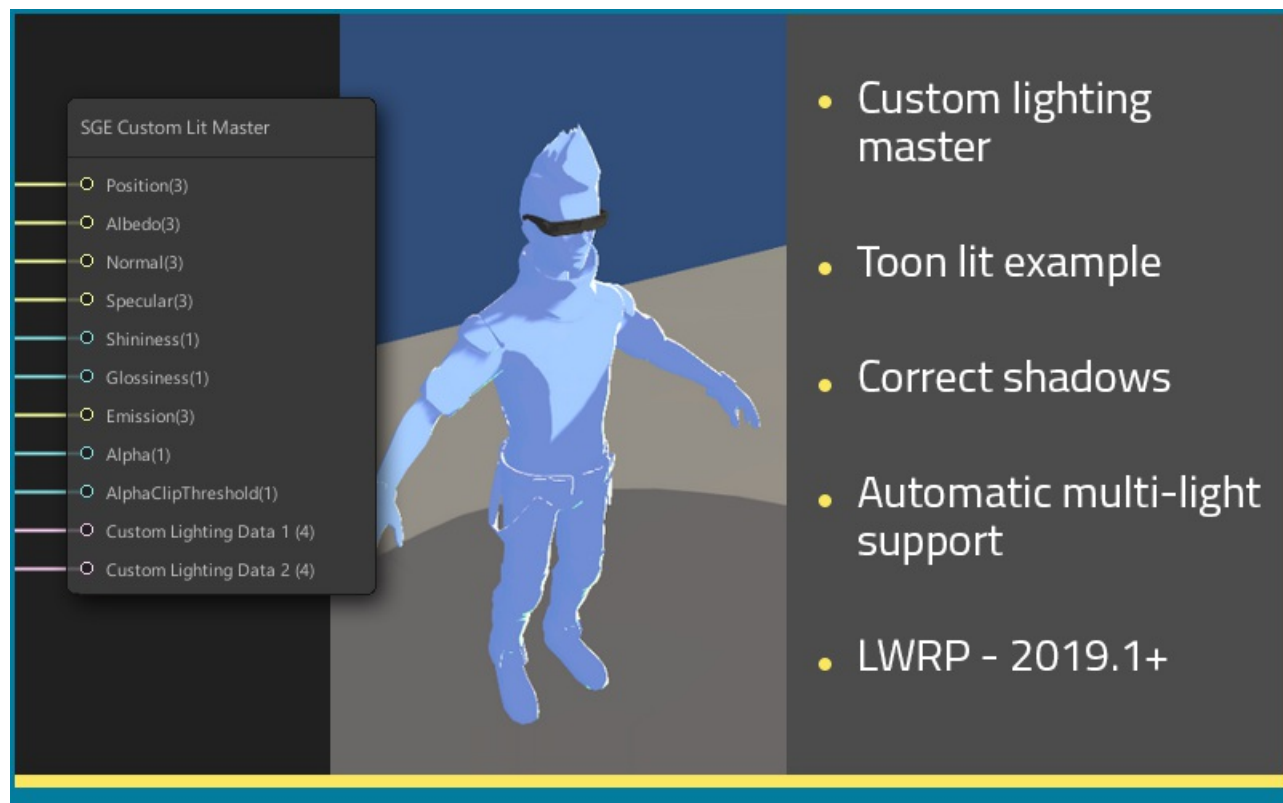
Water shader

A water shader made 100% in ShaderGraph, that feature both vertex displacement (CPU or GPU), lighting and foam (using depth texture). Demo scene included.

- Water shader 100% in Shadergraph

- CPU or GPU waves

- Depth-based foam effect

- LWRP

Getting Started

1. At the moment, ShaderGraph Essentials **MUST BE** in the default location, right under the Assets/Plugins folder because of how the master node templates work. If you would like this to change, please contact me.
2. When importing the plugin, import everything.
3. The getting started window will pop up (you can also find it under the menu item Tools -> ShaderGraph Essentials -> Getting Started). You can now import the URP plugin and test scene depending on your needs.
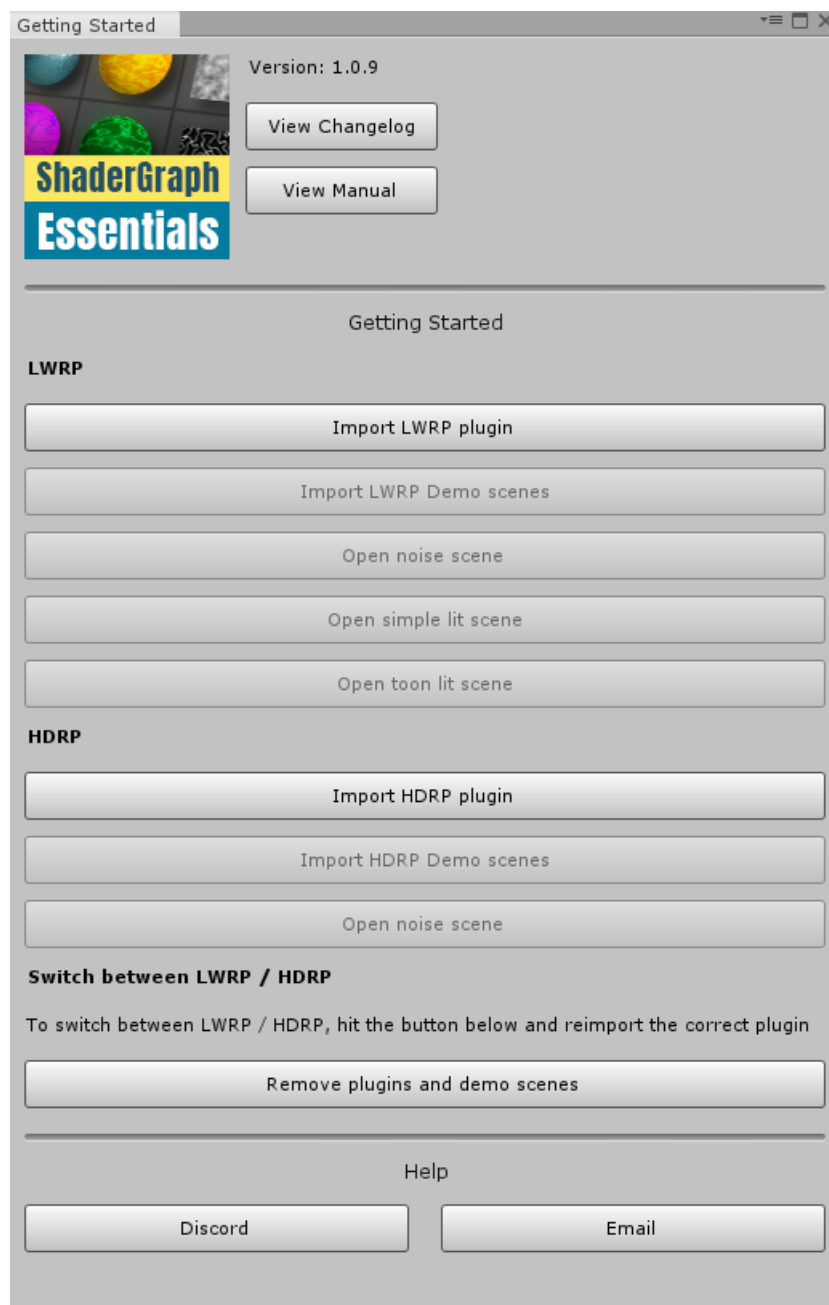
Noises

A one-in-all noise node, with a powerful UI that let you iterates quickly in ShaderGraph. In the *Demo_Base* folder, the Demo scene is full of different usage and cool demo of all supported noises!

Quick guide
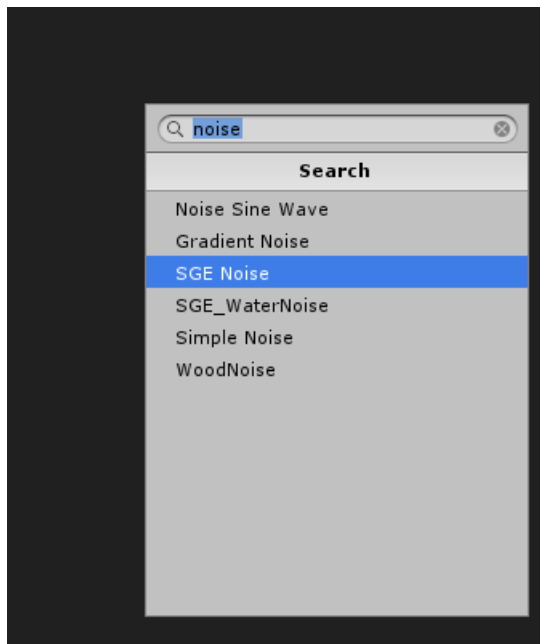
To use it, open a ShaderGraph asset and simply type in *Noise* in the node search, then look for *SGE Noise* (SGE means ShaderGraph Essentials). You can also find it under *Procedural/Noise*.

Parameters
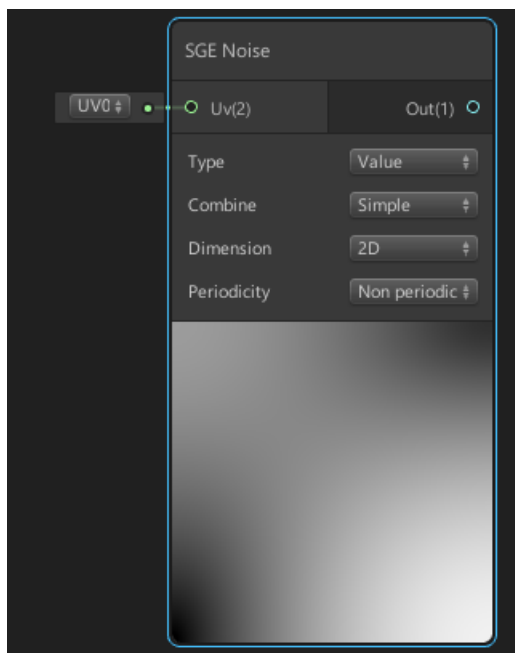
- **Inputs**: 2D Noises will use 2D input (e.g a UV channel). 3D Noises will use 3D input (like a UV channel + time, a 3D vector...).
- **Output**: All noises output a single float between 0 and 1.
- **Noise Type**: Value is the simplest (and cheapest), you also have Simplex and Perlin noises. Feel free to suggest other noise types that you'd like to see here !
- **Noise Combine**: Simple is the basic noise; Fractal / Turbulence / Ridge is a combination of 4 noises. When you change this parameter you'll notice, 2 other inputs appears

- **Persistence**: a weight for each subsequent noise value. Default is 0.5, which means the first noise is x0.5, the second x0.25, the third x0.125 ...etc.
- **Lacunarity**: a weight for each subsequent noise frequency. Default is 2.0, which means for each noise the frequency doubles.
- **Noise Dimension**: 2D / 3D. Please note that not all noise types support all dimensions right now; it would be possible but some noises get so expensive in 3D/4D that it's almost unusable.
- **Noise Periodicity**: Default is non periodic which means the noise will never repeat itself. Most of the time, it's what we want from a noise. Periodic is needed if you want to bake the noise in a texture so it can seamlessly tile (see Bake Texture).

Bake texture

Quick guide
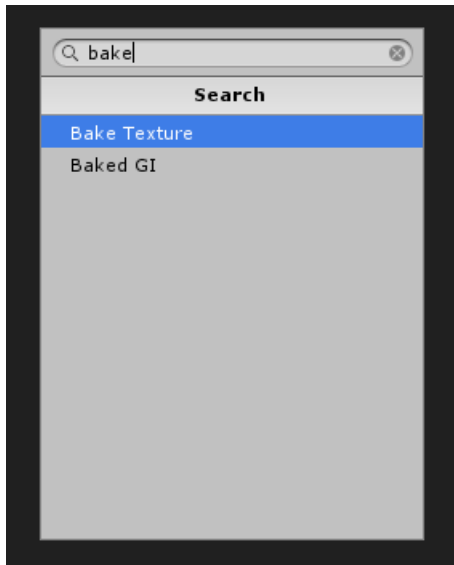
- Open a ShaderGraph asset and simply type *Bake* in the node search and look for *Bake Texture*. You can also find it under *Utility/*.
- Enter the wanted Width and Height, and hit bake!

Use case: How to optimize a noise node by baking the texture

Noises are very useful but can quickly get expensive in a shader, especially on mobile. If your noise is based on the mesh uv (which is very often the case), you can bake the noise into a precomputed texture and use this texture instead in your graph.
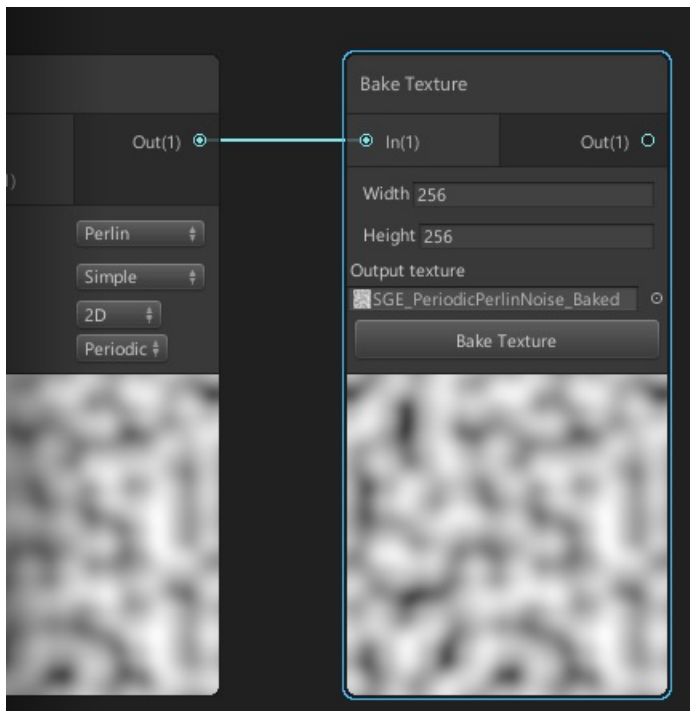
1. Change all your noise Periodicity to "Periodic".
2. Add a "Bake Texture" node to the output of the node you want to bake (usually a noise or combination of noise, but it can anything !).
3. Tweak the Width / Height of the Output texture.
4. Click on Bake Texture button.

5. The first time, it'll generate a texture directly in the *Assets/* folder. Later on, it'll override the texture content every time you hit "Bake Texture".
6. Go it your Graph, add a Texture2D node to replace the Noise (or combination of noise) and uses the baked texture.
7. You can keep your noise "generator" (e.g the noise that output in the BakeTexture Node) in the same graph, it won't have a performance impact if it's node linked to the Master node. This way you can have a very quick iteration time: when you bake the texture, everything will be updated automatically !

Extended master node settings

Quick guide

All master nodes from ShaderGraph Essentials (SGE) feature extented settings compared to the ShaderGraph ones. You can find those in SGE SimpleLit, SGE CustomLit (including toon lit) and SGE Unlit master nodes.\ At the moment there is no *SGE PBR master node* (e.g the PBR master node with more settings), but feel free to contact me as it's something that could definitely be added.

Settings
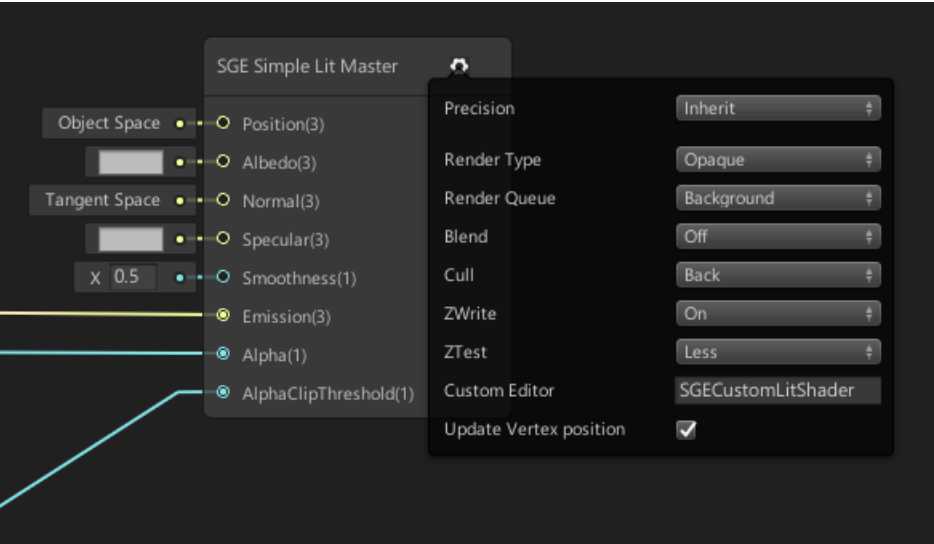
Most settings correspond to settings that are usually defined directly in shaders. As such, most settings have documentation on the Unity website (an example would be this).

NAME	VALUES	DESCRIPTION
RenderType	Opaque Transparent Transparent Cutout Background Overlay	Categorize how the shader will be used in rendering.
RenderQueue	Background Geometry Transparent Overlay Alpha test	Define in which order objects are going to be drawn.
Blend	Off Alpha Multiply Additive Premultiply	Define how the object will be composited with what's already been drawn. If needed, more options could easily be added.
Cull	Back Front Off	Define if and which vertex should be culled (front/back facing).

NAME	VALUES	DESCRIPTION
ZWrite	On Off	Should the object write to the Z-buffer (also called depth buffer).
ZTest	Less Greater L Equal G Equal Equal Not Equal Always	Define in which condition the ZTest should pass.
Custom Editor	String	The string should be the valid class name (**including the namespace**) of the C# editor class. Something like *MyRootNamespace.MySubNamespace.MyCustomShaderEditor*
Update vertex position	True False	True means the position you get in the pixel-shader part of the graph will be the updated position (according the any displacement you could have done in the "Position" master node input). False means that any position you get in the pixel-shader part of the graph is the pre-displacement position (default in other ShaderGraphs in Unity). More info and examples in the Water shader example.

Simple lit master node

Quick start

Create a new simple lit master graph

Right clic in the Project Explorer, then **Create -> Shader -> SGE Simple Lit Graph**.

Create a simple lit master node in an existing graph

In an existing graph, create a new simple lit master node by looking in *Master*.\ You can then right clic to set it as *Active* and delete the previous master node.

Parameters

Parameters are the same as Unity's SimpleLit build-in shader. It features a different, less-realistic, lighting model than the PBR node.\ It is also much more performant than the PBR master node, especially on mobile platforms.
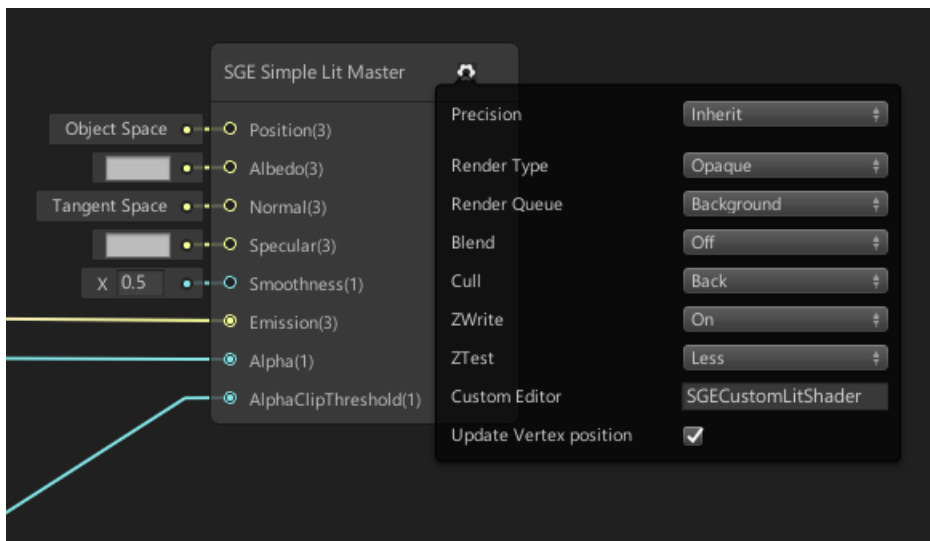
Custom and toon lit master nodes

Note that in the *Demo_URP/Shaders* folder, the SGE_ToonLit shader is a good demonstration of this feature. If you're interested in coding your own custom lighting, I will assume you know a bit of HLSL. The typical use case for this feature would be if you had a custom lighting function in Unity shaders and you would like to port it to ShaderGraph.
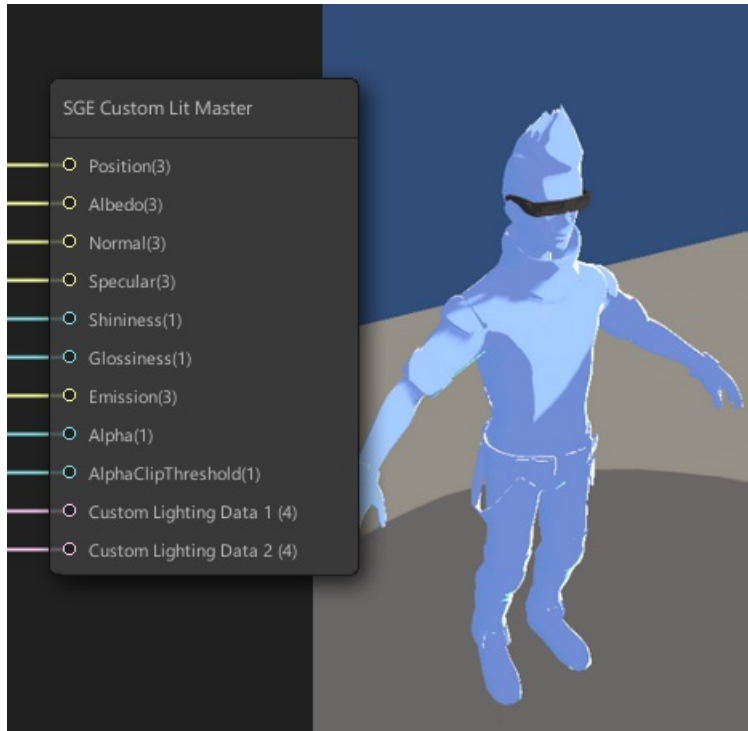
Quick guide

1. In project view, right clic **Create -> Shaders -> SGE Custom Lit Graph**.
2. This will create a graph, which already contains a custom lighting master node and another node.
3. The pre-existing node is a Custom Function Node that points to a shader. You want to duplicate it and have the Custom Function Node points to your new shader. Then open the shader.
4. It contains a bunch of helpful comments, and 3 functions:
   1. *SGE_DiffuseLightingCustom* which you can change, it's the diffuse lighting function. Just don't change the name or parameters, or it won't work.
   2. *SGE_SpecularLightingCustom* which you can change, it's the specular lighting function. Just don't change the name or parameters, or it won't work.
   3. *Passthrough_float* which you shouldn't change. It's just a dummy because Unity requires a function to be integrated in the graph, otherwise it's optimized out.

Toon lighting
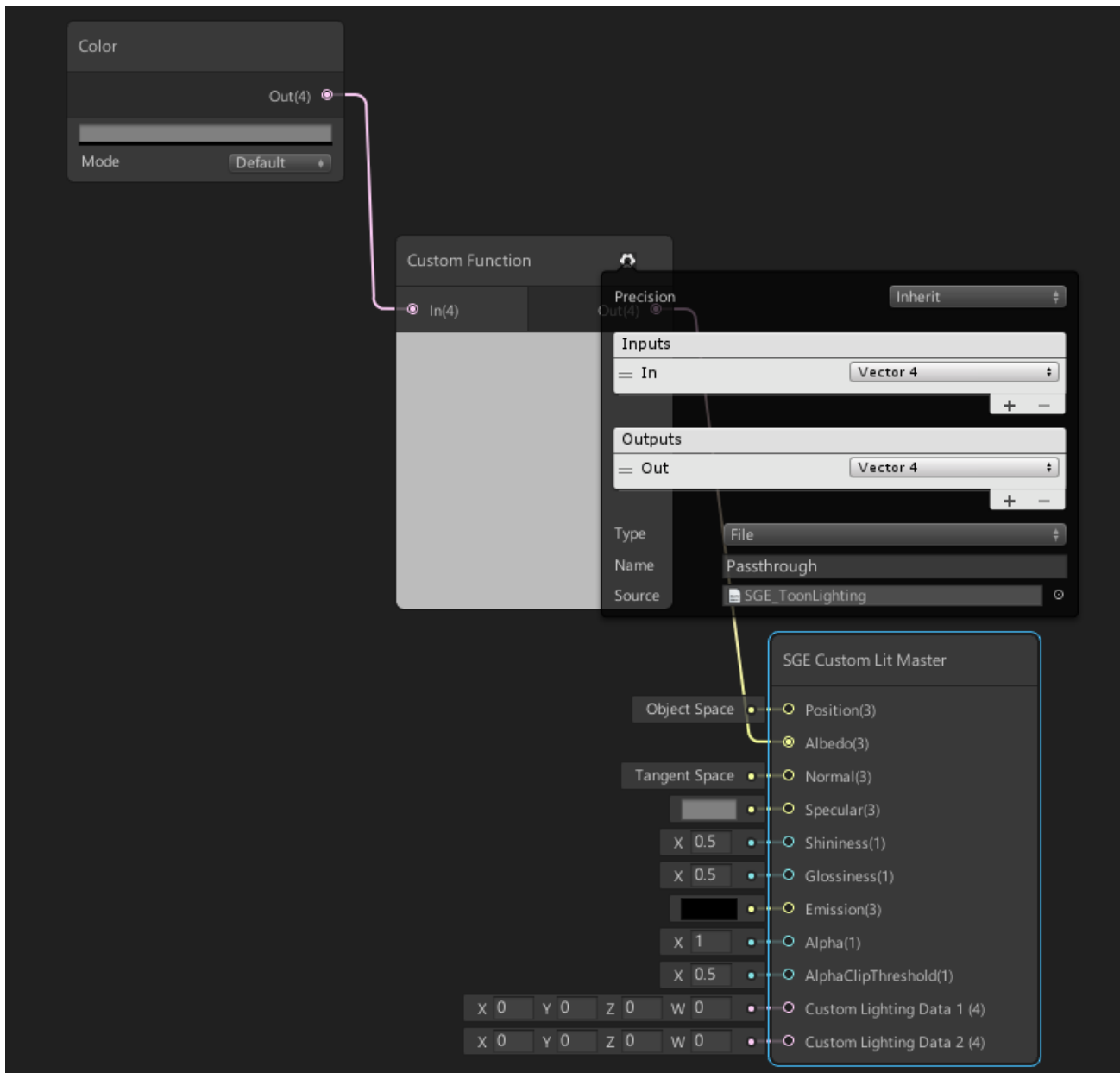
The toon lighting implemented in this lighting is simply a showcase of the custom lighting function. Feel free to use it how you want, it's definitely performant and production-ready! However you might want to change it to your tastes; maybe add a texture ramp, remove the rim light ...etc, which is totally doable if you're familiar with HLSL.\ Feel free to contact me if you would like this shader to support additionnal feature!
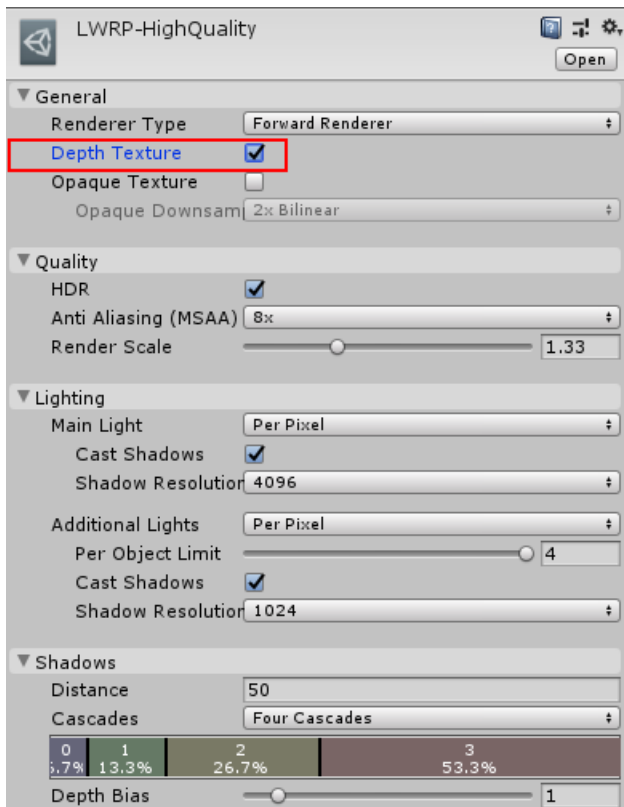
Water shader

Depth texture

Context

(time of writing, september 2019)

Almost nothing in theses water shader or script is special. You could almost do it directly yourself in classic ShaderGraph and it would have almost worked.\ Let me explain. Unity's shadergraph contains a bug (or missing feature, depending on how you view it) which makes effect based on position (world position, object position, screen position, any kind of position) impossible when using the "Position" input to move vertices.\ The reason is, when you move the position through the *Position* input and then use the position again in other parts of the graph, Unity gives you the pre-displacement position. And there are no workaround.\ This bug was reported several times, an example here. I noticed it too when working on a water shader in ShaderGraph, decided to fix it and provide a water shader example as bonus value.\ This fix will definitely work for anything, not only a water shader. Once you have ShaderGraph Essentials (SGE), the fix is trivial. You only have to active the *Update vertex position* in any SGE master node.
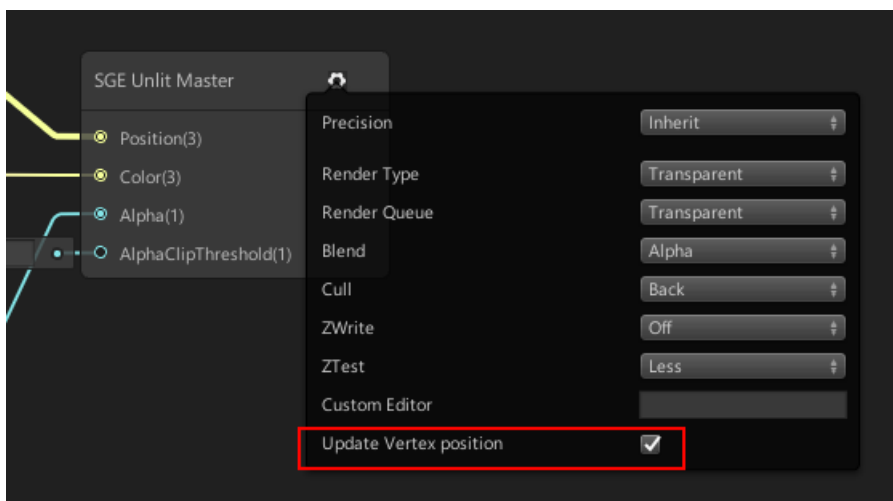
You might ask why it's not always enabled ?\ Well, one could argue that it could add a very small performance overhead. I profiled on mobile and it's unnoticeable. But because it's theorically here, I left the parameter optional.

Wave movement: CPU or GPU ?

The provided shaders support a wave movement along the up axis. It uses up to 3 sine-waves so it doesn't look repetitive and it is still reasonable performance-wise. You could definitely add more complex and realistic computations to suits your needs.

ShaderGraph Essentials provides to different shaders to move the vertices: a CPU and a GPU implementation. Using one or the other is very important choice and I am here to help you make the right one.\ You don't need to be a programmer to make the correct choice for your project, but if you do have a programmer I suggest you talk to him about this.

CPU vs GPU what's the difference?

CPU is your processor, where most of your game code run. It can run highly flexible code and do a lot of different things.\ GPU is your graphics card, where your shaders run. It is really fast at doing a very specific job (somewhat repetitive computations in parrallele).\ Moving each water vertice is something that can be costly in performance and you have the choice to do this job on the CPU or the GPU.

On the CPU

Pros:

- your game code can easily know the wave height at any point, and so support buoyancy (floating objects)

Cons:

- much slower than on the GPU

On the GPU

Pros:

- it's very fast

Cons:

- you can't really have buoyancy (floating objects in the water)

Where can I select one or the other ?

There are both examples in the *ShaderGraphEssentials_Showcase_Water* scene in the *Demo_LWRP* folder.\ Alternatively, there are two prefabs, *SGE_Water_CPU* and *SGE_Water_GPU*. Basically:

- There are two different shader graphs: SGE_LowPolyWaterGPU and SGE_LowPolyWaterCPU
- With the CPU version, it also needs a script to move vertices in C# code. The script is called *TesselatedWater*, and is meant to be put directly on the water object.
- With the GPU version, you will tweak displacement settings directly on the material.

Settings
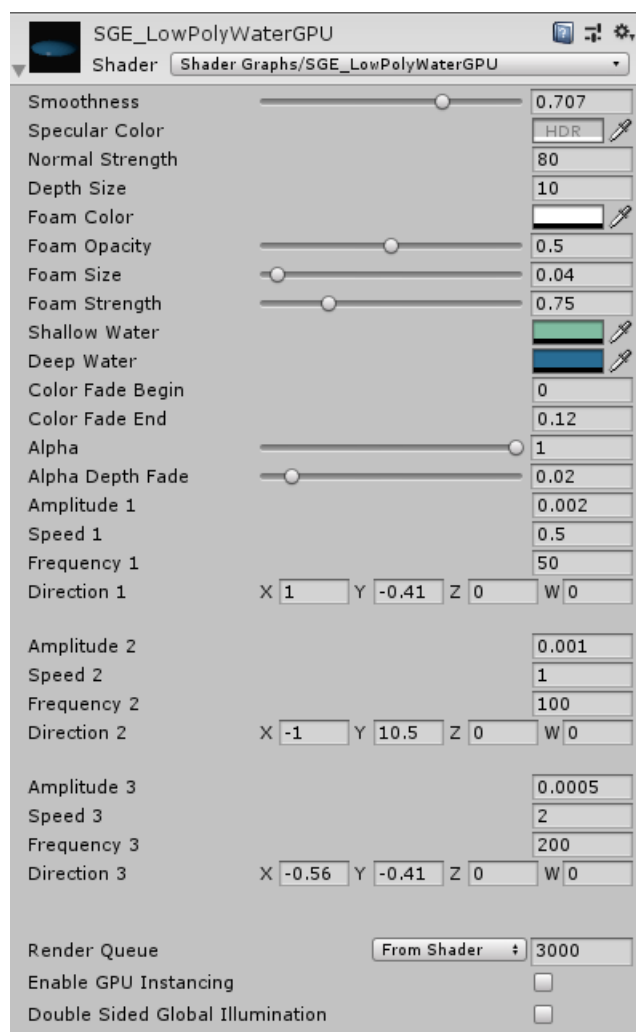
Wave movement

If you chose the CPU version, those settings will be on the Tesselated_Water script.\ If you chose the GPU version, those settings will be on the material.

Those settings are per-wave, all 3 waves are added on top of each other. What you generally want to do is have some waves with big amplitude and low speed, and some waves with lower amplitude and fast speed. This is what I've done in demo scene and it really helps breaking the sine wave repetition.

NAME	TYPE	DESCRIPTION
Amplitude	Float	Height of the wave (in object space). Should be > 0.
Speed	Float	Speed of the wave (arbitrary units). Should be > 0.
Frequency	Float	Length of the wave (arbitrary units). Should be > 0.
Direction	Vector2	Direction vector of the wave. Only the first 2 components matter (it corresponds to X and Z).

Depth and foam

NAME	TYPE	DESCRIPTION
Depth size	Float	In meters, represent the Ocean max depth. It doesn't directly impact visuals, but is used so all other parameters are relative (in %) instead of absolute. For this reason, I suggest that you enter a value here early on, then left it untouched. Because a lot of parameters depend on this one, you will have to re-tweak everything after touching the depth size.

NAME	TYPE	DESCRIPTION
Foam Color	Color	Color of the foam.
Foam Opacity	Float	Opacity of the foam, between 0 and 1.
Foam Size	Float	How big is the foam (relative to depth size).
Foam Strength	Float	Multiplier on the final foam color. A very high value will start to remove the foam fade-out near objects.

Color

NAME	TYPE	DESCRIPTION
Shallow Water	Color	Color of the water near objects.
Deep water	Color	Color of the water far from objects.
Color Fade Begin	Float	How close to the shore will the color begin to fade from Shallow to Deep water color. Should be inferior to *Color fade end*.
Color Fade End	Float	How close to the shore will the color end to fade from Shallow to Deep water color. Should be superior to *Color fade begin*.

Alpha

NAME	TYPE	DESCRIPTION
Alpha	Float	Overall opacity of the water.
Alpha Depth Fade	Float	Allow the water to smoothly fade when close to objects (based on depth texture). A value of 0 means no fade.

Lighting

The water shader is based on the SGE SimpleLit master node. If you would need to change it to a PBR master node, please contact me.

NAME	TYPE	DESCRIPTION
Smoothness	Float	Define how the specular will look like.
Specular color	Float	Color and intensity of specular highlights.
Normal strength	Float	Specific to the GPU implementation. Required because with the GPU normals are computed in the pixel shader.

Changelog

1.1.6

Submitted August 7th 2020.

- This version only affects 2019.2 / 2019.3 / 2019.4 (as all changes were already in the 2020.1 version).
- Completely changed how the plugin integrates with ShaderGraph. It is much more robust and future proof!
- Fixed possible NaNs error when using 3D perlin Ridge noise node.
- Fixed all scripts warnings.

1.1.5

Submitted July 24th 2020.

- Added 2020.1 support.
- 2020.1 only: Completely changed how the plugin integrates with ShaderGraph. It is much more robust, and will be backported to 2019.x later if no issues are found.
- 2020.1 only: Fixed possible NaNs error when using 3D perlin Ridge noise node. Will be backported later.
- 2020.1 only: Renamed all "LWRP" instances to "URP". Will be backported to 2019.3 later.
- 2020.1 only: Fixed all scripts warnings.

1.1.4

Submitted March 8th 2020.

- Only impact 2019.3: Now supports and requires URP / HDRP 7.2.1.

1.1.3

Submitted February 20th 2020.

- Only impact 2019.3: Fixed shader compilation error when using SGE master nodes on Android VR.

1.1.2

Submitted January 29th 2020.

- Fixed a compilation error starting from 2019.3.0f6 (with URP 7.1.8).
- Improving SGE registration. This should solve errors when updating URP or Unity and getting "is inaccessible error". If you still have an issue similar to this, please contact me on the discord server.
- Better support if you have ShaderGraph package locally in your asset folder.
- Typo fix: "Periode" to "Period".

1.1.1

Submitted November 14th 2019.

- Added support for Unity 2019.3 (2019.3.0b11 and above).

1.1.0

Submitted September 18th 2019.

- Added new setting in all SGE master nodes: "Update Vertex Position".
- Added new water demo scene and shaders, with vertex displacement and depth-based foam to showcase the new setting
- Re-did the entire ShaderGraph Essentials asset store page, marketing visuals and added extensive online doc.

1.0.9

Submitted August 3rd 2019.

- Only impacts 2019.2; if you're on 2018 or 2019.1 it's not useful to update.
- Add 2019.2 support (require ShaderGraph / SRP 6.9.1 and not 6.9.0 which contains a bug with custom functions and subgraphs). If you update from 2019.1, please delete the ShaderGraph Essentials folder and re-download the new one after upgrade to make sure there're no leftover files.
- Removed some demo shaders that were unused and fixed minor bugs in other demo shaders

1.0.8

Submitted on July 22th 2019.

- If you're on Unity 2019.1.3 or newer and using LWRP, then you need to update to LWRP/ShaderGraph 5.16.1 when getting this version.
- Fixed a bug in SimpleLit / CustomLit shaders. The bug was giving errors in editor, but the shader was still working in editor and in builds.
- Fixed the version number in Getting Started window being wrong.

1.0.7

Submitted on July 15th 2019.

- Fixed a bug with the default custom lighting graph, referencing a wrong path (hlsl file).
- Fixed a minor bug that was throwing an GUI error the first time the Getting Started window was opened
- Fixed a bug that blocked the Getting Started window to import the HDRP package or demo scenes.

1.0.6

Submitted on June 5th 2019.

- Introduced the "Getting Started" window. It's useful to both existing and new users!
- Restructured how file are organized in the plugin. Also moved the entire plugin from Assets/ShaderGraphEssentials to Assets/Plugins/ShaderGraphEssentials. This has been requested by many users as to reduced the visual clutter of the root folder (as the plugin can't be moved at the moment!).
- Move the menu items from SGE to Tools/ShaderGraph Essentials for the same reasons.

1.0.5

Submitted on May 4th 2019.

- Updated SimpleLit master node to follow what Unity's been doing on LWRP/SimpleLit shader.
  - Removed Glossiness node (it wasn't used)
  - Renamed Shininess to Smoothness
- Added a new fix for "Internal Errors" that some users were getting when opening ShaderGraphEssentials the first time (and sometimes on later open too).

1.0.4

Submitted on April 24th 2019.

- 2019 LWRP version now requires ShaderGraph 5.13.0 or newer (if you're on 2018.3 or on HDRP you're fine)
  - If you don't want to update to 5.13.0, please contact me.

- Added new Custom Lighting master node (LWRP - 2019.1 only)
- Added new Toon Lighting master node example (LWRP - 2019.1 only)
- In all master nodes, added a "CustomEditor" setting so you can specify custom editor scripts like in regular shaders.

1.0.3

Submitted on April 19th 2019.

- Added support up to ShaderGraph 5.13.0.

1.0.2

Submitted on April 17th 2019.

1.0.2a
- Uploaded a new 2019.1 compatible version.
- The 2018.3 version is unchanged.

1.0.2b
- Fixed an issue with permission which was creating errors on load with a 2019.1 project.

1.0.2c
- Turns out the previous fix was working on windows only; so this version add the permission fix on mac.

1.0.1

Submitted on March 26th 2019.

- Fixed a bug in SGESimpleLit and SGEUnlit master nodes that made it impossible to work on Mac/Unix systems.

1.0.0

Submitted on March 16th 2019.

- Initial release

Troubleshooting

In Unity 2018.3, ShaderGraph is still in experimental preview; which means it's not stable, it often throws errors that can or can't be fixed, and it can crash. In 2019.x it's a bit more stable, but ShaderGraph is still pretty new.

For the same reason, the API is changing a lot in every release; I've tested as much as I can, but if you update Unity / SRP / ShaderGraph package and ShaderGraph Essentials doesn't work anymore, please try the following:

- in HDRP if you've got errors in the console but the demo scene is working (nothing's pink), then you're fine. Unity is wrongly throwing errors, if it bothers you, you can remove the Demo.
- make sure you're using a version that ShaderGraph Essentials supports (Unity 2018.3.x or newer and SRP / latest ShaderGraph package)
- try quickly in an empty project with your Unity / package version and import ShaderGraph Essentials to identify if it comes from ShaderGraph Essentials
- if you have a pink object / preview, try to open the ShaderGraph, save it and close it.
    - Note: if this fixes your problem and you need to do that on multiple shaders, you can also select them then right clic - > Reimport. It will sometimes work too !

- if this doesn't work or you've got errors in your console, try closing / reopening Unity.
- if nothing works or you aren't sure what to do, don't hesitate to contact me and I'll help you ! Please describe the problem as clearly as you can, screenshots, stacktrace, anything can help !

How-to

How to update a Unity major version (e.g 2019.2 -> 2019.3) and the plugin without problems?

Because the plugin is extremely tied to ShaderGraph, I have uploaded a different version of the plugin for each supported major version of Unity.

When you update a major version of Unity (e.g 2019.2 -> 2019.3), Unity doesn't automatically fetch the "correct" version of the plugin, because Unity never override what's in your Assets folder. What's worse is that at the moment Unity won't download the correct plugin version if you already downloaded the same plugin version from a previous Unity version. It's a bug in Unity, and they are aware of it. To bypass this bug, here are the steps you need to follow:

- Update your project to the version of Unity that you want. You will probably have console errors, that's ok for now.
- In Windows explorer, go to the folder **"%appdata%\Unity\Asset Store-5.x\PH GRAPHICS\Editor ExtensionsEffects"** which is where Unity keep its cache of all assets from the asset store.
- If you are on Mac, contact me and we will figure it out together.
- **Delete the package in this folder**.
- Go on the asset store from Unity
- Re-download and re-import the plugin, it will be the correct version now.
- If you have pink materials, please try to select all ShaderGraphs in your project and right clic-> Reimport!

Performances

All nodes in ShaderGraph Essentials have been tested on

- Low and High-end PC
- Mobile (Android / OpenglES 3.0+)
- VR (Oculus)

The SGE Unlit and SGE Simple are obviously very fast because it directly rely on Unity's implementation.

All noise implementations are among the fastest it possibly can. That being said, **using noises in a shader can be costly** so here's a few tips about performance:

- Using *Combine: Simple* is **4x faster** than the other options. The other options are all very close in term of performance. The noise shader implementation doesn't rely on any texture; it means you won't have any CPU / GPU bandwidth or memory problems. The only thing is to be careful about the pure GPU ALU cost (GPU pure processing power), **which is directly related to the number of pixel your object occupies on screen**. It means if it's a very small object on screen, you can get away with much more complexe noises; whereas if it's very big on screen and always in your face, you should think twice about what you do in the shader. Please note that if you use noises (or any node) to change the "Position" in the master node, then it goes in vertex shader and the cost is now related to the vertex count of your mesh; not the pixel it occupies on screen.
- Periodic / Non periodic changes almost nothing about performances.
- 2D / 3D impacts performance quite a lot. 3D is often several times more expensive that the 2D ones. If you have performance problems with the nodes, don't hesitate to contact me and I will help you and give you advice.
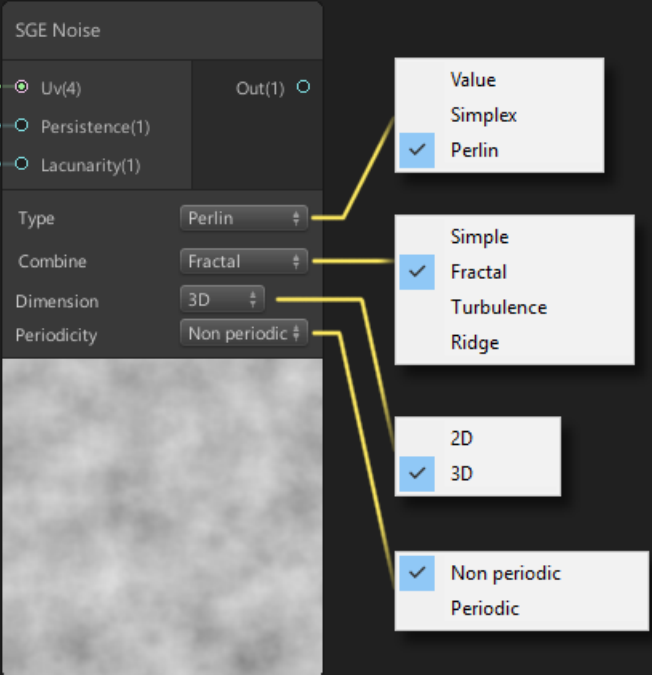
Contact

Don't hesitate to contact me about bugs, feature requests, or any question you might have before buying ShaderGraph Essentials.

The fastest way to contact me is through the ShaderGraph Essentials discord server. You can also send me a private message from there.
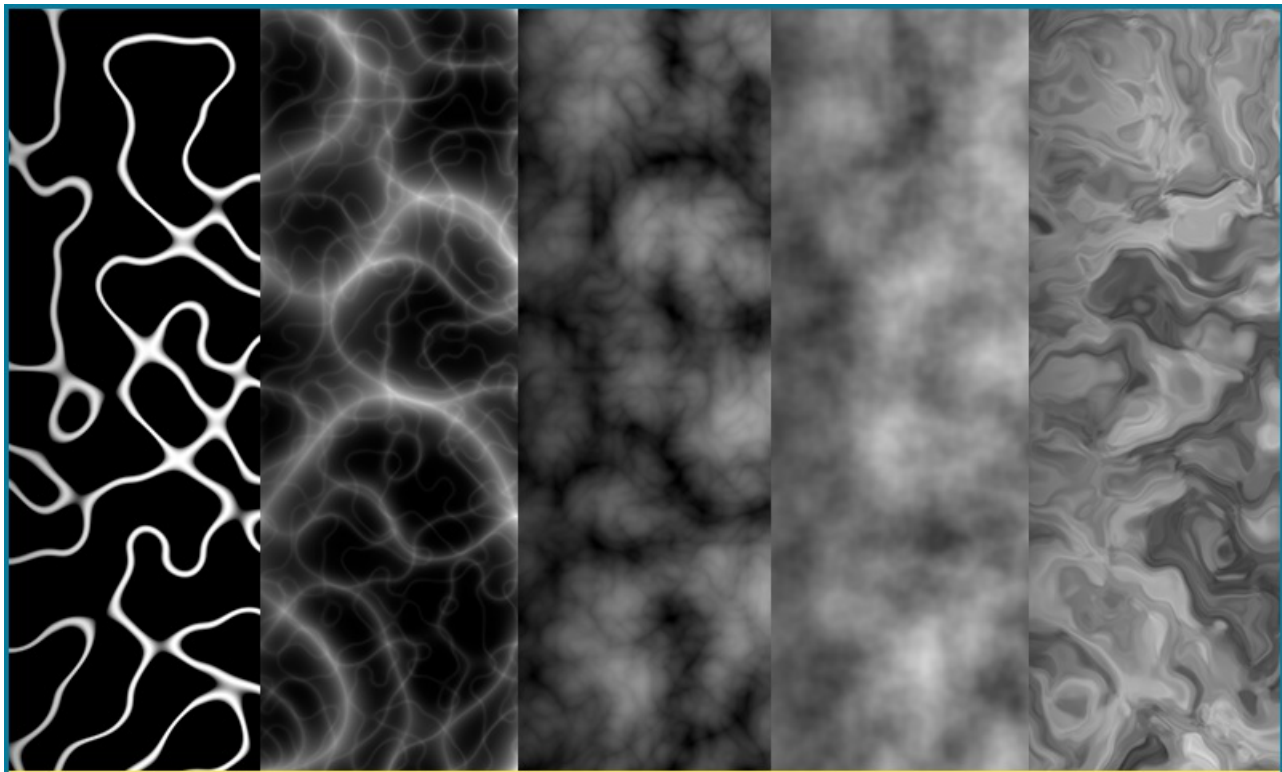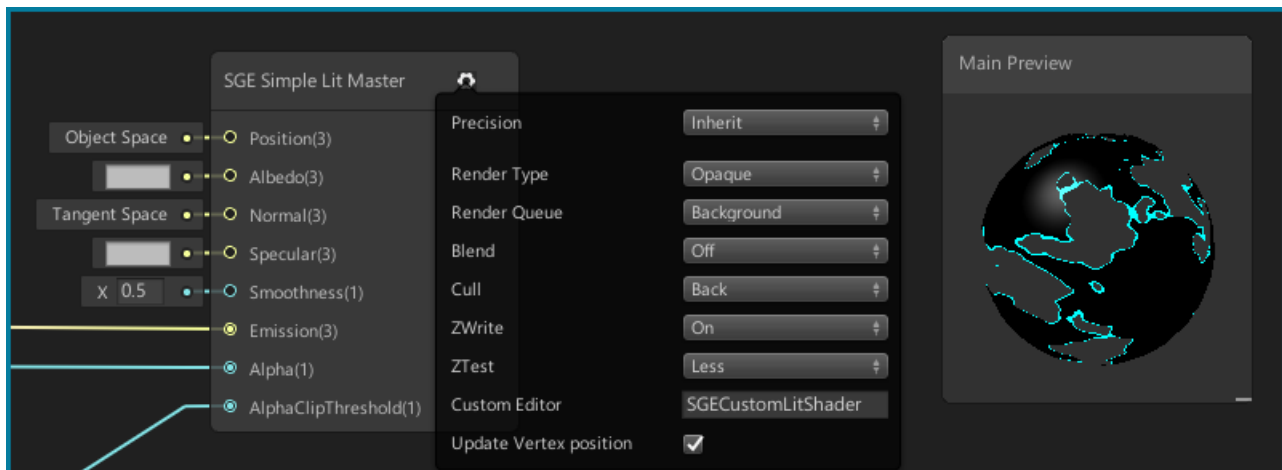
You can also contact me by email:

- about ShaderGraph Essentials at ph.graphics.unity@gmail.com
- for other business inquiries at info@phbarralis.com
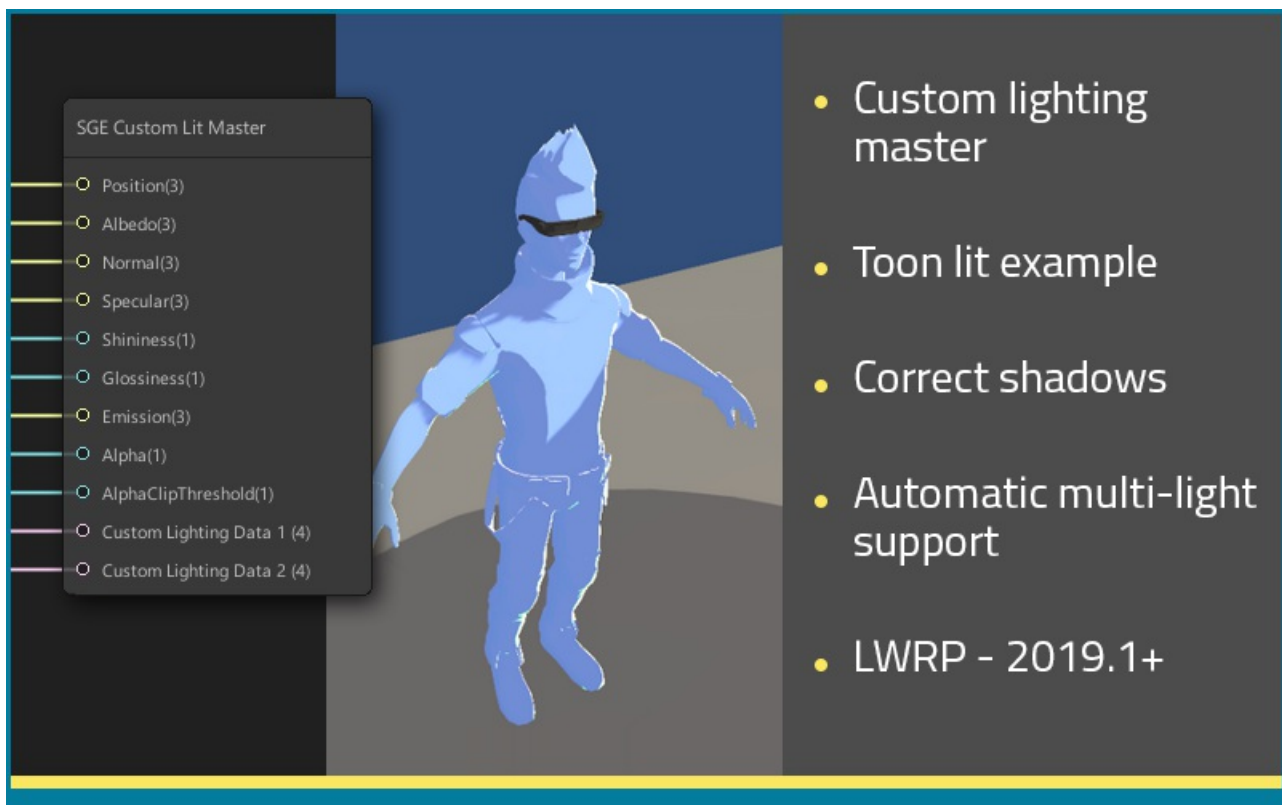
ShaderGraph Essentials gallery

- All-in-one noise
- 30+ variations
- 2-click changes
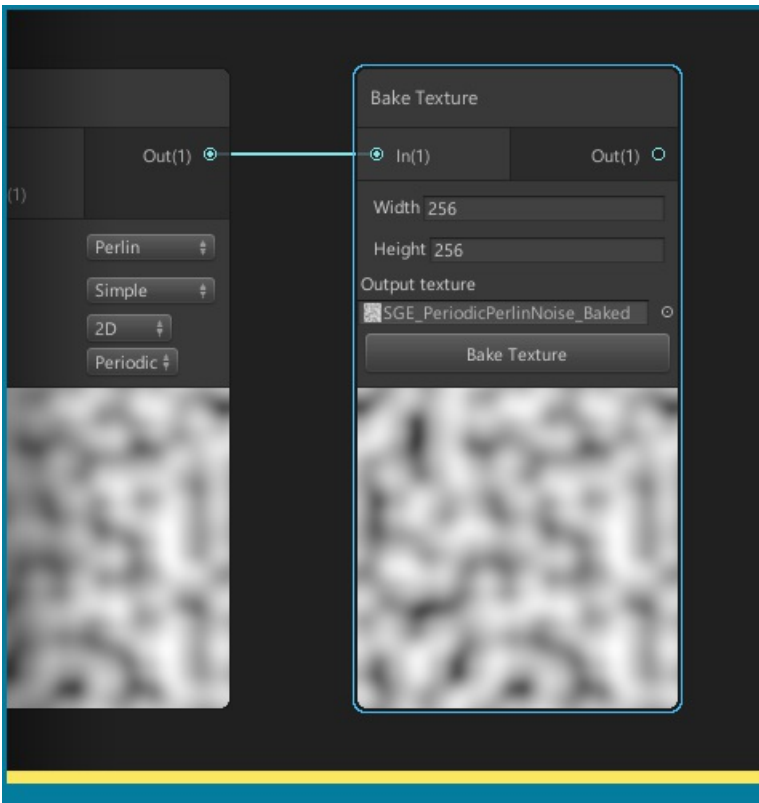- Fast
- Mobile / VR / PC
- LWRP / HDRP

- Simple Lit master node
- Custom blend mode
- Custom ZWrite / ZTest
- Custom editor
- LWRP

- Custom lighting master
- Toon lit example
- Correct shadows
- Automatic multi-light support
- LWRP - 2019.1+

- Bake texture from any node
- Optimize noises
- Optimize generated normals

- Water shader 100% in Shadergraph
- CPU or GPU waves
- Depth-based foam effect
- LWRP