Simple Laser Documentation

September 2022

version 1.0.0

Contents

1 Greetings **2**

2 Project setup **2**
- 2.1 Dependencies . . . 2
- 2.2 Contents . . . 2
- 2.3 Setting up the Built-In pipeline . . . 3
- 2.4 Setting up the Universal pipeline . . . 4
- 2.5 Setting up the High Definition pipeline . . . 5

3 Code **6**
- 3.1 Laser Beam Controller . . . 6
- 3.2 Laser Particle Controller . . . 6
- 3.3 Projector Controller . . . 6

4 Laser shader settings **6**

1 Greetings

Thank you for downloading Simple Laser, a shader with minimal code to get you great looking laser beams to your project. In the following short documentation we'll look at the project setup and discuss the customization options for the laser shader.

2 Project setup

2.1 Dependencies

The project uses Unity's Shader Graph for its shaders, even for the Built-in renderer. Enabling this in the project is a must and it has to be a version that supports the built-in render pipeline.
The shader graph is added to the package definition as a dependency with the version that was used to create it, so when you download the package, Unity should automatically include the dependency as well. However if you encounter purple shaders that signify an error, please make sure you have the appropriate version of Shader Graph downloaded.

2.2 Contents

The project comes with support for Built-in, HDRP and URP and there are separate materials, prefabs and scenes set up for each of these pipelines.
The settings for the laser prefabs might differ a bit for each pipeline for a similar result and also, some elements like projectors are unique to each pipeline (more on that later).
Please make sure to always use the appropriate assets for your project's pipeline.

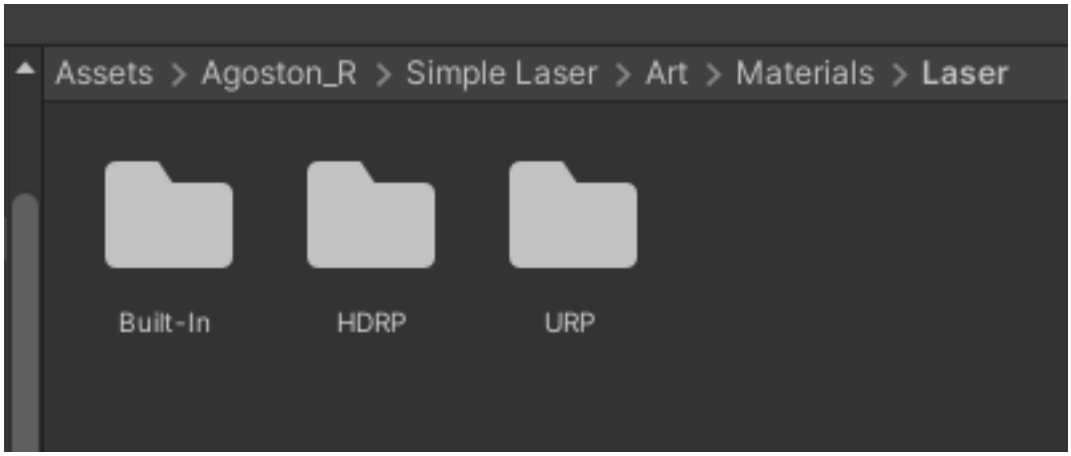


Figure 1: Example folder structure

2.3 Setting up the Built-In pipeline

The built-in pipeline requires Shader Graph too for the laser shader to compile. If for some reason the editor does not download this alongside the package as a dependency, please make sure you download a version that supports the built-in pipeline.

For projection, the built-in version uses Unity's Projector component.
As a headsup, if you use your own texture for the projector, make sure you set its *Wrap mode* to *Clamp* or the texture will wither extend its edge or repeat. Also make sure that the edges of the texture are black.

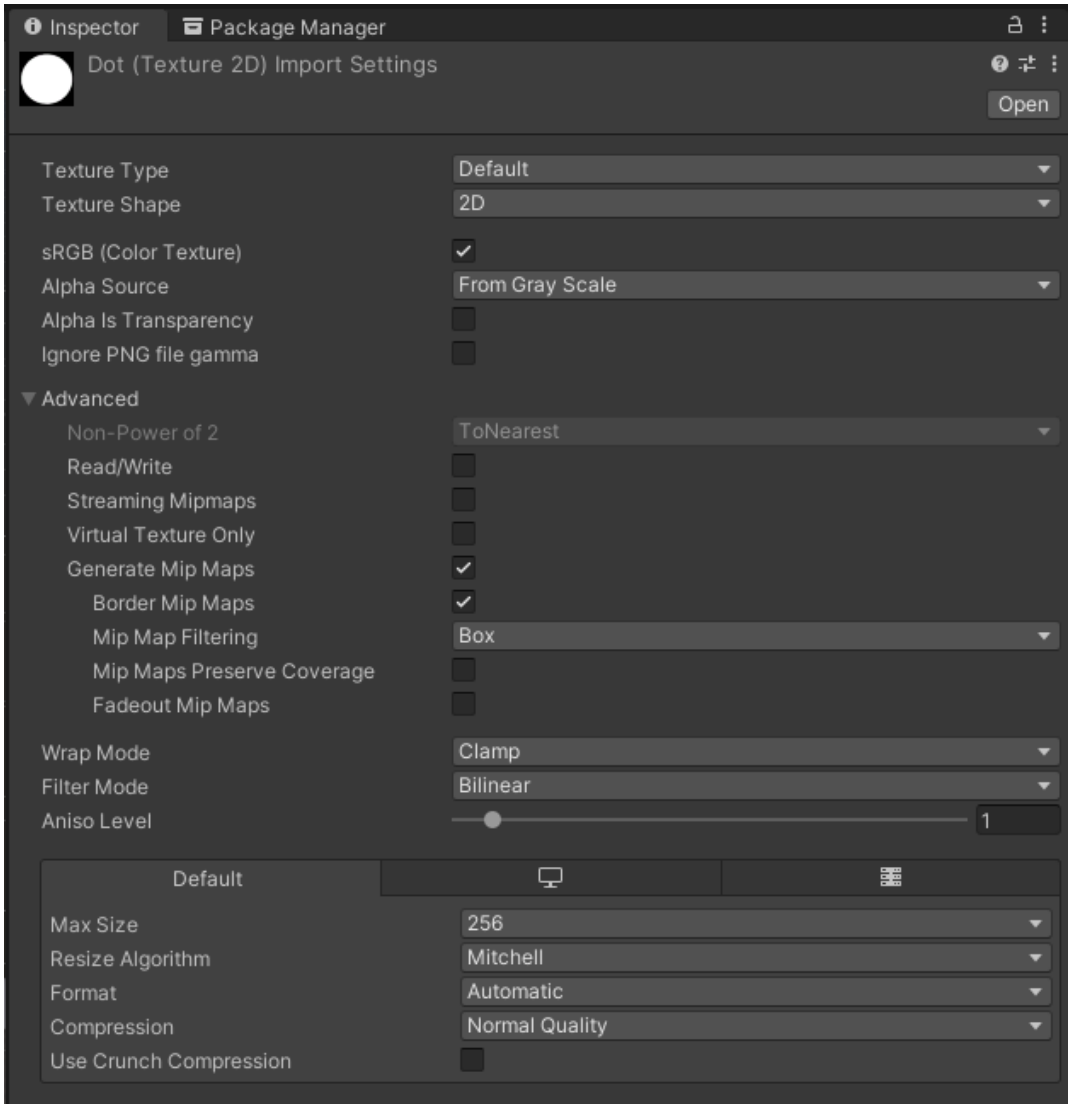


Figure 2: Projector texture settings example

If these are set then you may drag and drop a laser prefab from the Built-in folder to your scene or just use the one of the laser materials or the laser shader to create your own laser beams.

2.4 Setting up the Universal pipeline

If you upgrade your project to URP manually then it requires some configuration that is posted on Unity's website. But to check out the asset as URP it's easier to just create a new URP project as a template.

For projection, URP uses its own Decal Projector that is distinct from Unity's built-in projector. It also does not support the built-in Projector and that is a reason why I had to keep the prefabs separate.

Unfortunately, decals are not enabled by default.

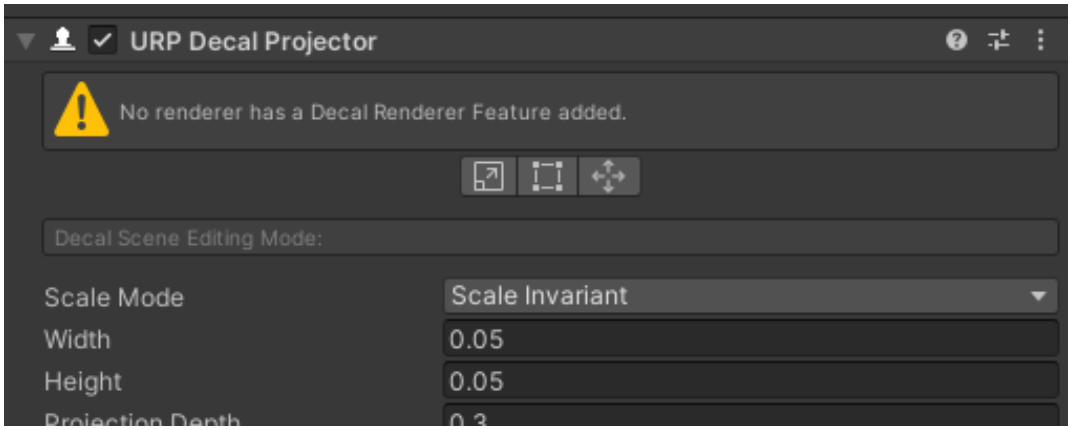
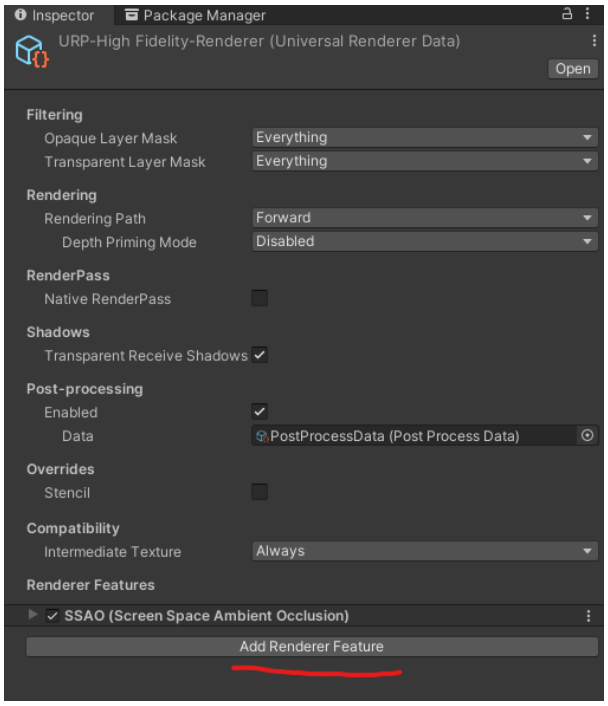
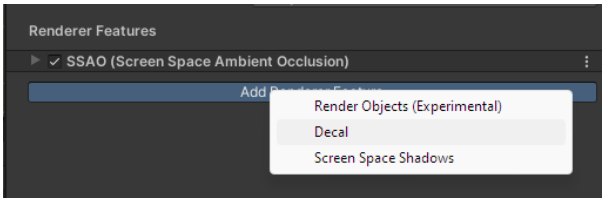


Figure 3: Missing decals config warning

To enable decals, look for a URP renderer asset and add the decal feature.
Also make sure that your pipeline asset uses the renderer asset which has the decal feature enabled.

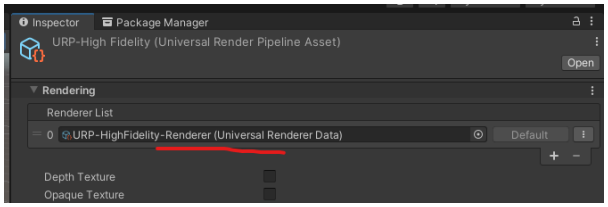


(a) Add decals 1



(b) Add decals 2

Figure 4: Adding decals to a renderer



(a) Renderer with decals

(b) Decals in action

Figure 5: Decals in URP

2.5 Setting up the High Definition pipeline

HDRP's setup is very similar to URP's so for the details, please check that out. It has its own Decals Projector as well, and like URP it does not support the built-in Projector component. Enabling Decals is similar to the process defined in URP.

Much the same as the other pipelines, if you use HDRP please make sure to always use the prefabs / materials etc from HDRP folders and open the HDRP test scene so it matches your project's settings.

3 Code

The asset uses a minimal amount of code to control the projectors, particles and shaders.

3.1 Laser Beam Controller

The Laser Beam Controller is responsible for handling the laser shader, the raycasts that determine the Line Renderer's points and it invokes the particle systems to play at on laser hit.

You do not need to create particles or projector for the laser beam to work. If you have no particle to play on laser hit, it'll just invoke an empty implementation.

3.2 Laser Particle Controller

The Laser Particle Controller handles the particle systems on command of the beam controller. You may set this up for a list of particle systems that you want to color and control. If you have particle like smoke that should ignore coloring, use the *IgnoreColoring* component on them.

3.3 Projector Controller

Contains logic to enable or disable the projectors. As it's a common class for all pipelines, preprocessor directives are enabled in the project's assembly to ignore the pipelines that are not enabled.

4 Laser shader settings

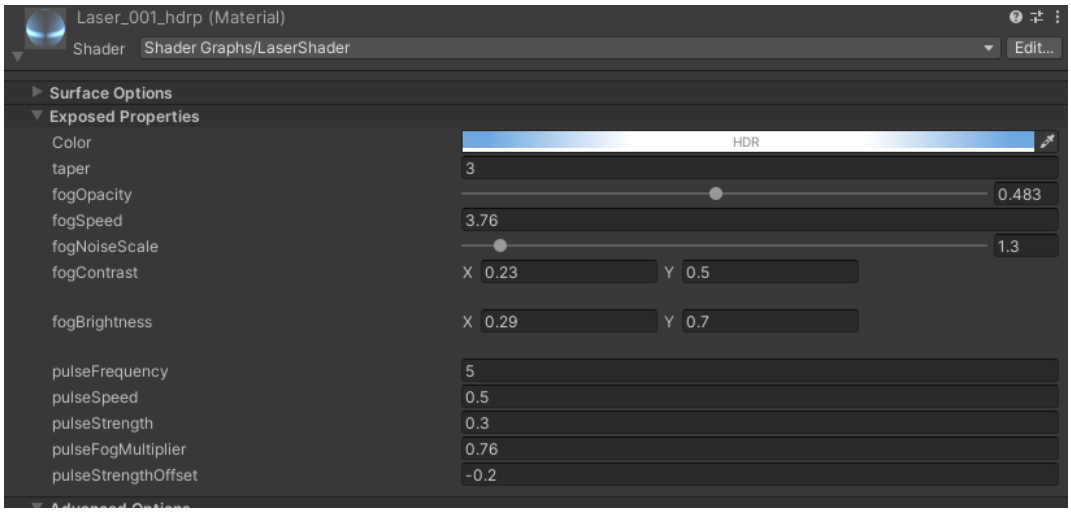


Figure 6: Laser shader settings

The laser shader's settings can be found above.

- **Color** is a HDR color that sets the color and intensity of the laser. To make it pop you may have to enable and tweak Bloom in your pipeline's settings. Bloom settings are specific to each render pipeline.
- **taper** is a smoothing at the end of the laser shader, to reduce harsh endpoints. Values less than 0.5 do not take effect (that would be the middle of the laser beam). The higher the value, the sharper the beam's endpoints.
- **Fog opacity** the opacity of the animated fog that surrounds the laser beam.
- **Fog noise scale** the scale of the noise generator that creates the fog.
- **Fog Contrast and Fog Brightness** these are like Levels in Photoshop, for the fog's noise generator.
- **Pulse frequency** the laser has a periodic pulse that indicates its direction. It's a Sine wave and this is it's frequency. Increase this value for more frequent pulses.
- **Pulse speed** sets how fast the pulse goes.
- **Pulse strength** sets how big the difference in color intensity for the pulse, it's the sine wave's amplitude.
- **Pulse fog multiplier** sets how much pulse affects the laser's fog.
- **Pulse strength offset** offsets the pulse - it's the sine wave's translation along Y.