Simple Laser Documentation

Nov 2022

version 1.1.0

Contents

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 3D and 2D support

The project supports both 3D and 2D physics. This is selected on the *LaserBeamController* as an enum. In case of a 3D laser, the raycast is issued forward, in case of a 2D one the raycast is issued to the right.

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.
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.

Note that the projection materials may appear purple for the URP and HDRP pipelines if you use the Built-in pipeline. The reason is that the HDRP and URP uses a shader graph based projection material. Those projection shaders are not used in the Built-in pipeline.

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.

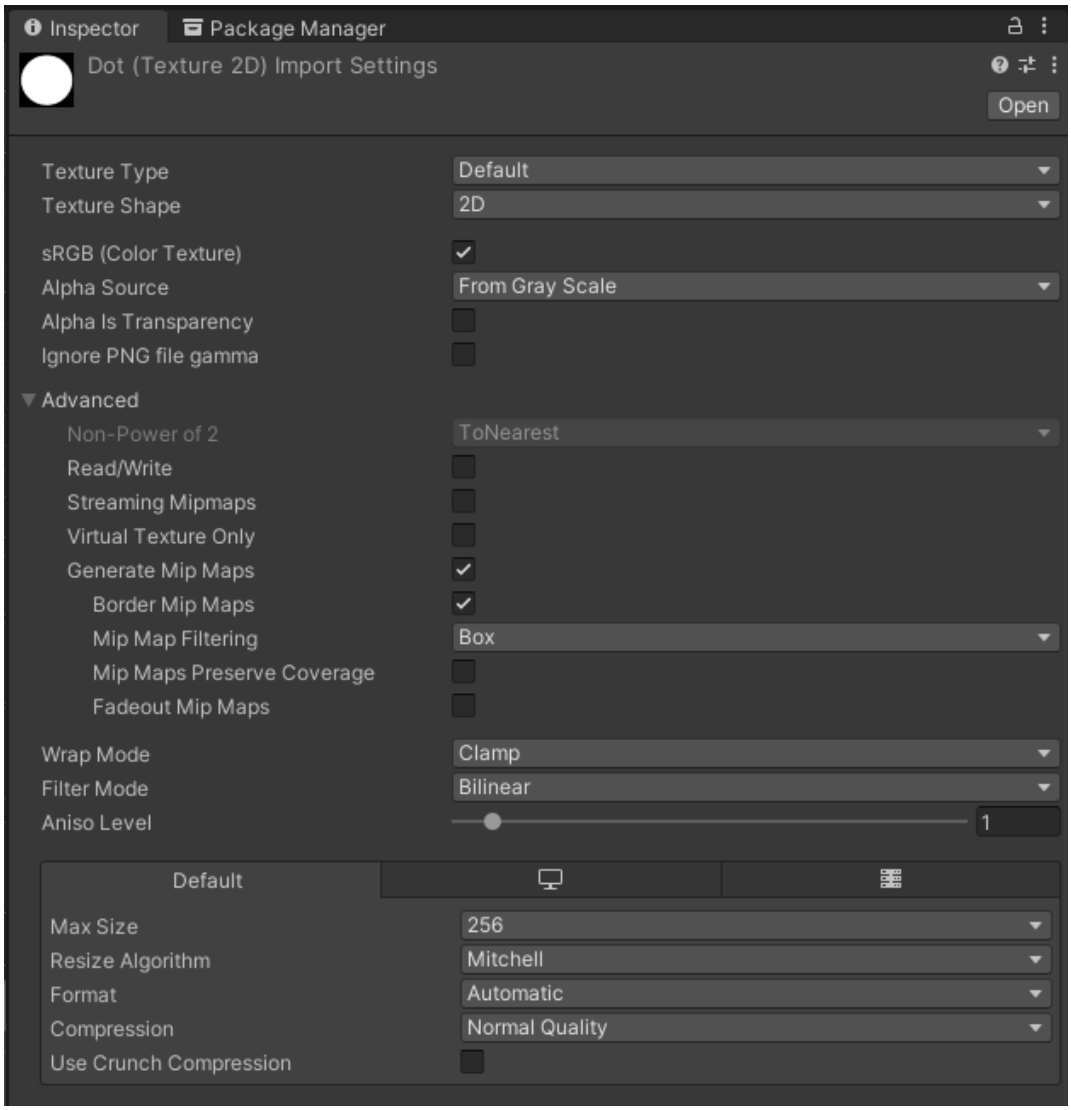


Figure 1: Projector texture settings example

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 create a bootstrapper to add the components runtime. It is only needed for the prefabs though, naturally you can just pick a render pipeline of your choice and add the projector manually if you need it.

Unfortunately, decals are not enabled by default.
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.

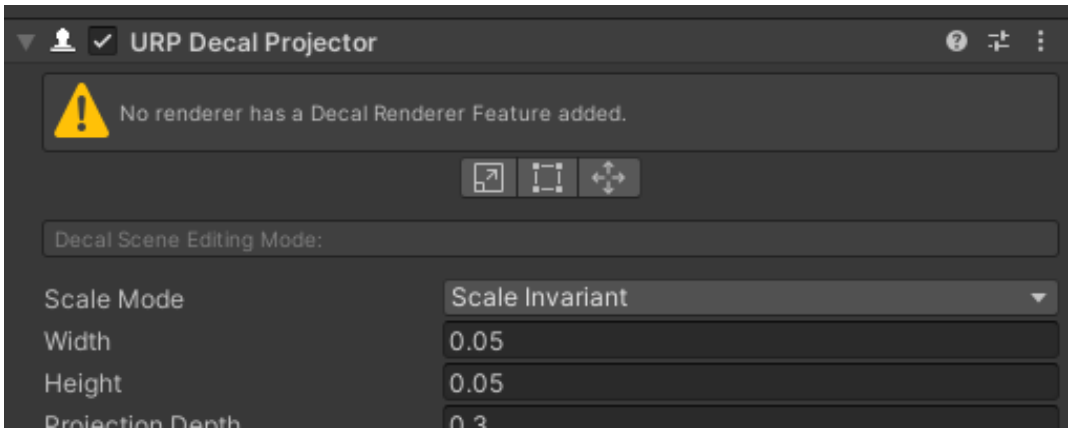
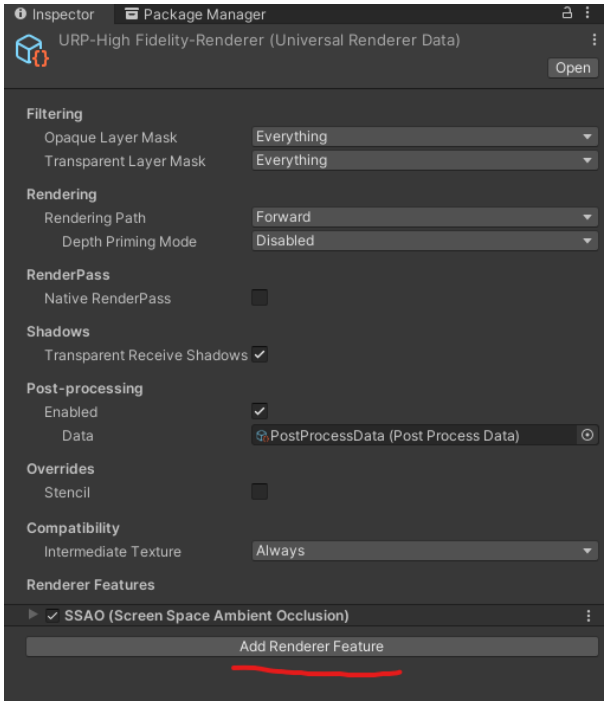
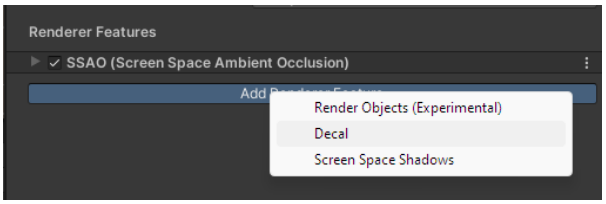


Figure 2: Missing decals config warning



(a) Add decals 1

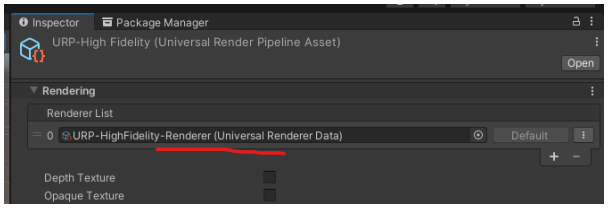


(b) Add decals 2

Figure 3: Adding decals to a renderer

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.



(a) Renderer with decals

(b) Decals in action

Figure 4: Decals in URP

3 Code

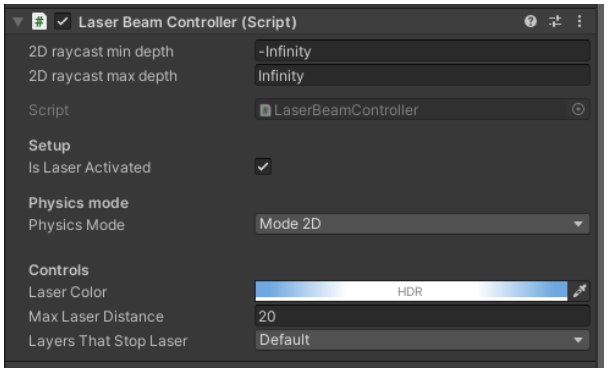
The asset uses a minimal amount of code to control the projectors, particles and shaders.

3.1 Laser Beam Controller

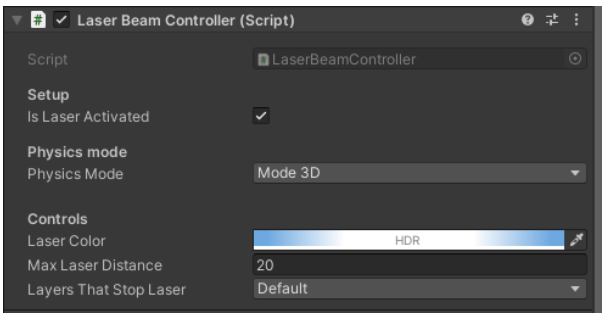
The Laser Beam Controller is responsible for handling the laser shader, the raycasts that determine the Line Renderer's points and it invokes the particle systems to play at on laser hit.

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.

The laser beam has options to use 3D or 2D physics which will determine the physics interaction with its environment. It is set by a simple enum in **physics mode**.
The 2D mode has two additional options, the max depth and min depth. These are used by the *Physics2D.Raycast* as min and max depth, that is, the Z index limits to determine how you want your laser to interact with objects at a given Z position - an objects outside of these values will let the laser pass through. The default settings interact with objects of any Z coordinates.



(a) Beam with 3D settings



(b) Beam 3D settings

Figure 5: Decals in URP

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.

3.4 Projector Bootstrapper

To support various pipelines with prefabs without forcing dependencies to the receiving project, a bootstrapper sets up the two decal projectors for URP and HDRP, or the Projector component for the Built-in pipeline. Naturally, you can use the shaders and the projector controller in your own project just by adding the appropriate projector and omitting the controller.

4 Laser shader settings

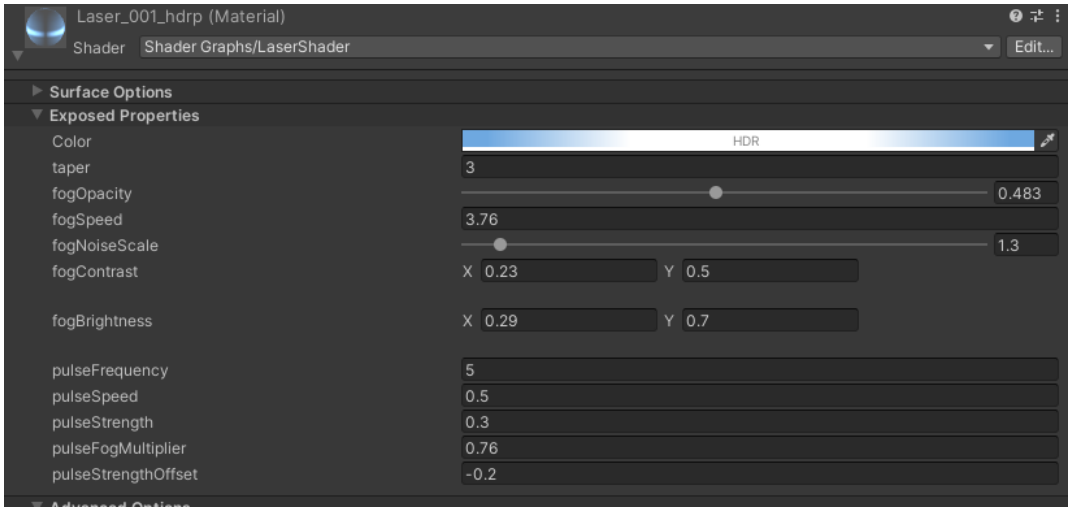


Figure 6: Laser shader settings

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.