Yel Scrypt Fire Studio

Space Shooter Projectile 2D + 2.5D Part 1

SCRIPTS AND CUSTOM SHADERS DOCUMENTATION

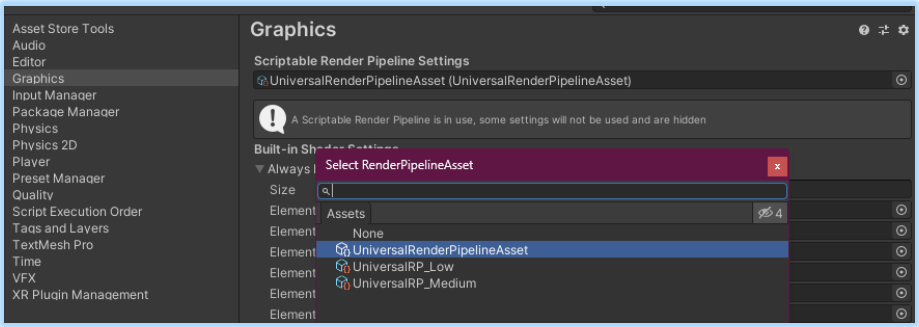
Version 1.1

CONTENIDO

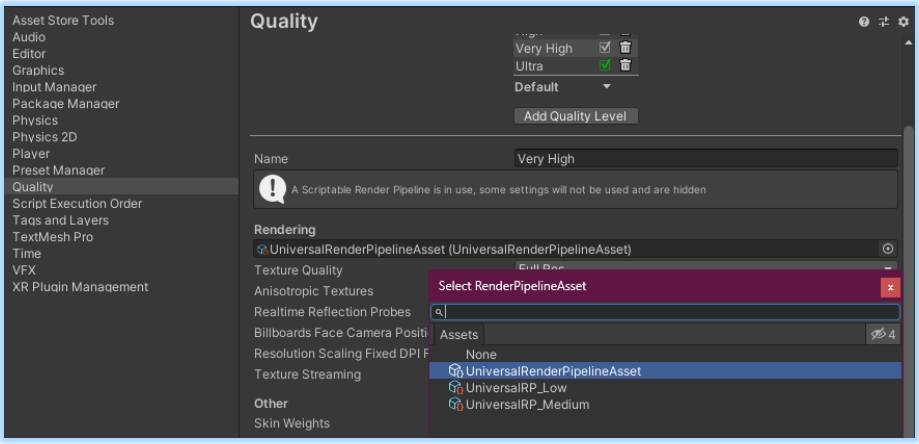
HOW TO ACTIVATE UNIVERSALRP

*This step is not needed if your project is configured in URP by default. *

- Go to **Edit > Project Settings**
- In the configuration windows, go to **Graphics** and set the URP asset, clicking on the circle and selecting the asset.



- Go to **Quality** and set the same URP asset.

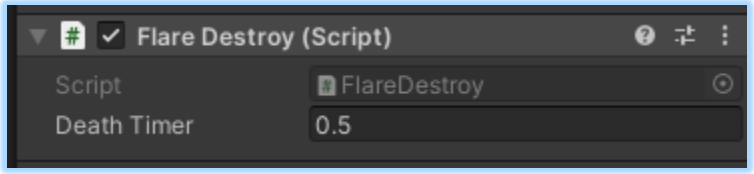


- **If files are not recognized, please reimport the URP assets.**
- To change from URP to Bult-in, just select **None** instead of the URP asset.

FLARE DESTROY SCRIPT

It's installed in the "**Flare**" and "**Hit**" prefabs.

It calls for the destruction of the object by a timer. Timer can be set by the variable "Death Time" in seconds.



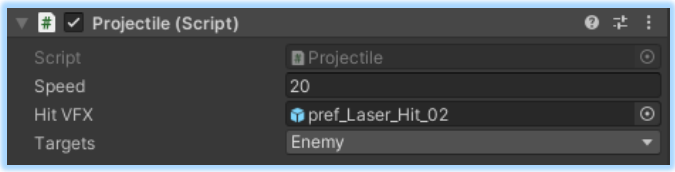
PROJECTILE SCRIPT

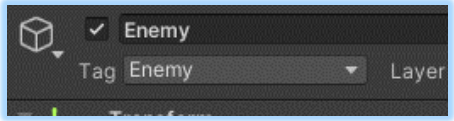
It's installed in the **Projectile** prefabs.

Gives the forward motion of the instantiated projectile. The speed can be changed by the variable **Speed**.

It instantiates the "Hit" prefab when projectile hits an **enemy**. The **Hit** prefab must be referenced here.

The **Hit** target can be changed to **Enemy** or **Player.** For hit detection, the target object needs to be tagged as **Enemy** or **Player**.





SORTING LAYER ORDERS

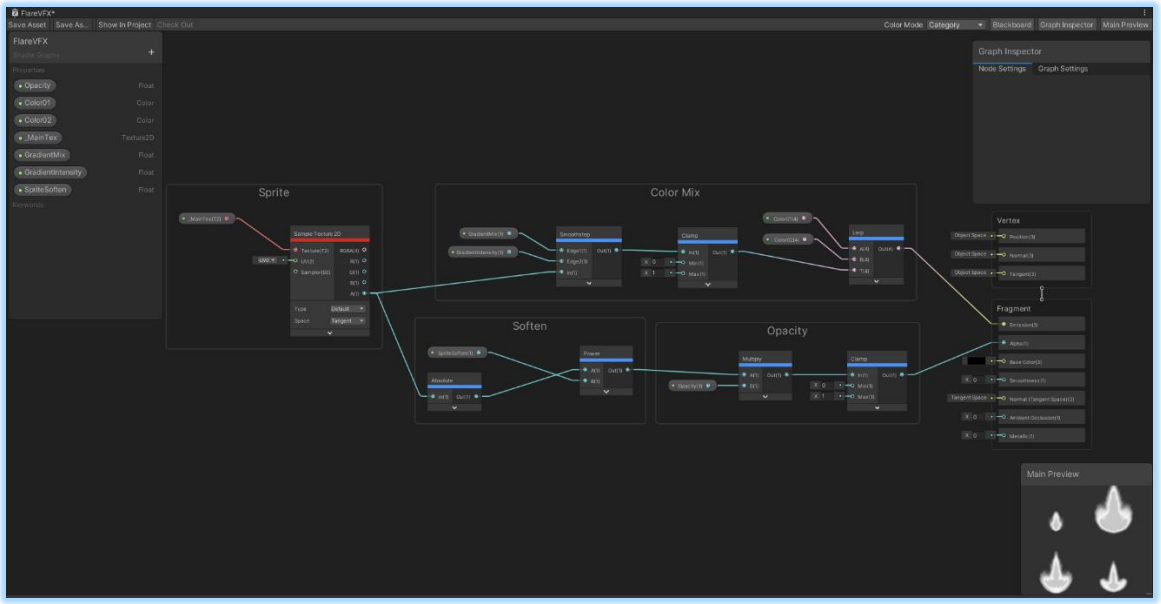
Sprites are used instead of textures to be able to use **Sorting Layers** and make sure objects are rendered in the correct order.

General order for **Flares** and **Hits**: **Sparks** < **Flares** < **Waves** < **Flashes** < Other **VFX**

General order for **Projectiles**: **Projectile** < **Trail** < **Sparks**

FLAREVFX SHADER GRAPH

Shader for **Flares**, **Flashes**, **Sparks** and **Waves** materials.



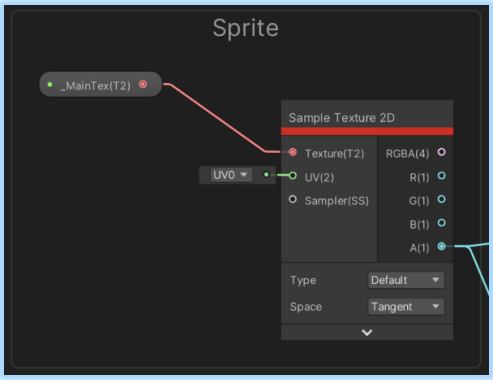
SPRITE

Input of the texture/sprite. Sprites are preferred.

The shader uses the **Alpha** channel. Transparency and color gradients are created from **Alpha** channel.

Output: → Color Mix section

→ Soften section

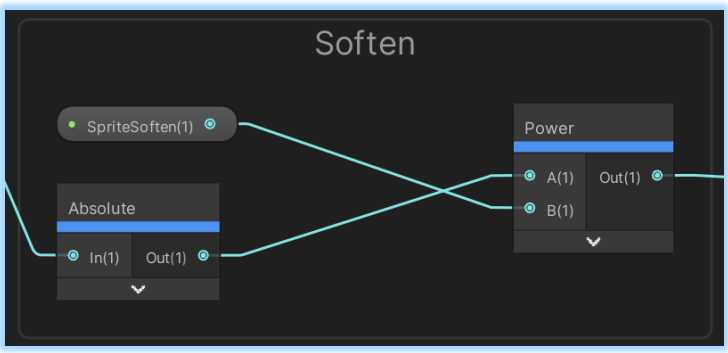


SOFTEN

It uses a **Power** node to make the edges of the sprite more soft and less opaque, or hard and more opaque.

Absolute node avoids negative values in **Power** node.

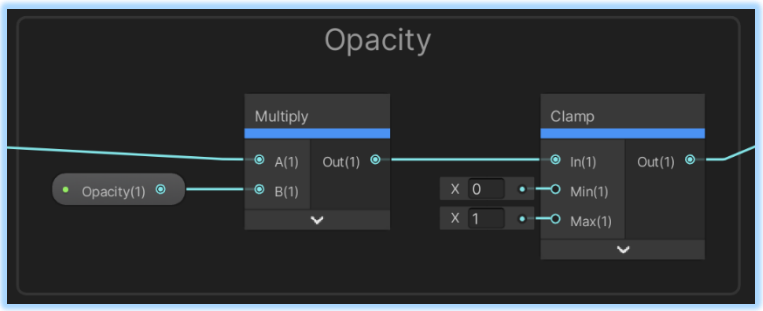
Output: → Opacity section



OPACITY

It uses **Multiply** node to control the opacity of the entire shader. **Clamp** node is to prevent negative values from **Multiply** node.

Output: → Alpha Output.

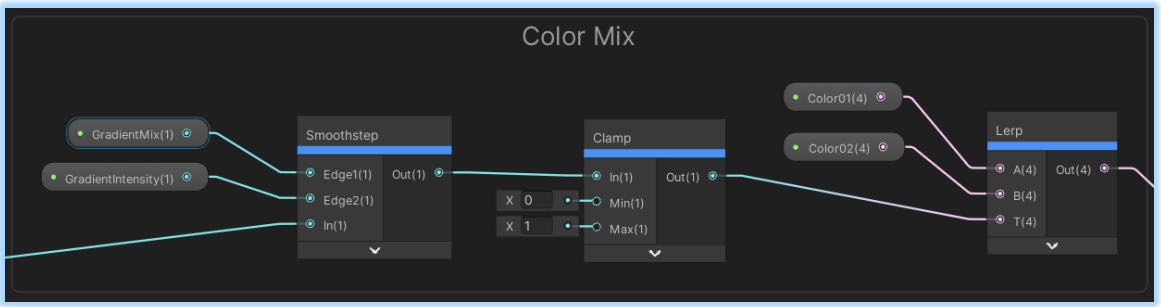


COLOR MIX

It uses **Lerp** node to mix two color as a gradient.

Smoothstep node is used to control the distance between both ends of the gradient.

Output: → Emission Output



ON INSPECTOR

Opacity: the opacity of the entire shader.

(0 <-> 1)

Color01 (HDR): outer color (Low end of the gradient).

Color02 (HDR): inner color (high end of the gradient).

_MainText: texture/sprite input. Sprites are preferred.

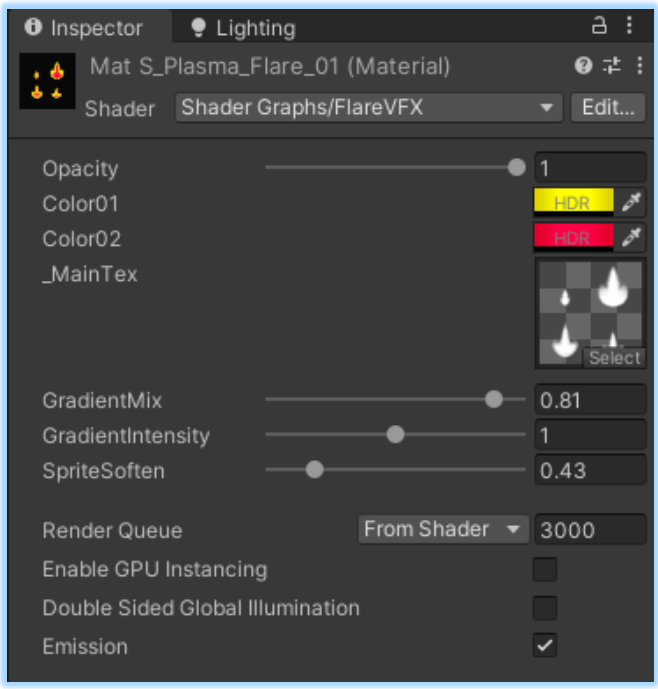
GradientMix: the amount of mixing between both colors by rising the low end of the gradient. *(-1 <-> 1)*

GradientIntensity: the amount of intensity of the inner color by lowering the high end of the gradient. *(0 <-> 2)*

SpriteSoften: the smoothing and opacity of the edges of the sprites.

(0.3 <-> 2.5)

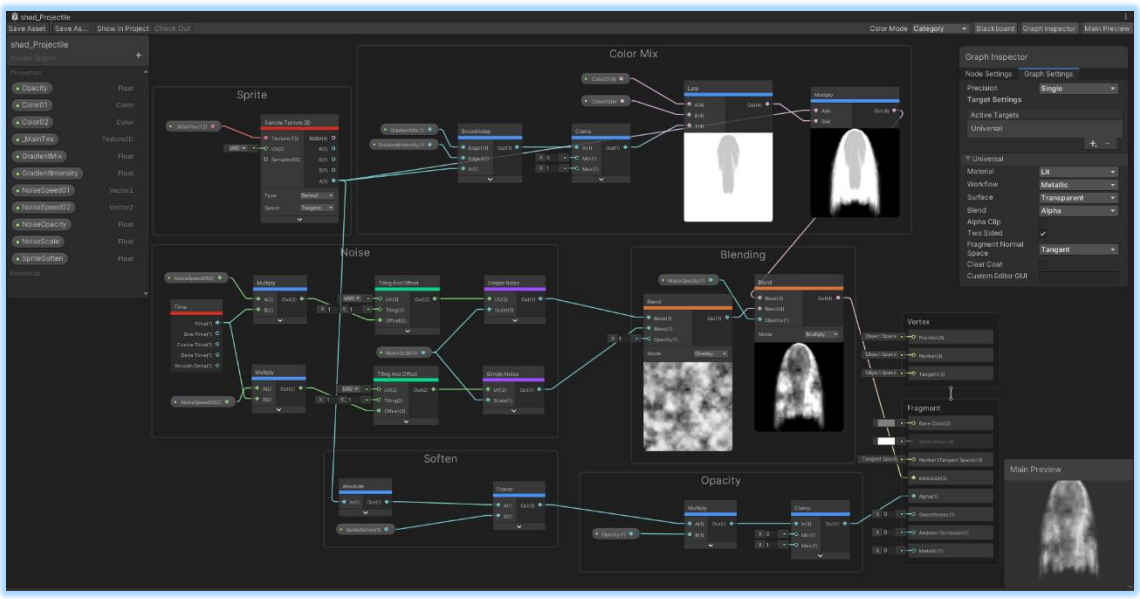
Note: "Emission" must be activated to prevent receiving shadows from other objects.



PROJECTILE SHADER GRAPH

Shader for **Projectile** sprites.

Combines a sprite with a noise effect, coloring with two colors using the **Alpha** channel as the parameter for the color gradient.



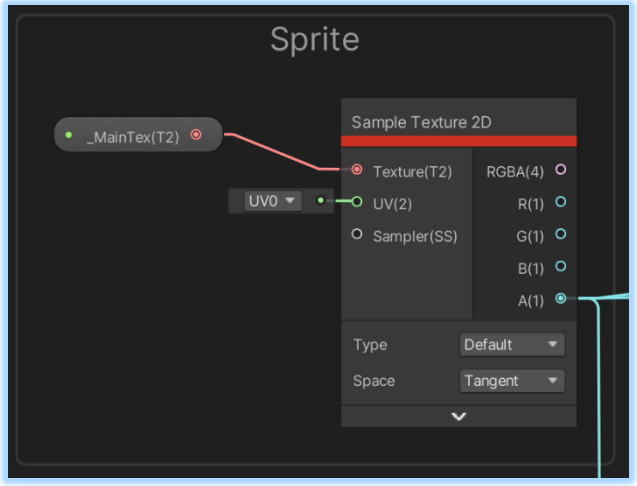
SPRITE

Input of the texture/sprite. Sprites are preferred.

Transparency and color gradients are created from **Alpha** channel.

Output: → Color Mix section

→ Soften section



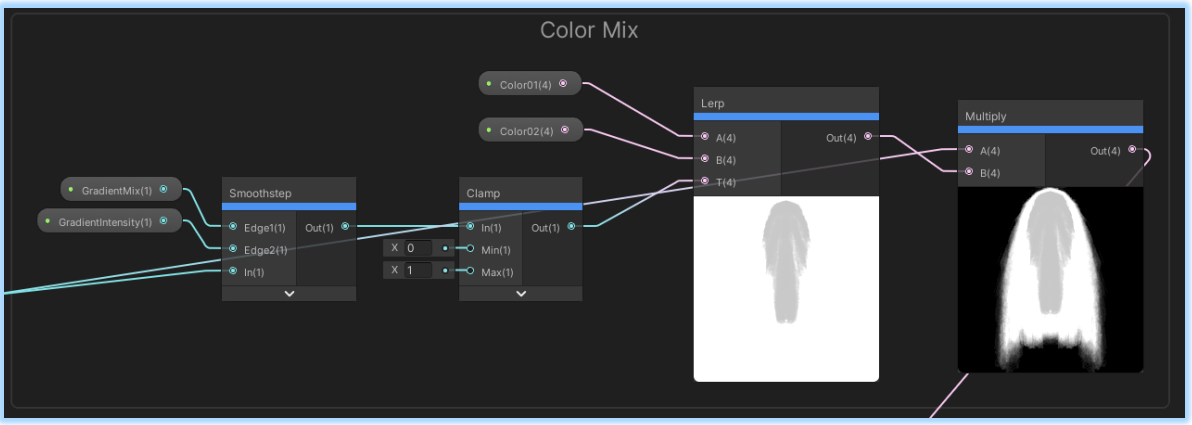
COLOR MIX

It uses **Lerp** node to mix two color as a gradient.

Smoothstep node is used to control the distance between both ends of the gradient.

The output is multiplied by the **Alpha** channel from the **Sample Texture 2D** node. This corrects the values from **Lerp** node.

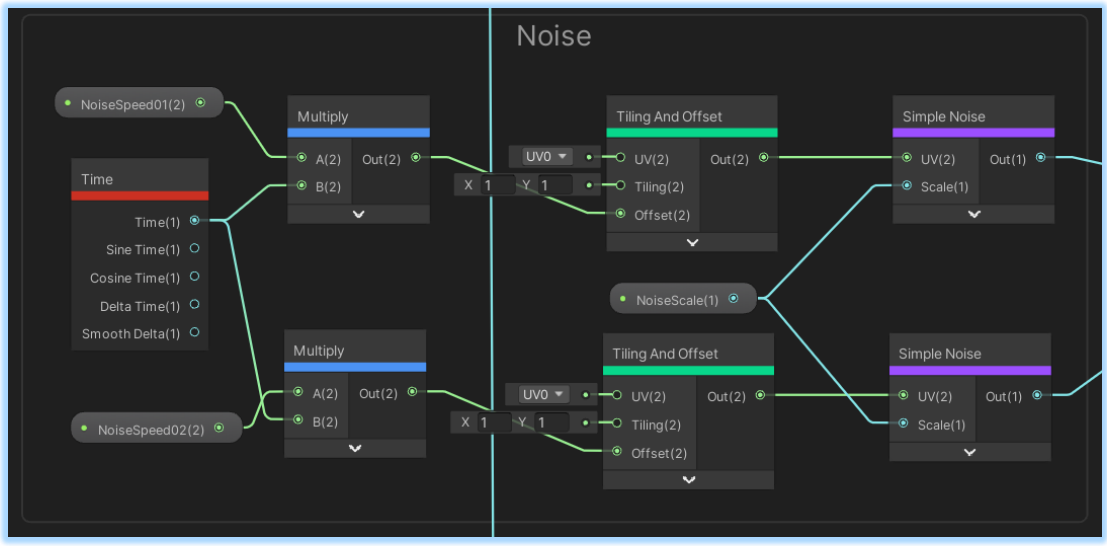
Output: →Blending section



NOISE

It uses double **Simple noise** nodes controlled by **Tiling and Offset** nodes. The movement is automated by **Time** node.

Output: →Blending section

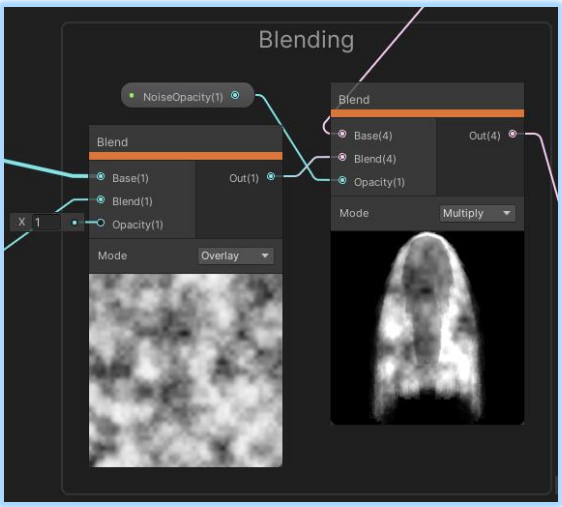


BLENDING

It takes the outputs from **Color Mix** section and **Noise** section.

Combines the two outputs from **Noise** nodes using **Overlay** blending, and then it's combined with the output from **Color Mix** section using **Overlay** blending too.

Output: → Emission Output

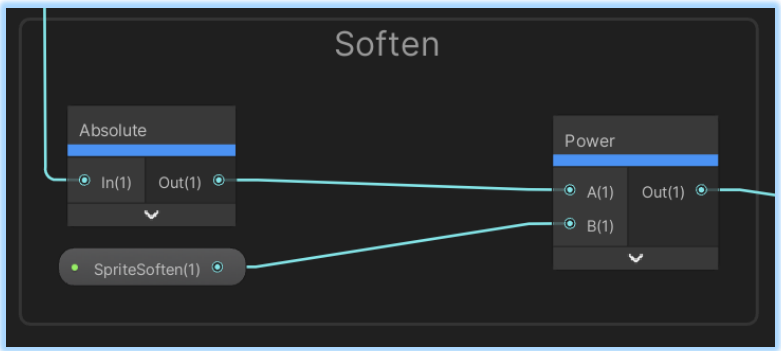


SOFTEN

It uses a **Power** node to make the edges of the sprite more soft and less opaque, or hard and more opaque.

Absolute node avoids negative values on **Power** node.

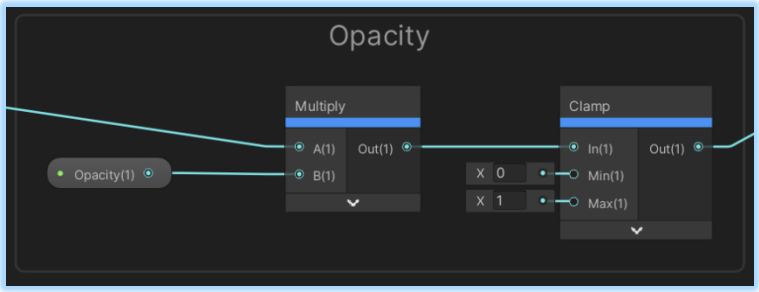
Output: → Opacity section



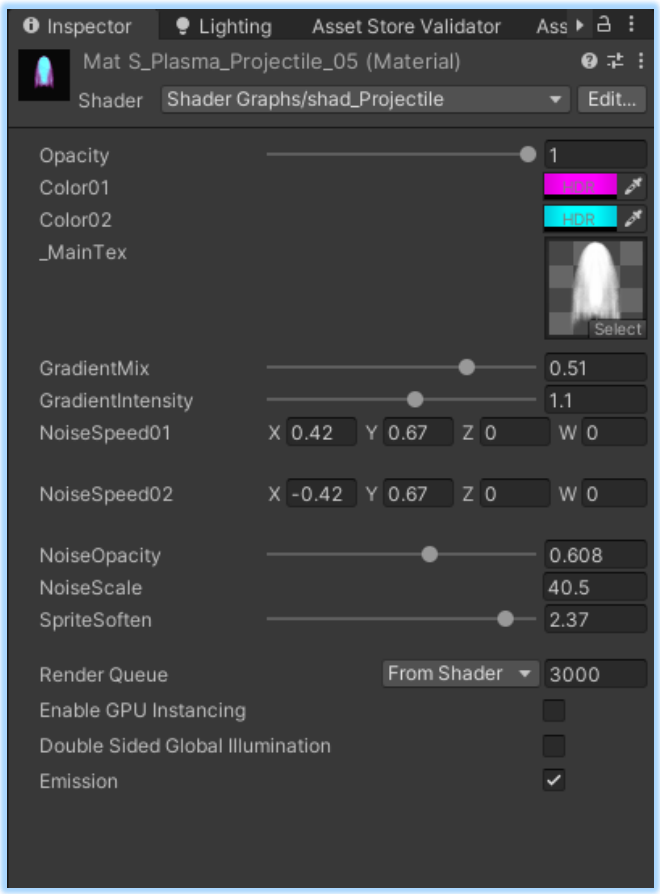
OPACITY

It uses **Multiply** node to control the opacity of the entire shader. **Clamp** node is to prevent negative values from **Multiply** node.

Output: →Alpha Output.



ON INSPECTOR:



Opacity: the opacity of the entire shader.

(0 <-> 1)

Color01 (HDR): outer color (Low end of the gradient).

Color02 (HDR): inner color (high end of the gradient).

_MainText: texture/sprite input. Sprites are preferred.

GradientMix: the amount of mixing between both colors by rising the low end of the gradient. *(-1 <-> 1)*

GradientIntensity: the amount of intensity of the inner color by lowering the high end of the gradient. *(0 <-> 2)*

NoiseSpeed: speed/direction of the noise effect. There are two noise effects combined. Speed/direction parameters are independent between the two noises.

NoiseOpacity: the amount of noise that is added to the sprite. *(0 <-> 1)*

NoiseScale: the scale of the **Noise** node.

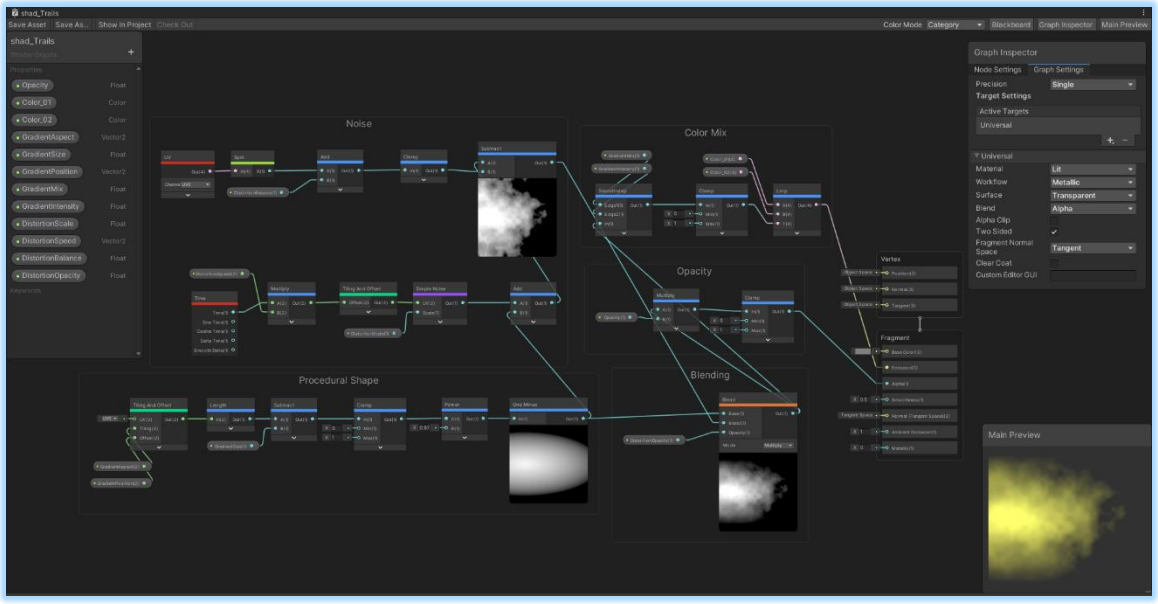
SpriteSoften: the smoothing and opacity of the edges of the sprites.

(1 <-> 2.5)

Note: Emission must be activated to prevent receiving shadows from other objects.

TRAILS SHADER GRAPH

Shader for **Trails** VFX

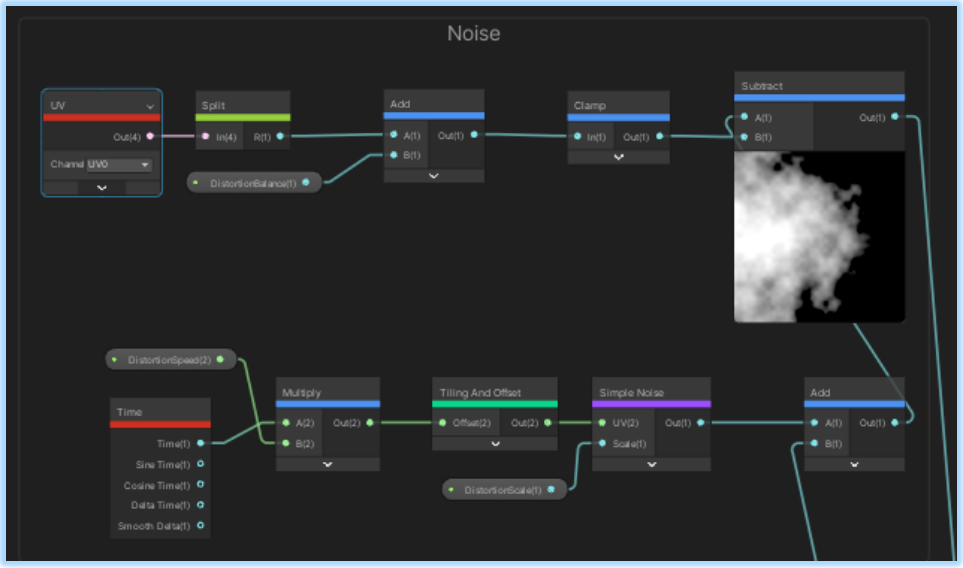


NOISE

It uses **Noise**, **Tiling and Offset**, and **Time** nodes to create the visual effect, plus it uses an **UV** node to create a gradient. The output is a fade-out noise.

The output from **Procedural Shape** section is added to the **Noise** output node, and the results are subtracted to the gradient result, shaping the noise.

Output: → Blending section



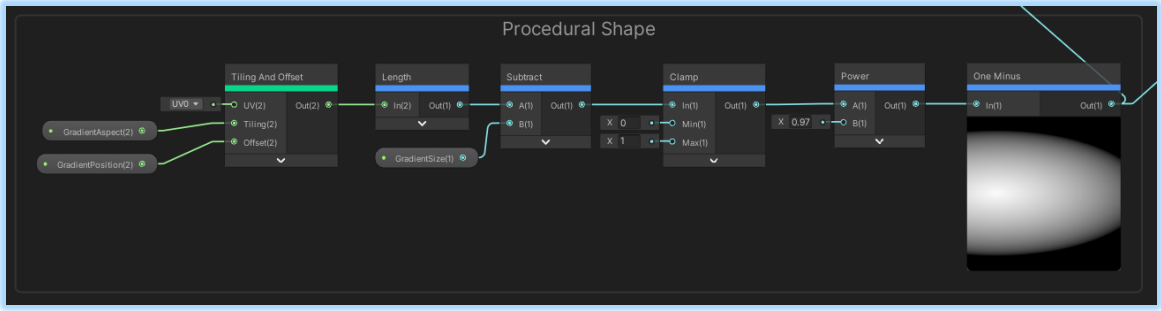
PROCEDURAL SHAPE

It creates a radial gradient that can be moved and changed in aspect ratio.

Uses the **UV** mapping of the **Tiling and Offset** node and creates a radial gradient using the **Length** node.

Output: →Noise section

→Blending section

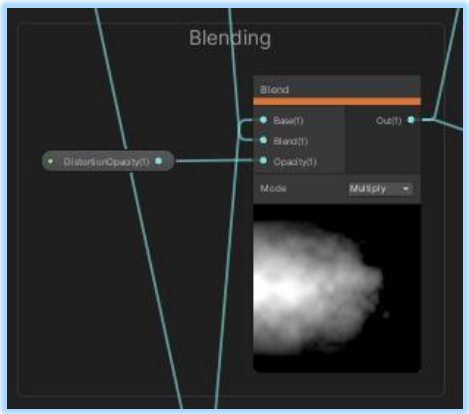


BLENDING

Combines **Procedural Shape** section with **Noise** section using **Multiply** blending, and it's controlled by the variable **DistortionOpacity**.

Output: → Color Mix section

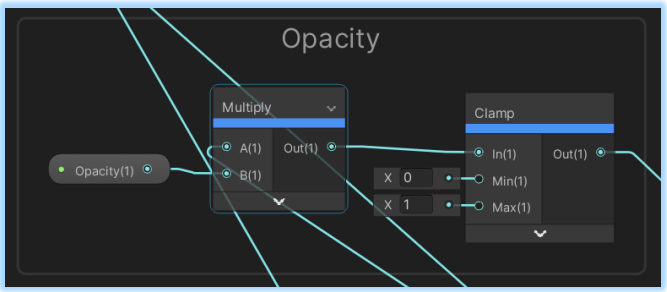
→ Opacity section



OPACITY

It uses **Multiply** node to control the opacity of the entire shader. **Clamp** node is to prevent negative values from **Multiply** node.

Output: →Alpha Output.

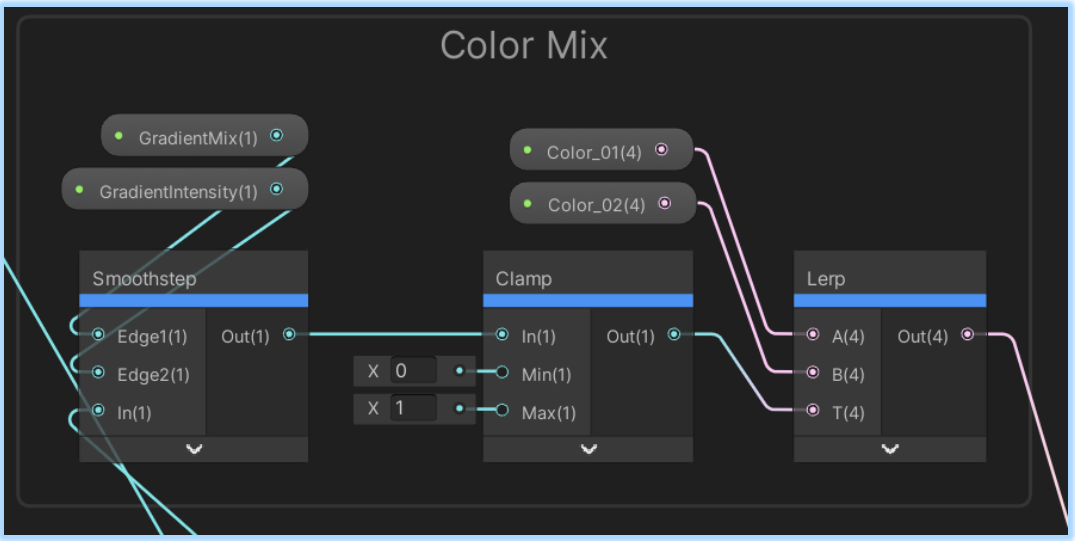


COLOR MIX

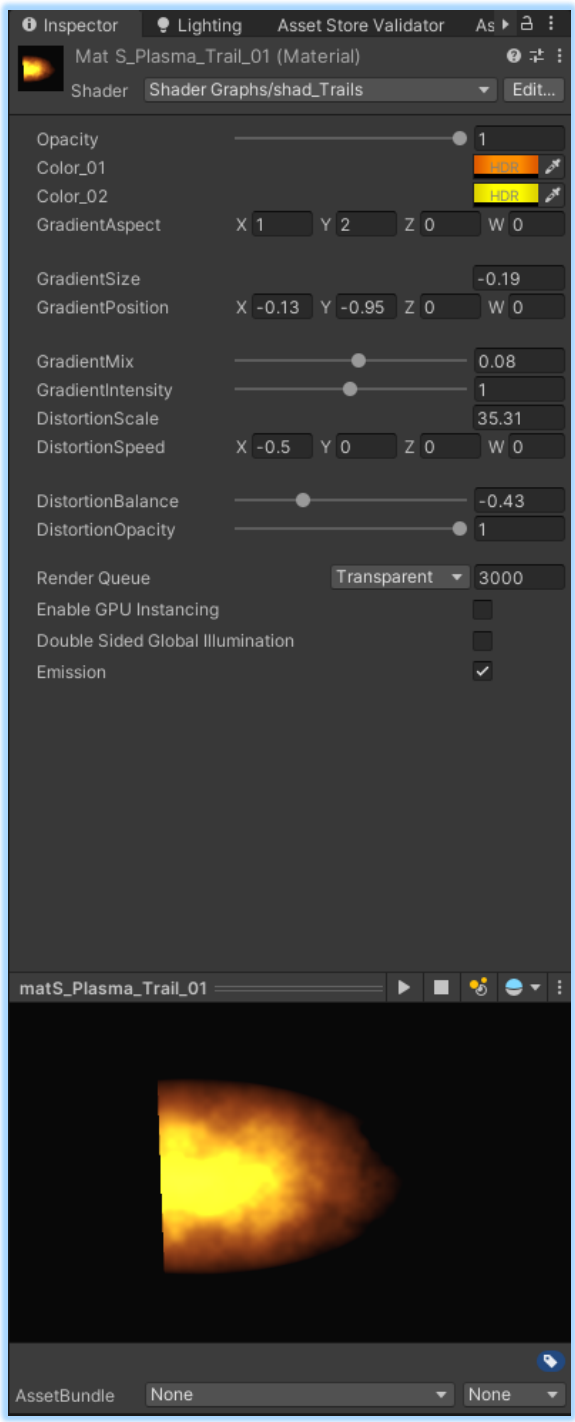
It uses **Lerp** node to mix two color as a gradient.

Smoothstep node is used to control the distance between both ends of the gradient.

Output: → Emission Output



ON INSPECTOR



Opacity: the opacity of the entire shader.

(0 <-> 1)

Color01 (HDR): outer color (Low end of the gradient).

Color02 (HDR): inner color (high end of the gradient).

GradientAspect: aspect ratio of the procedural shape. Greater values make shape thinner.

X > width.

Y > Height.

GradientSize: the size of the shape.

GradientPosition: position of the shape. Its position depends of the GradientAspect, so it must be adjusted if the aspect changes.

DistortionScale: the scale of the **Noise** node used for the distortion effect.

DistortionSpeed: it controls the speed and direction of the noise.

DistortionBalance: it controls the amount of gradient modifying the noise. *(-1 <-> 1)*

DistortionOpacity: the amount of distortion that is applied over the trail.

Note: Emission must be activated to prevent receiving shadows from other objects.