Yel Scrypt Fire Studio

Space Shooter Projectile 2D + 2.5D Part 1

SCRIPTS AND CUSTOM SHADERS DOCUMENTATION

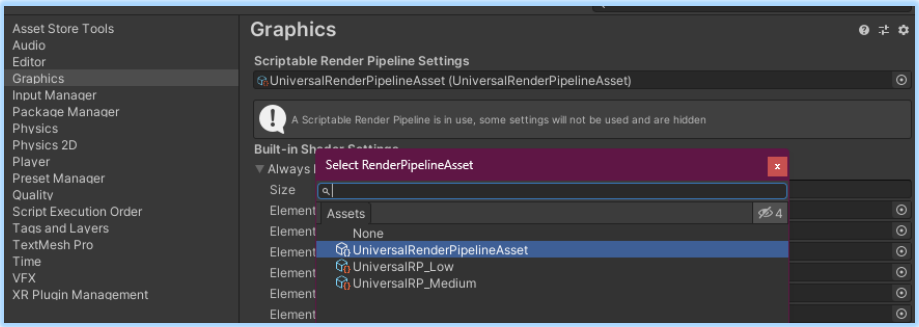
Version 1.2

CONTENIDO

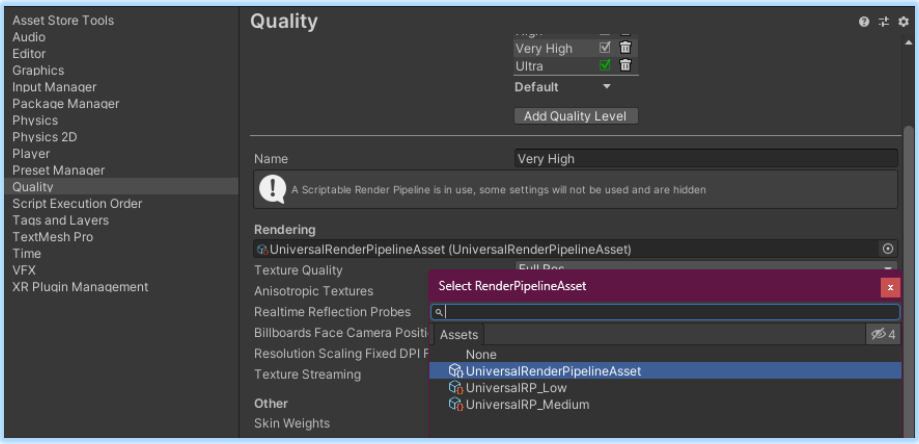
HOW TO ACTIVATE UNIVERSALRP

*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.

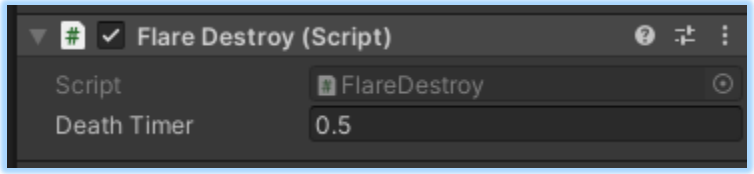


- **If files are not recognized, please reimport the URP assets.**
- To change from URP to Bult-in, just select **None** instead of the URP asset.

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.



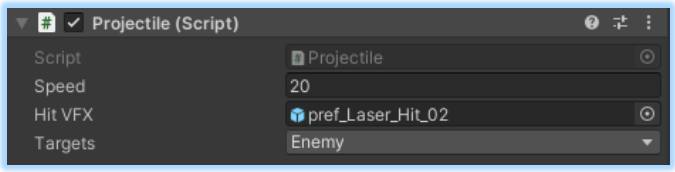
PROJECTILE SCRIPT

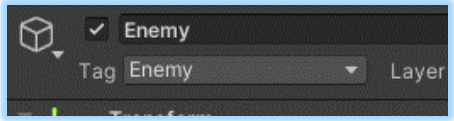
It's installed in the **Projectile** prefabs.

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**

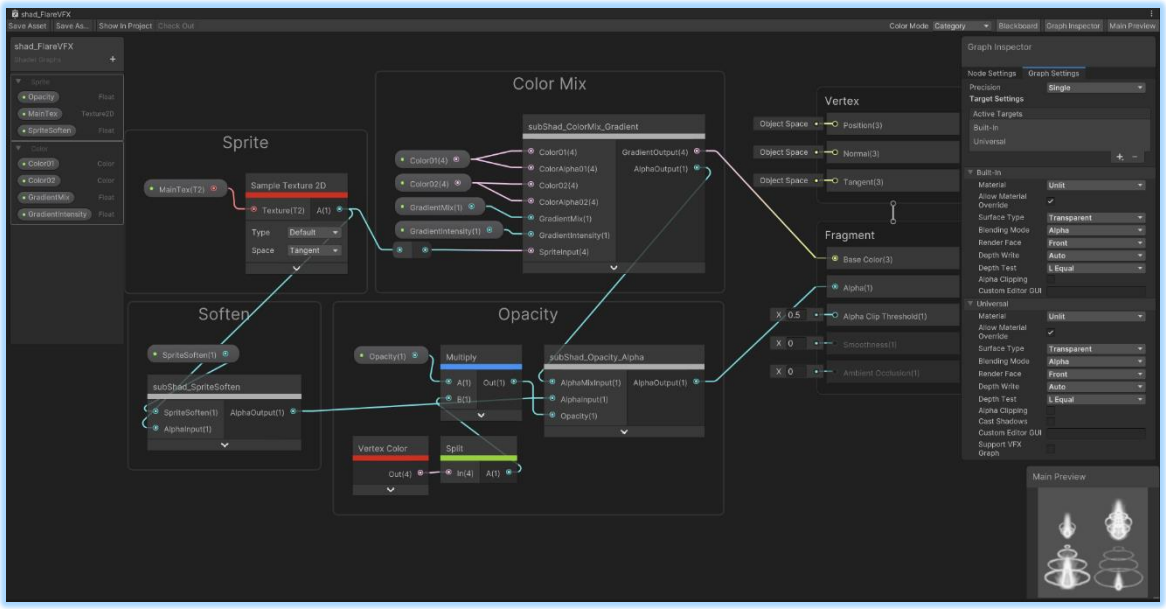
General order for **Projectiles**: **Projectile** < **Trail** < **Sparks**

NOTES

- This manual version was created based on Unity 2021.3.0f1. The version for Unity 2020.3 will have visual differences and will not support Built-In Render Pipeline, but anything else will work as described here.

FLAREVFX SHADER GRAPH

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



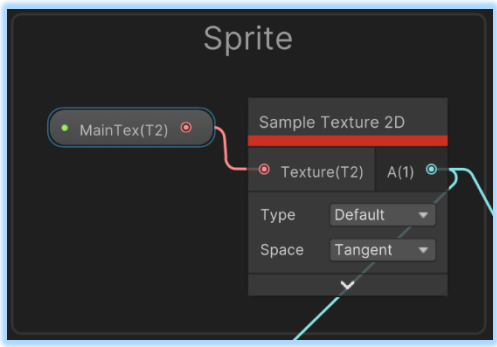
SPRITE

Input of the sprite.

This shader uses the **Alpha** channel from sprites to calculate color gradient and transparency.

Output: → Color Mix section

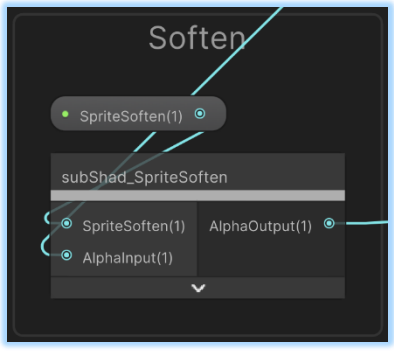
→ Soften section



SOFTEN

Makes the edge of the sprite smooth. See [SpriteSoften].

Output: → Opacity section



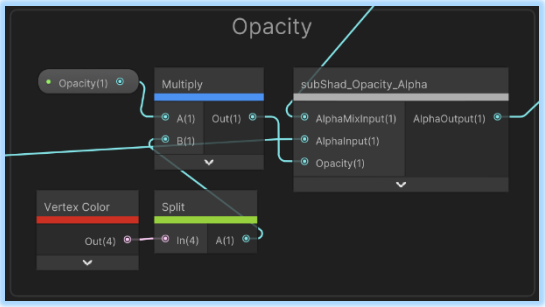
OPACITY

Combines **Alpha channel** from the **sprite**, the **vertex color** and **colormix alpha output**.

See [Opacity_Alpha].

Vertex Color Alpha channel is extracted with a **Split** and multiplied by the **Opacity.**

Output: → Alpha Material Output.

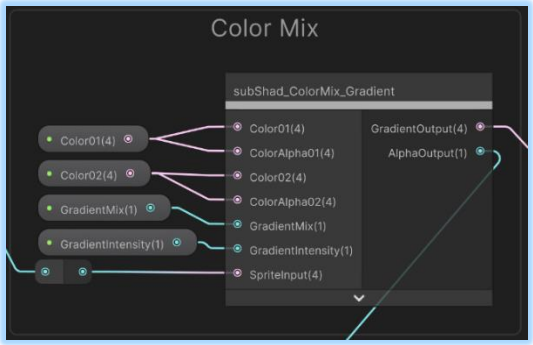


COLOR MIX

Consist on the [ColoMix_Gradient sub graph] and its variables.

Uses the values of a gradient (**Alpha channel**) to determine the 2 colors and their alpha values.

Input: Sample Texture → SpriteInput



Output: GradientOutput → Emission Material Output

Alpha Output → Opacity_Alpha SG

ON INSPECTOR

Opacity: the opacity of the entire shader. *(0 <-> 1)*

MainText: sprite slot.

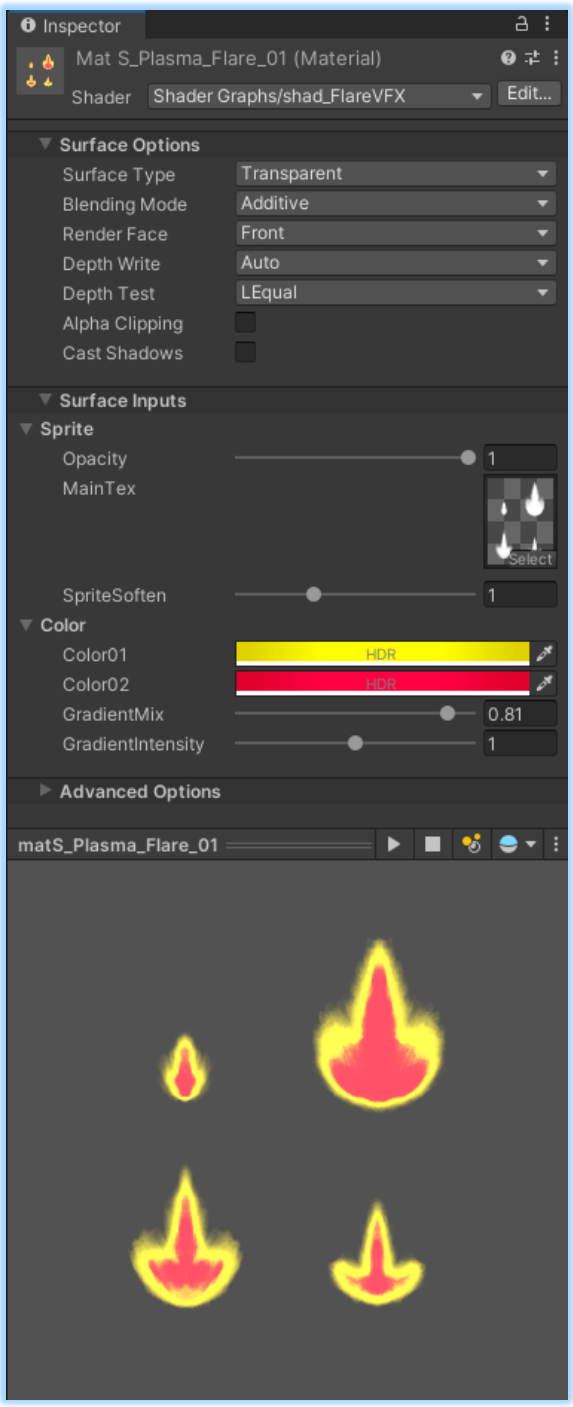
SpriteSoften: the smoothing and opacity of the edges of the sprites. *(0.3 <-> 2.5)*

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

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

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

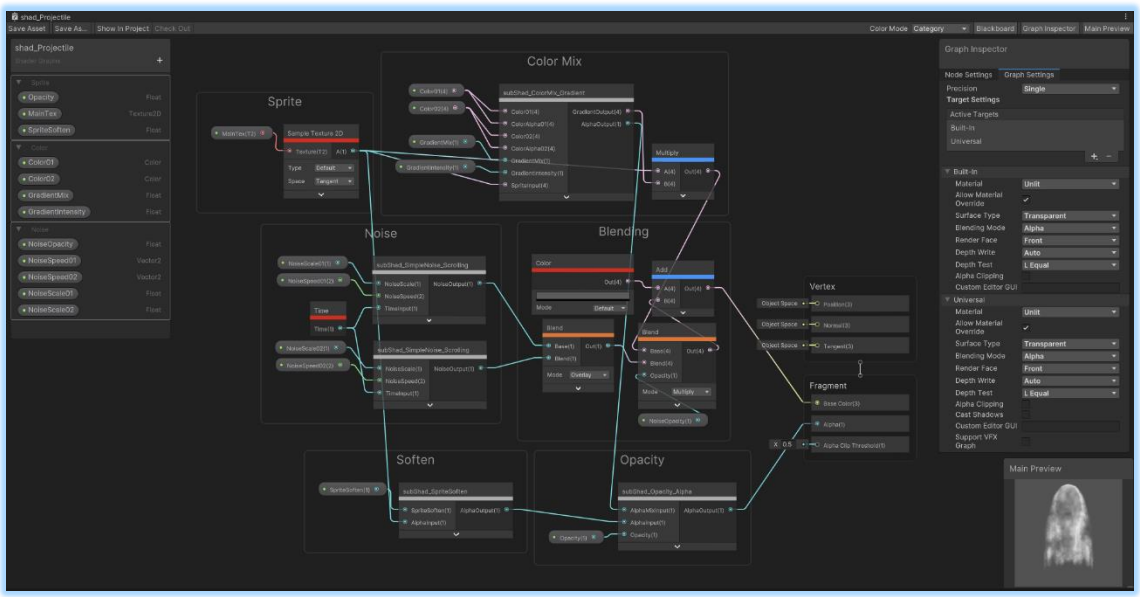
GradientIntensity: the amount of intensity of the inner color by lowering the high end of the gradient. *(0 <-> 2)*



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.



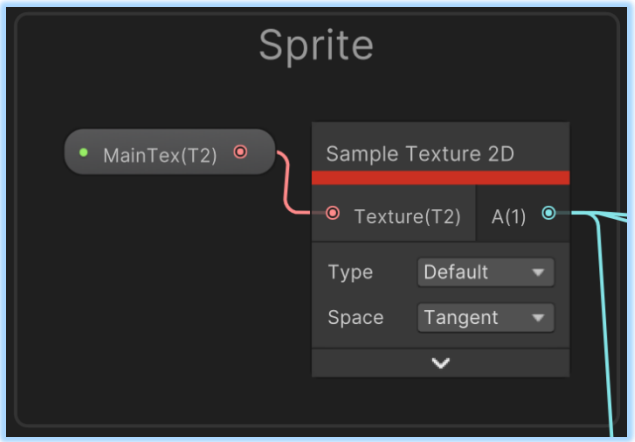
SPRITE

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

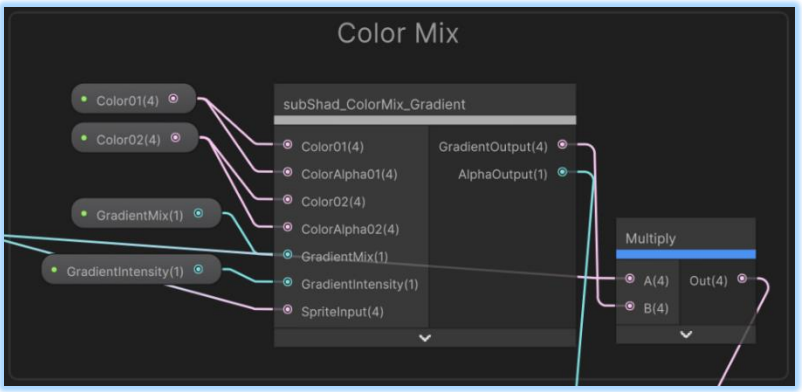
Consist on the **ColoMix_Gradient sub graph** and its variables.

Uses the values of a gradient (**Alpha channel**) to determine the 2 colors and their alpha values.

GradientOutput is multiplied by the sprite **Alpha channel** to correct some offset of the values from the sub graph.

Input: Sample Texture → SpriteInput
Sample Texture → Multiply

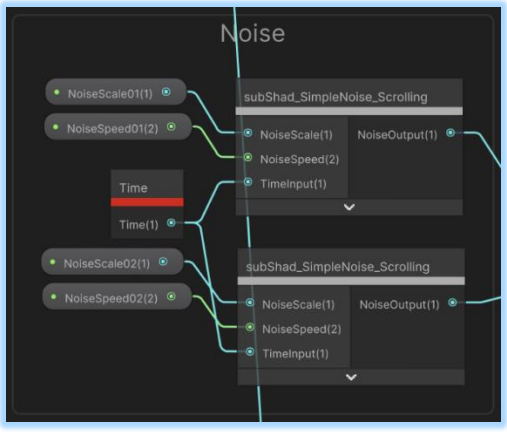
Output: GradientOutput → Blending section
Alpha Output → Opacity_Alpha SG



NOISE

It uses double **SimpleNoise_Scrolling sub graph** to generate 2 noise streams. The movement is independent and automated by **Time** and the speed controlled by 2 independent variables. The scale is controlled independently by two variables, one for each sub graph.

Output: →Blending section



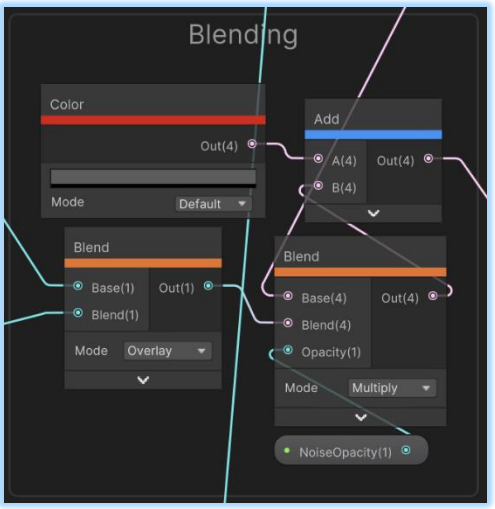
BLENDING

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 **Multiply** blending. A color variable is added to the output to adjust the brightness.

Input: SimpleNoise_Scrolling SG x2 → Blend 1
Color Mix section → Blend 2

Output: Add → Base Color Material Output

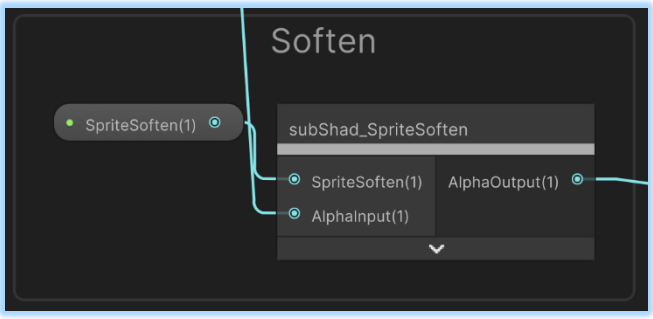


SOFTEN

Makes the edge of the sprite smooth. See [SpriteSoften].

Input: Sprite section → InputAlpha

Output: → Opacity section

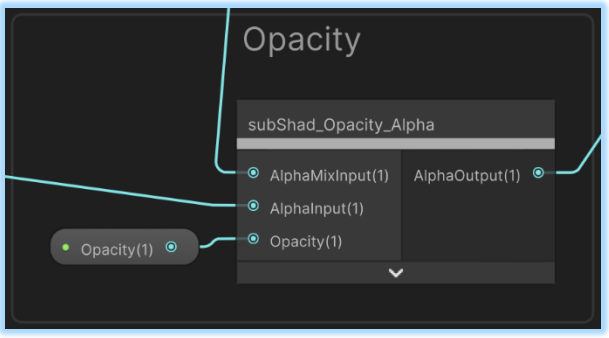


OPACITY

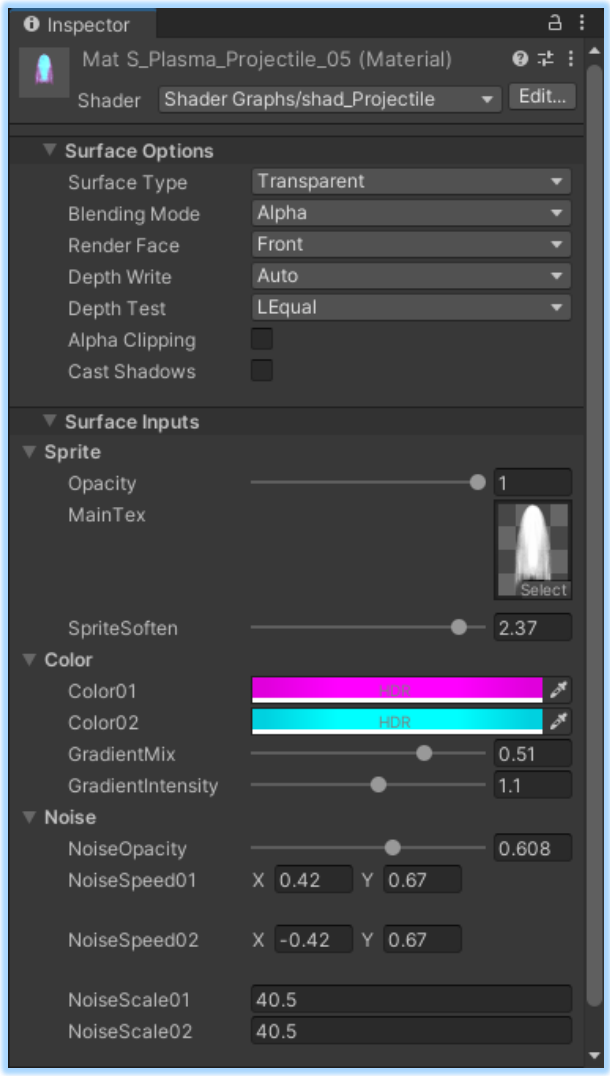
Combines **Alpha channel** from the **sprite** and **Colormix Alpha output**. See [Opacity_Alpha].

Input: ColorMix_Gradient SG → AlphaMixInput
SpriteSoften SG → AlphaInput

Output: →Alpha Material Output.



ON INSPECTOR:



Opacity: the opacity of the entire shader. *(0 <-> 1)*

MainText: sprite input.

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

(1 <-> 2.5)

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

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

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)*

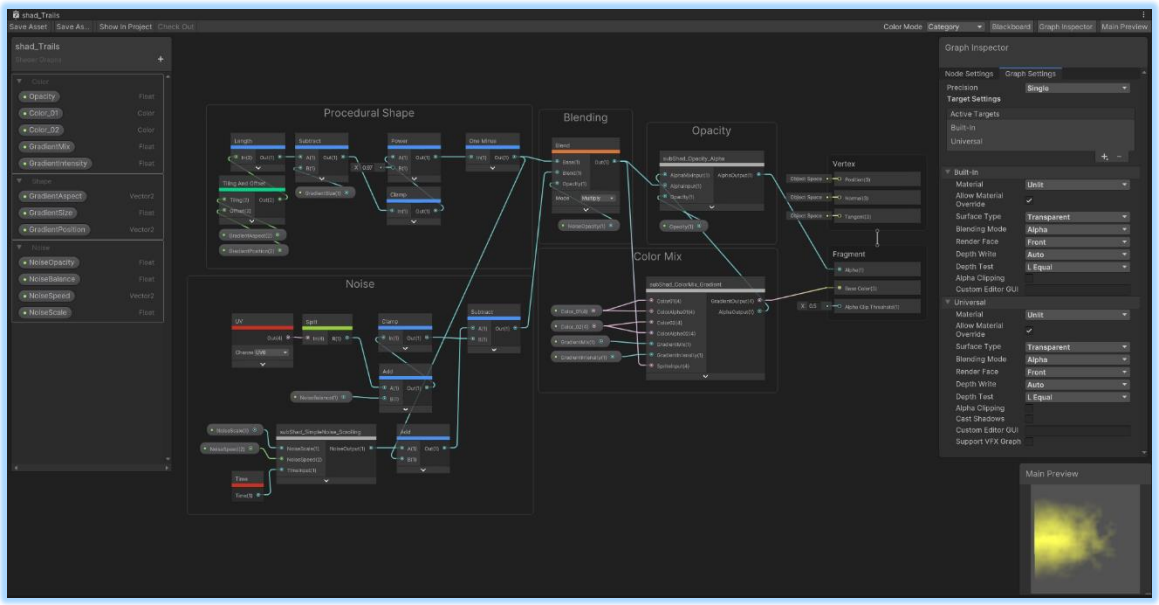
NoiseSpeed: speed/direction of the noise effect. There are two noise effects combined. Speed/direction parameters are independent between the two noises.

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

NoiseScale: the two scale variables of the **Noise** effects.

TRAILS SHADER GRAPH

Shader for **Trails** VFX



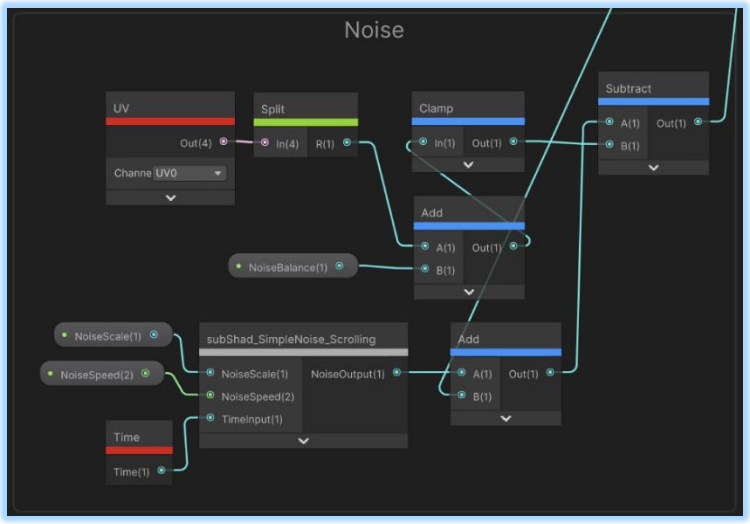
NOISE

It uses the **SimpleNoise_Scrolling** sub graph to generate noise as distortion effects, 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.

Input: Procedural Shape Section → Add to SimpleNoise_Scrolling output

Output: Subtract Node → Blending section



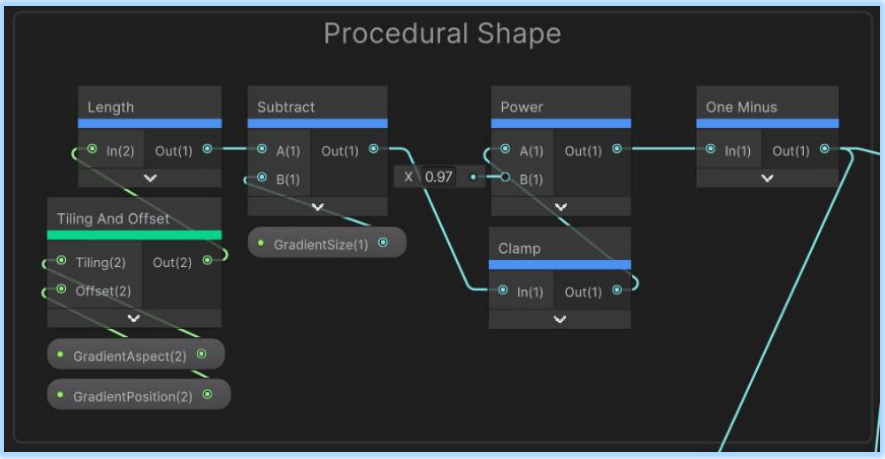
PROCEDURAL SHAPE

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: One Minus → Noise section

One Minus → Blending section

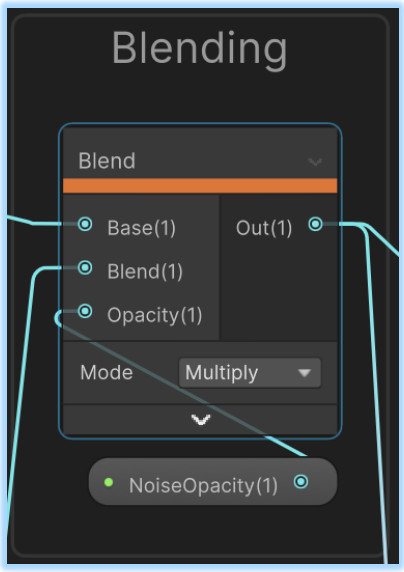


BLENDING

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

Output: Blend → Color Mix section

Blend → Opacity section



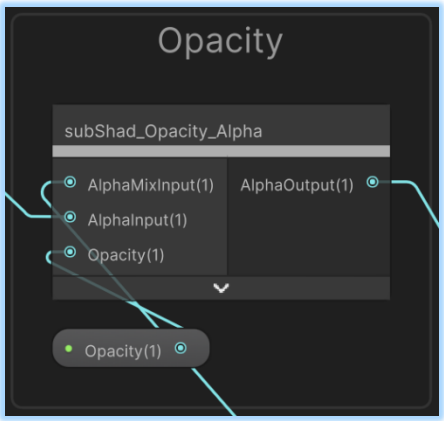
OPACITY

Combines **Alpha channel** from the **sprite** and **Colormix Alpha output**. See **Opacity_Alpha**.

Input: Blendinig Section → AlphaInput

Color Mix Section → AlphaMixInput

Output: AlphaOutput → Alpha Output.



COLOR MIX

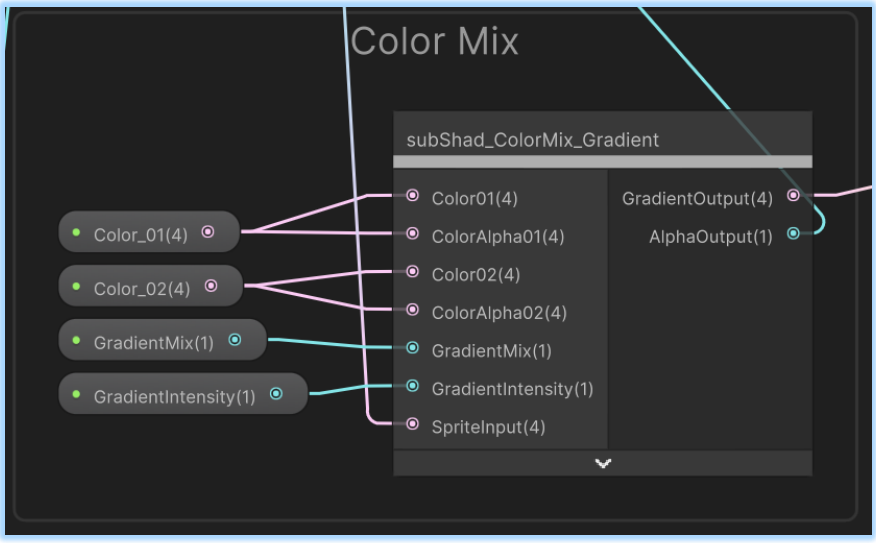
Consist on the **ColoMix_Gradient sub graph** and its variables.

Uses the values of a gradient (**Alpha channel**) to determine the 2 colors and their alpha values.

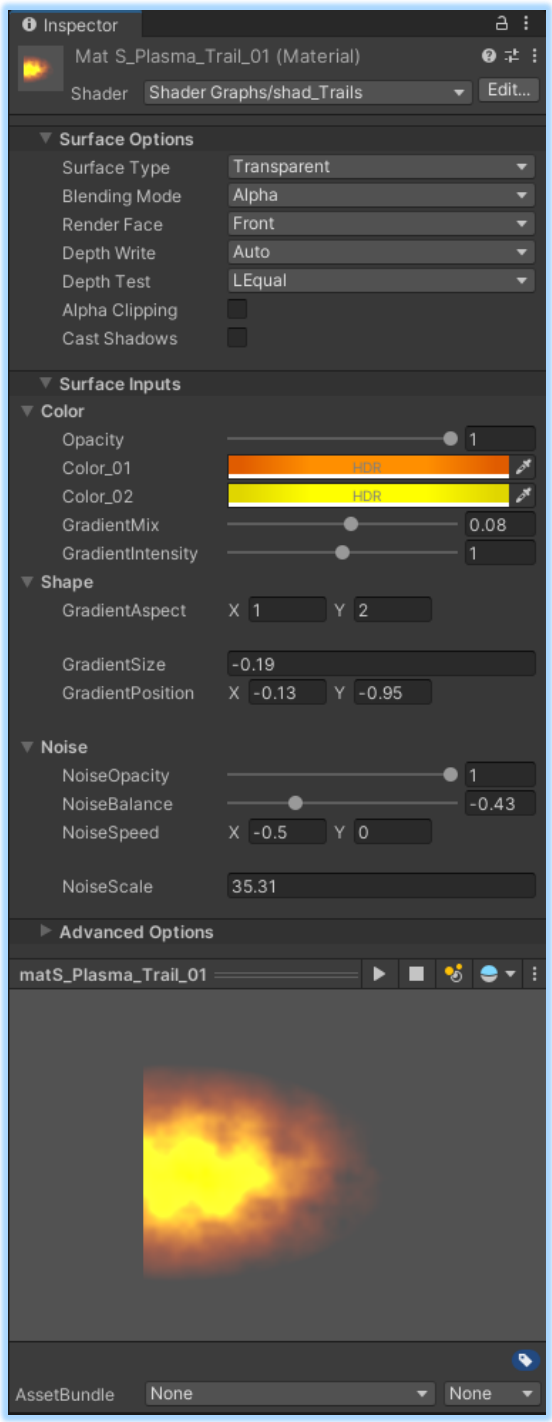
GradientMix and GradientIntensity variables are used to adjust the gradient.

Input: Blending Section → SpriteInput

Output: GradientOutput → Emission Output
AlphaOutput → Opacity Section



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).

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)*

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.

NoiseOpacity: the amount of distortion that is applied over the trail. *(0 <-> 1)*

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

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

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

SUB GRAPHS

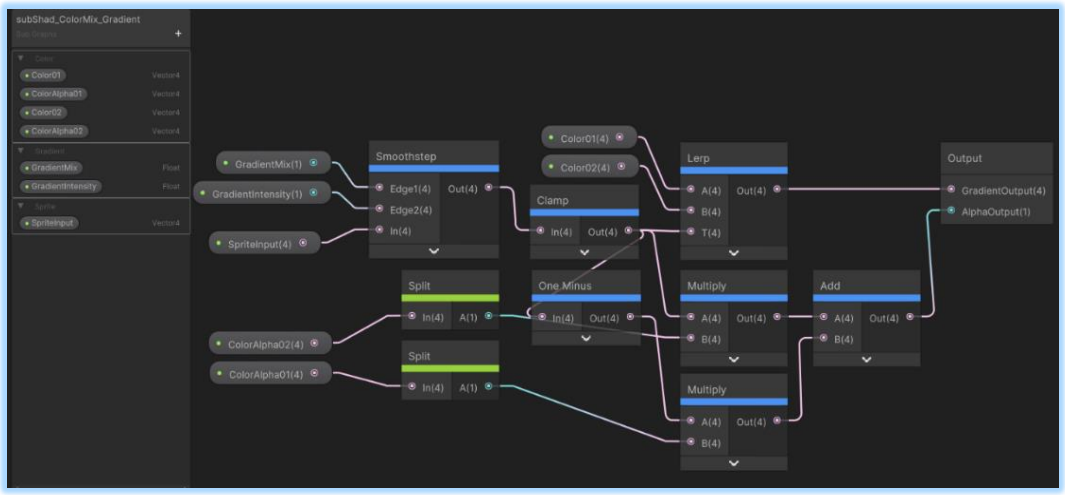
COLORMIX_GRADIENT SUB GRAPH

It uses **Lerp** node to mix two colors as a gradient like logic.

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

Alpha channel of **Color variables** is taken to calculate the opacity where each color has influence. The **Alpha channel** of both colors is multiplied by the output of the **Clamp node**, inverting one of the **Clamp outputs**. Then, both results are combined with an **Add node**.

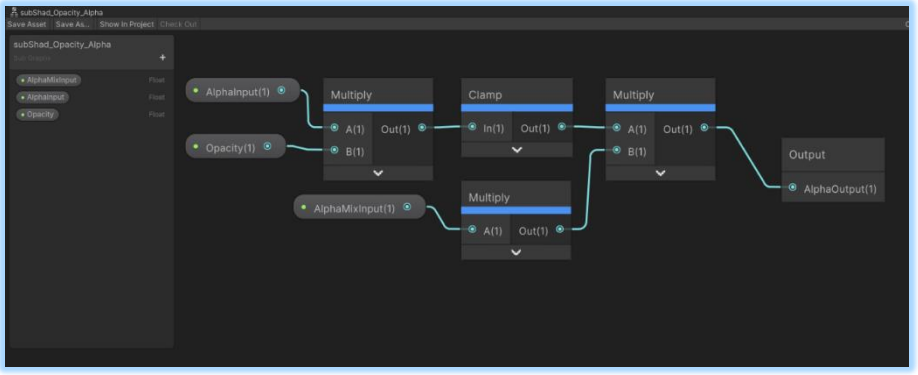
Lerp output and **Add output** are sent separated to the shader graphs.



OPACITY_ALPHA SUB GRAPH

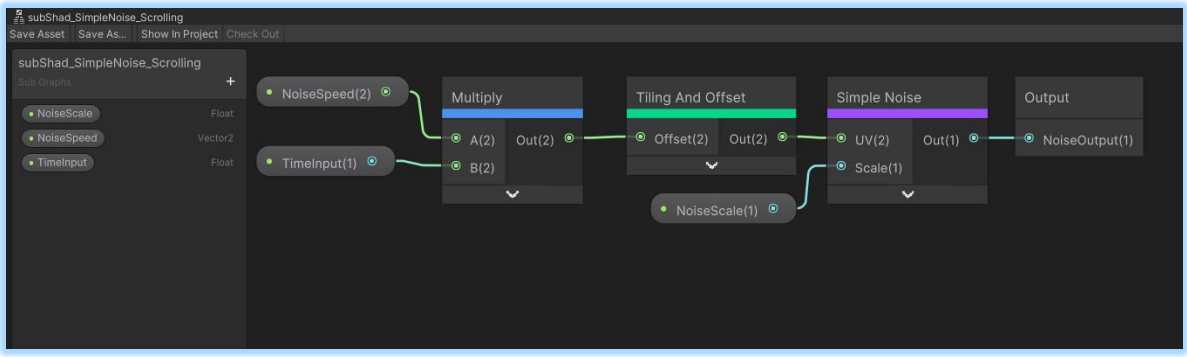
It uses **Multiply** node to control the opacity of the entire shader. **Clamp** node is to prevent negative values from **Multiply** node.

A second path is used to combine the **Alpha channel** from the **Color variables** with the opacity of rest of the shader. The input must be from the **ColorMix_Gradient sub graph**, where the **Alpha channel** is separated properly.



SIMPLENOISE_SCROLLING SUB GRAPH

It uses a **Simple Noise node** controlled by a **Tiling and Offset** node. The movement can be automated by **TimeInput variable**.



SPRITESOFTEN SUB GRAPH

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.