Corgi Fog Gradient

Distance Gradient Fog For URP

latest documentation here



Info

Quickly and easily drop in replace URP's linear fog with a color gradient.

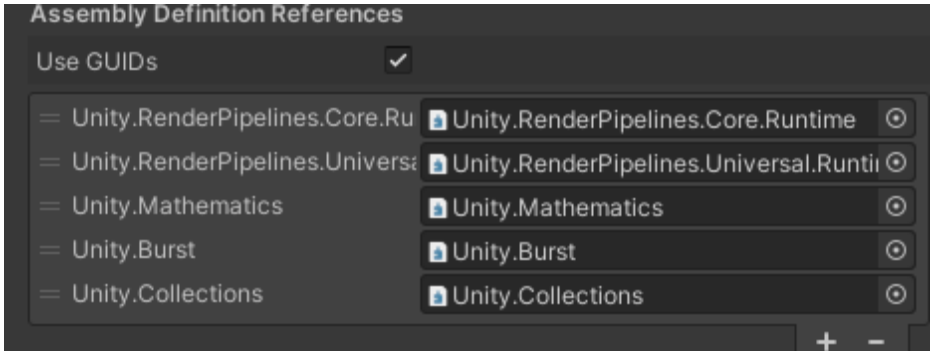
Features

- Very simple drop in. Add RenderFeature and add Post Process Volume Component and BAM, immediately have fog you can edit in real time.
- Supports a quick and easy to use post processing render mode AND a much more optimized shader macro that can be easily dropped into any HLSL shader.
- Highly optimized fog system, with some options for making it even faster if necessary.
- Supports both HSV blending for better visuals, and typical RGB lerps for slower devices.

PS: if Unity ever adds support for a 'final color' block in ShaderGraph, Corgi Fog Gradient already has a ShaderGraph node ready to drop in to support it, for ShaderGraph fog in any material.

Requirements

- Unity 2020.1 or higher. I recommend Unity 2021.2 or higher.
- Universal Render Pipeline package: `com.unity.render-pipelines.universal`
- Unity's Mathematics package: `com.unity.mathematics`
- Unity's Burst package: `com.unity.burst`
- Unity's Collections package: `com.unity.collections`



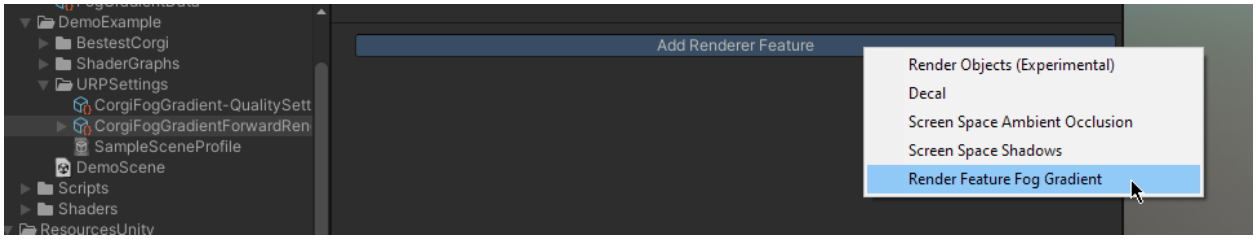
Getting Started

In order to use Corgi Fog Gradient, you must be using Unity 2020 or above with URP installed.

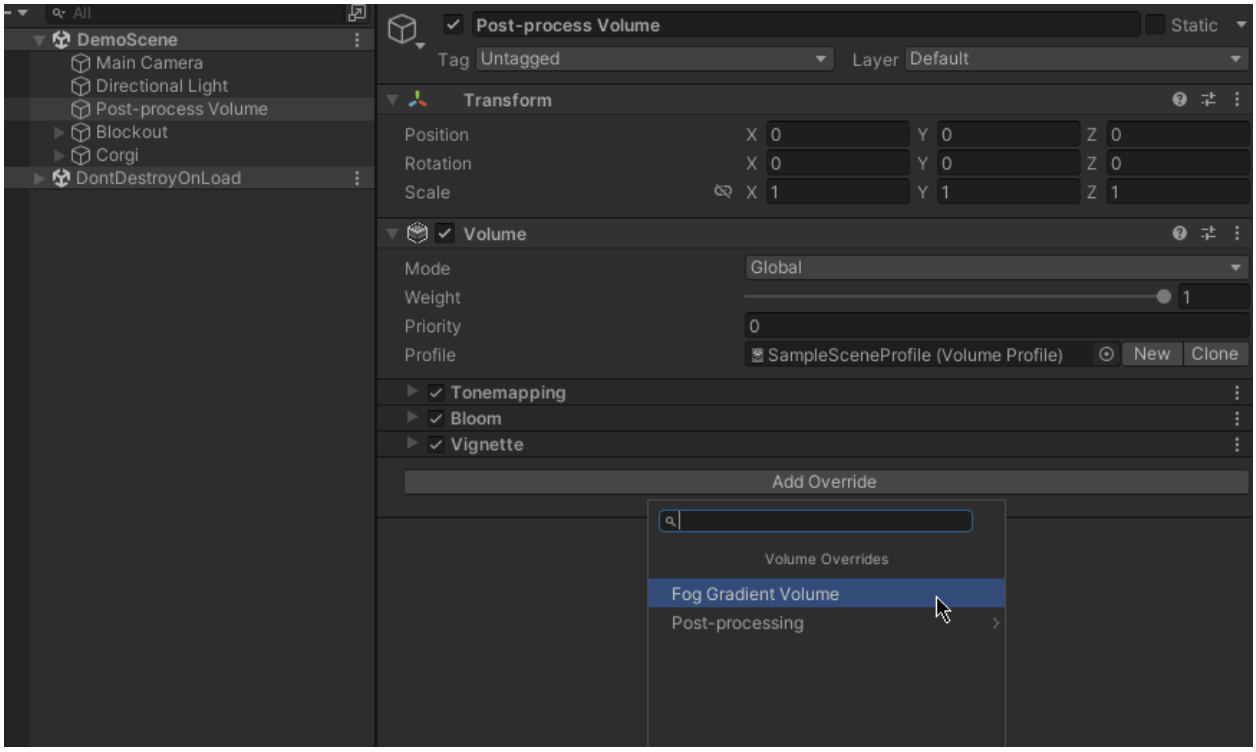
This system has two parts:

1. The Unity URP RenderFeature, to render the effect or setup material stuff.
2. The ‘post-process’ VolumeComponent.

So to get started, navigate to your project’s URP Renderer Data and add the RenderFeature.



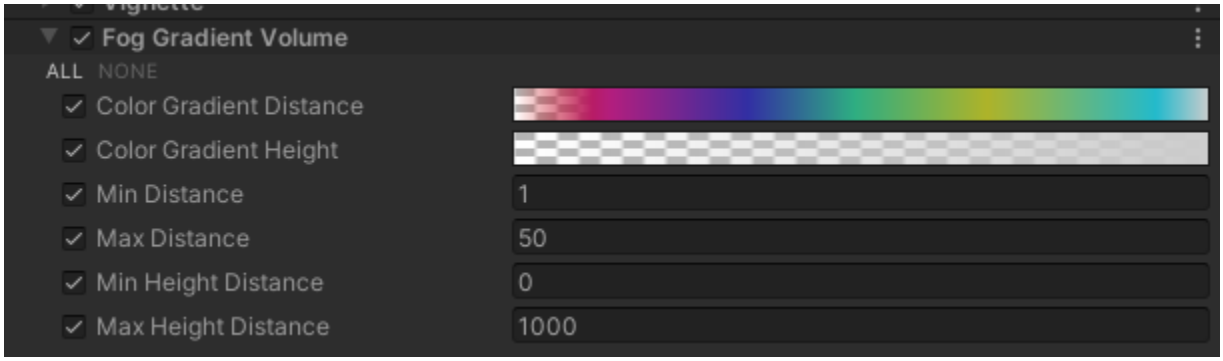
Once this RenderFeature has been added to your renderers, you can add a VolumeComponent to any ‘post-process’ Volume in your scene. In this example, I’m adding it to a global volume.



If nothing shows up for you, please verify that your camera and URP settings have post processing enabled.

Configuring the VolumeComponent

Here’s a screenshot of the post process volume component.



Color Gradient Distance

This Color Gradient is used for the x-axis in the internal fog texture. This is used for the distance from the camera to the object it’s looking at in front of it.

Color Gradient Height

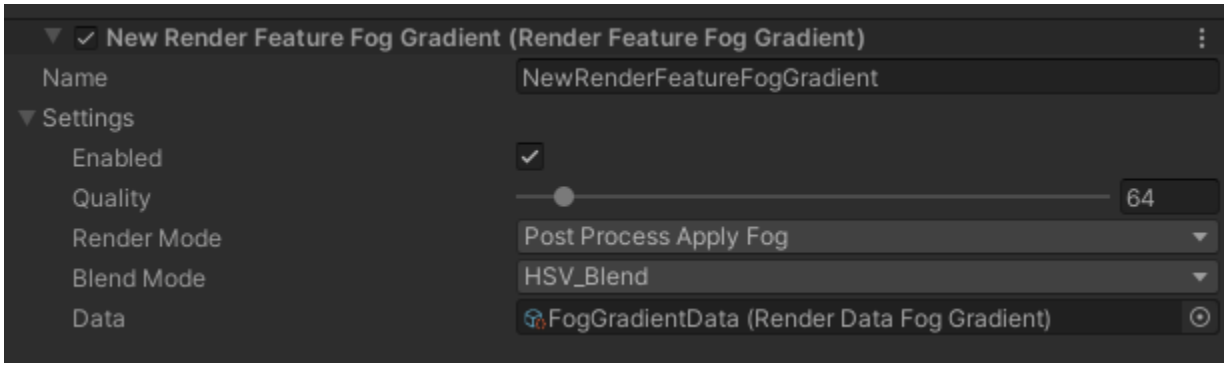
Similar to the distance gradient, this one is the y-axis in the internal fog texture, being used for height from the camera. Height in this context is referring to the |absolute value| of the difference between the camera’s world space y value and the world space y value of anything it is looking at. Useful for making tips of tall buildings fade into clouds, or pits of darkness below you.

Min/Max Distance, Min/Max Height Distance

These values control the distances and height distances when sampling the internal fog texture.

Configuring the RenderFeature

Here's a screenshot of the RenderFeature.



Quality

This slider controls the square dimensional size of the internally generated fog texture, which is generated based on the gradients defined. Distance is the x axis, height is the y axis, which are both blended together and sampled from to determine the final fog sample color. In the Unity Editor, this texture is updated every frame. However at runtime in standalone or other platform builds, this texture will only be updated if you call `RenderPassFogGradient`'s `TriggerUpdateTexture();`

Render Mode

There are two `FogGradientRenderMode` values: `PostProcessApplyFog` and `MaterialsApplyFog`. When set to the default, `PostProcessApplyFog`, the fog is applied as a full-screen post-process shader, drawn in the typical URP fashion by drawing a mesh over the whole screen with a special shader. When set to the alternative, `MaterialsApplyFog`, the system no longer draws anything extra. Instead, the fog properties (data texture, data floats, and keywords) are set and it is expected that the user of the plugin will manually sample the fog gradient texture themselves, using the provided hlsl macros or ShaderGraph subgraph.

Apply Blend Mode

There are two `FogGradientBlendMode` values: `HSV_Blend` and `RGB_Lerp`. When set to the default, `HSV_Blend`, the final fog color will be blended with the base color in HSV space. This can be computationally expensive depending on your target hardware, but usually looks much nicer than a typical color lerp. If you simply dislike the look, or really need the extra performance, an alternative blend mode is available. In these cases, try using the `RGB_Lerp` mode for faster and simpler blends.

For `ApplyBlendMode`, the cost of using `HSV_Blend` is on the GPU.

Gradient Blend Mode

Similar to the above setting, but for blending the gradients themselves.

There are two `FogGradientBlendMode` values: `HSV_Blend` and `RGB_Lerp`. When set to the default, `HSV_Blend`, the gradients will be blended in HSV space when creating the texture used by the plugin for the fog.

For `GradientBlendMode`, the cost of using `HSV_Blend` is on the CPU.

Render After Transparents

When this is true, the post process mode of the render feature will delay it’s execution until after transparent materials have been drawn.

Update Internal Texture Every Frame

When this is true, the internal texture that is created will be updated every single frame. If you do not want to waste time on this, either because you don’t need to blend between volumes or your game knows a better time to update, then disable this and call the static function `RenderFeatureFogGradient.TriggerTextureRefreshOnAllGradients();` instead.

Data

This is an internal reference data block. It should be automatically set and does not normally need to be changed. If you want to customize the asset, feel free to swap out the material used for doing the post process blit, which is referenced in here.

HLSL Reference

The file `FogGradient.hlsl` has all that you need in order to use Corgi Fog Gradient's fog in your custom URP shaders. Simply include it and call the macro `FRAGMENT_FOGGRADIENT(color, distance, height)` - you will also want to multi_compile for `_FOG_GRADIENT_ON` and `_FOG_GRADIENT_HSV_BLEND_ENABLED`.

```
// may be in a different location for you
#include "CorgiFogGradient/Shaders/Include/FogGradient.hlsl"

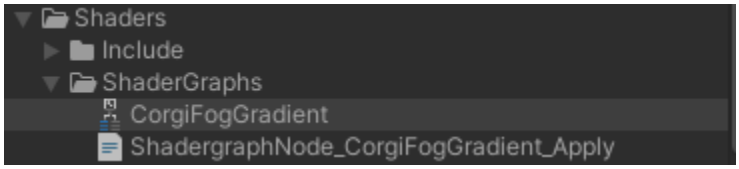
// multi compile somewhere
#multi_compile _ _FOG_GRADIENT_ON
#multi_compile _ _FOG_GRADIENT_HSV_BLEND_ENABLED

// in your fragment shader
FRAGMENT_FOGGRADIENT(color, distance, height)
return float4(color, your alpha);
```

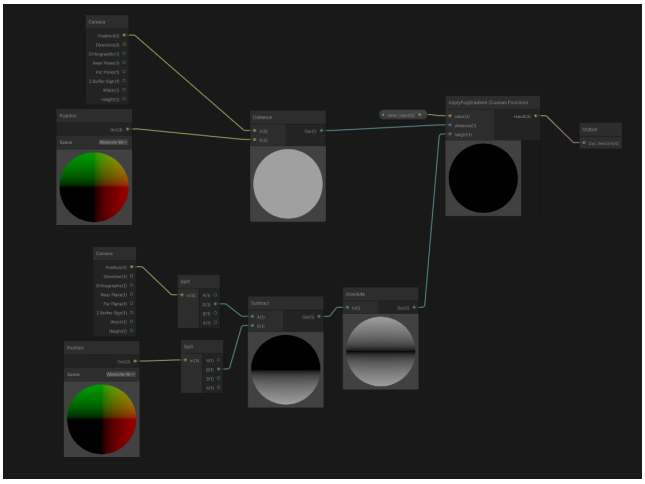
ShaderGraph Reference

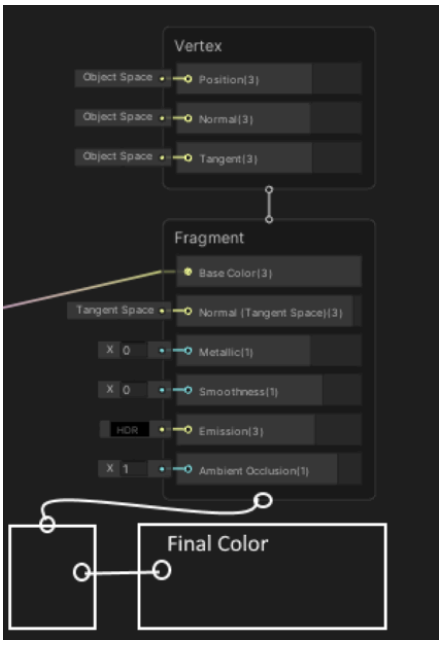
Corgi Fog Gradient includes a subgraph for using it within the ShaderGraph system. Unfortunately, as of the writing of this document, Unity has failed to include a ‘final color’ block in ShaderGraph. This means that any sort of fog system drawn per material within shadergraph is simply not possible without forking ShaderGraph. Despite this, I am hopeful that they will eventually add this very simple feature. If or when they do, I have provided a subgraph which you will then be able to simply drop in and use in any ShaderGraph shader.

Here is the subgraph, CorgiFogGradient. It includes a custom node within it to use the hlsl include file mentioned in the previous documentation section.



Inside should look something like this.





If or when Unity implements a **final color** block, like in my little example drawn here, you will want to drop in this subgraph between any lit fragment output and the final color output.

Note: For the HSV/RGB blend mode dropdown to work, your shadergraph will need a `global multi_compile` keyword for `_FOG_GRADIENT_HSV_BLEND_ENABLED`

You may also want to add in the `global mult_compile` keyword for `_FOG_GRADIENT_ON`

Contact Me!

If you run into any issues or just need some help, feel free to reach out and contact me.

- You can email me at coty@wanderingcorgi.com
- Or you can hit me up on Discord (Coty#2845).
- There's also a support discord here: https://discord.gg/n23MtuE