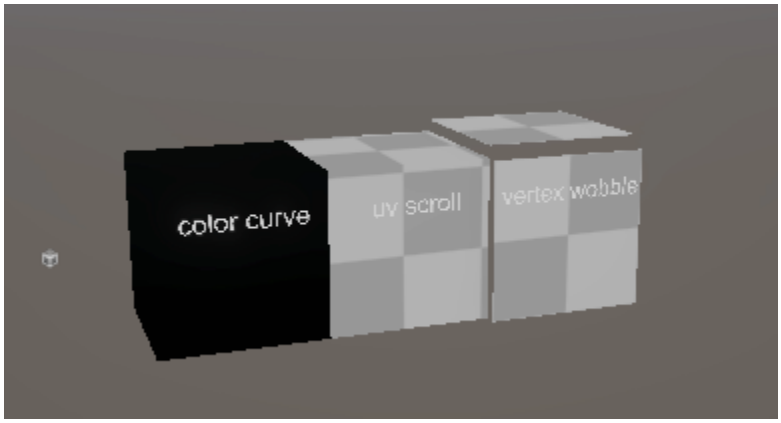
Corgi Shader Curves

Curve support in ShaderGraph and custom Shaders

[latest documentation here]



Info

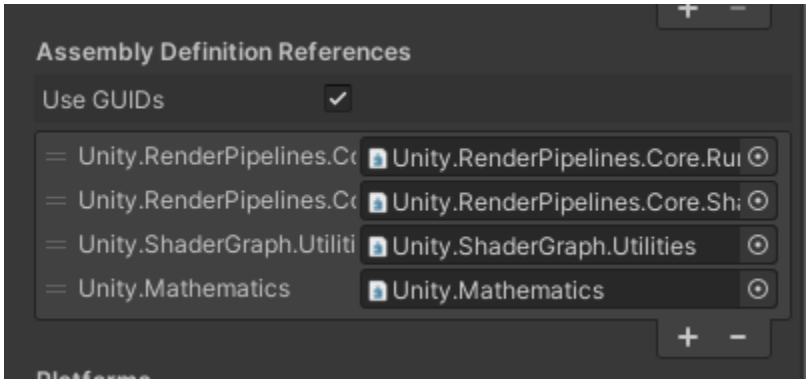
Easily use AnimationCurves in both ShaderGraph materials and your own custom Shaders.

Features

- Very simple drop in. Add `PfbCorgiCurveManager` to your scene and a sub-graph to your shaders and you're good to go!
- Supports as many curves as your platform can support (limited only by texture array limits) - the editor informs you of common mistakes.
- Highly optimized curve sampling. Don't worry about performance with these curves!
- Supports multiple precision and quality modes. If you're shipping on mobile, lower curve sampling quality and texture precision, if absolutely necessary!

Requirements

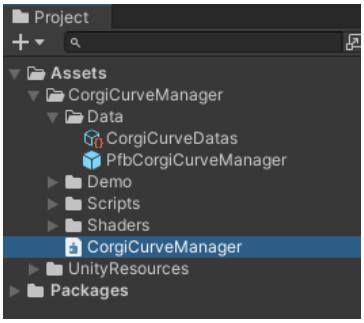
- Unity 2020.1 or higher. I recommend Unity 2022.3 or higher.
- Unity's Mathematics package: `com.unity.mathematics`



- Optionally, if ShaderGraph and URP are detected, the plugin can react to that for a better material editing experience.

Getting Started

In order to use Corgi Shader Curves, you must be using Unity 2020 or above. ShaderGraph is optional, but setup is different with or without it.



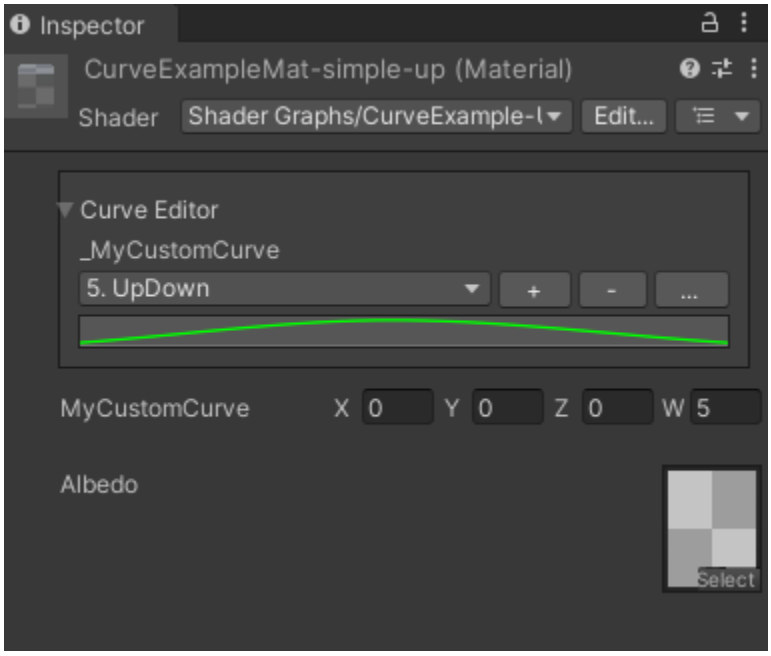
This system has three parts:

1. The manager prefab: PfbCorgiCurveManager. This must be placed in your scene. Only one should ever exist.
2. The scriptable object used for storing curves: CorgiCurveData - You only need one, it is referenced in the prefab. If one is not created, the system will try to create one for you.
3. A custom material editor CurveMaterialEditor. When you make a new shader, use this material editor (either directly, or inherit from it).

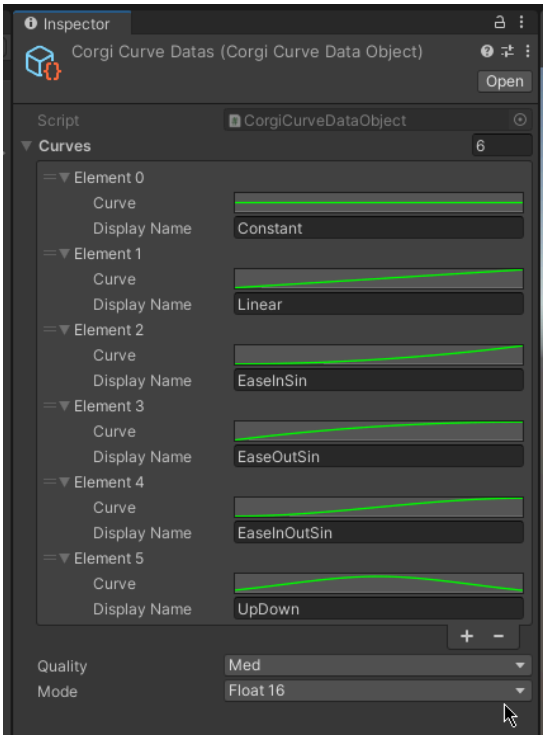
Editing Curves

When `CurveMaterialEditor` is being used, any material property that contains the word “curve” (not case sensitive) will be converted into the system’s property drawer. This property drawer will handle displaying and editing of curves, for each “curve property” detected. You can use the dropdown to select from your existing curves, or use the + or - to add or delete curves. The … button will select the curve data file. The plugin’s inspector will display the curve editor once per curve property you define, but it will not hide them from the default inspector.

As you might notice, the plugin is only currently using the W component of the vector - this is intentional, and means you can edit this value directly if you prefer (it’s the index into the curve array). X, Y, and Z currently do nothing, but may do something in the future.



Curve Data Object



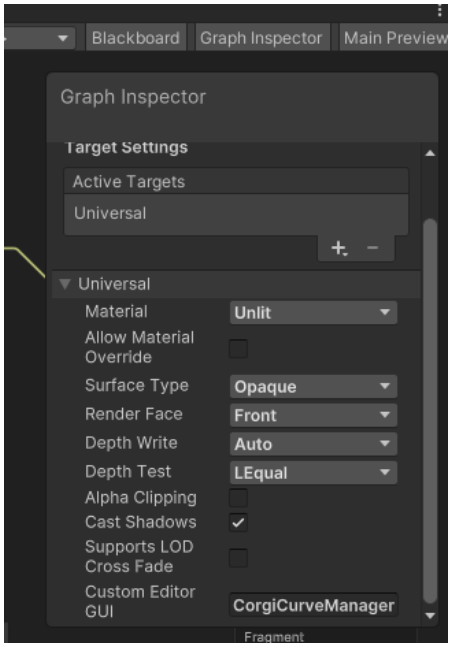
The scriptable object that contains the curve data also contains some settings for quality and precision. `Quality` refers to “curve sample quality” - it’s how many samples will be taken from the curve when generating the GPU data used by your shaders. If your curves are relatively simple, you can leave this on the lowest setting to save GPU memory. If your curves are complex, increase this value. The `Mode` setting refers to the precision of the GPU data used for representing the curve. If you see low quality sampling results, increasing precision may help. `Float8` (`fixed`) has one limitation: your curve values MUST be between 0 and 1. `Float16` (`half`) and `Float32` (`float`) do not have this limitation.

When you are done editing this file, you may have to press “**`Refresh`**” on the prefab `PfbCorgiCurveManager`’s inspector.

If any problems are detected in your curves (such as using Mode Float8 and a curve value goes below 0 or above 1) - a warning will be displayed in this inspector. Some warnings can be hidden in the inspector.

Shader Graph Usage

If you are using ShaderGraph, you'll need to somehow use Corgi Shader Curves' custom material drawer. The material drawer handles curve editing and selection within the Material Inspector windows.

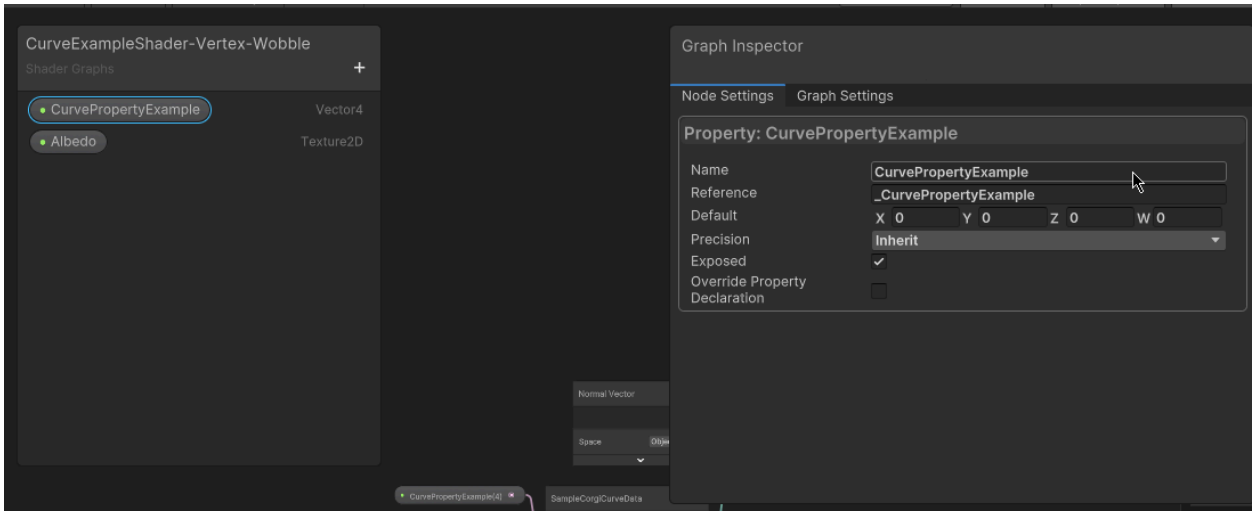


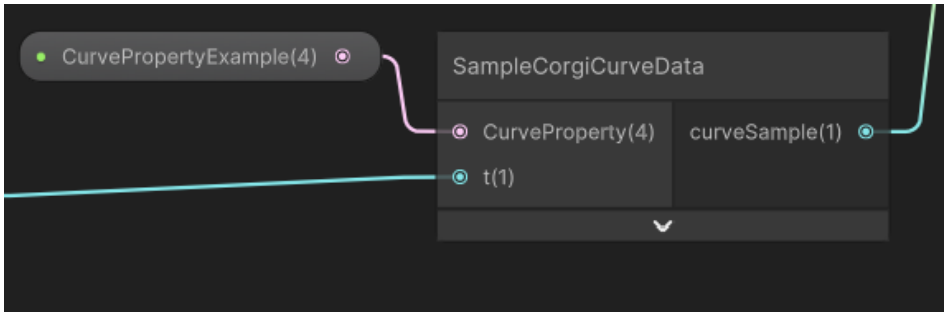
For ShaderGraph, it's pretty simple: Just paste the following text into your graph's "`Custom Editor GUI`":

```
CorgiCurveManagerEditor.CurveMaterialEditor
```

This will handle the drawer for you. If you already are using a Custom Editor GUI - you can either make yours inherit from ours, or you can call the functions from ours directly within yours.

To actually use a curve, you must define a `Vector4` and include "curve" in the name. For example, `_MyExampleCurve` or something along those lines.





Then, you can use our SubGraph “SampleCorgiCurveData” and input your custom Vector4 parameter into it. The `t` value refers to how far along into the curve you wish to sample. Sampling is always using `_LinearClamp`, so if you want to loop you’ll need to do it manually.

The Demo folder contains several ShaderGraph examples, please check them out for demonstration on how to use the subgraph!

HLSL Usage

The file CorgiCurveData.hlsl has everything you need for sampling from the curve data in custom HLSL shaders. Simply #include it and use the function

```
half SampleCurve(half4 curveProperty, half t)
```

If you would like to customize curve sampling, feel free to copy this file and edit it however you'd like. For example, you could replace `_LinearClamp` with `_LinearRepeat`, if that suits your needs. Or change the precision, if necessary.

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