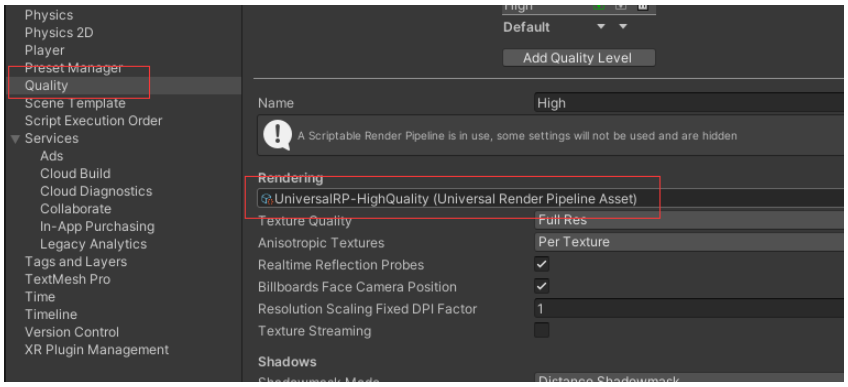
Clean Outline URP

Clean Outline URP is a post-processing solution for Unity URP pipeline. The outline shader will sample the depth and normal or nearby pixels to create outlines.
The general purpose of the shader is very simple, just to focus on creating outline without adding other effects to it. And it already considers some case like the distance of pixel and the threshold of the calculations.

Requirement:
Unity 2020.3x+ URP
Tested in Unity 2021.3 URP as well, should work fine.

How To Use:
Addindg a custom post processing effect in URP is a little bit complicated, as Unity URP's default post processing stack doesn't recognize your custom post volume which means you have to have a custom RendererFeature and RenderPass to support custom post volumes.
So you need to have a render feature that contains this outline post effect.



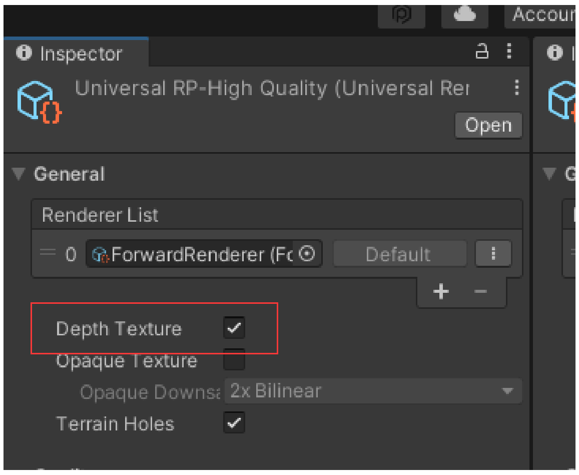
Physics
Physics 2D
Player
Preset Manager
Quality
Scene Template
Script Execution Order
Services
Ads
Cloud Build
Cloud Diagnostics
Collaborate
In-App Purchasing
Legacy Analytics
Tags and Layers
TextMesh Pro
Time
Timeline
Version Control
XR Plugin Management

Default
Add Quality Level

Name: High
A Scriptable Render Pipeline is in use, some settings will not be used and are hidden

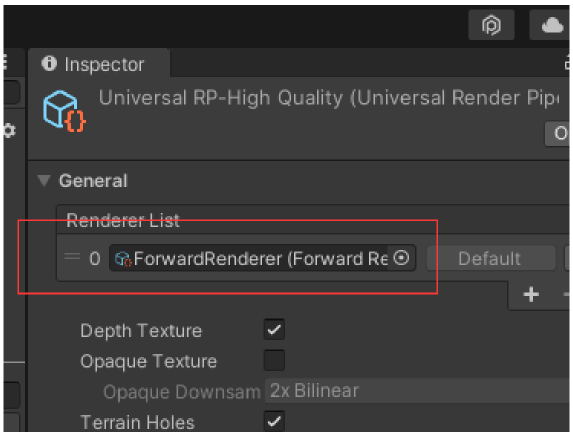
Rendering
UniversalRP-HighQuality (Universal Render Pipeline Asset)
Texture Quality: Full Res
Anisotropic Textures: Per Texture
Realtime Reflection Probes: ✓
Billboards Face Camera Position: ✓
Resolution Scaling Fixed DPI Factor: 1
Texture Streaming
Shadows

First open the project setting window and find Quality, where you'll find the URP asset of the qualities.

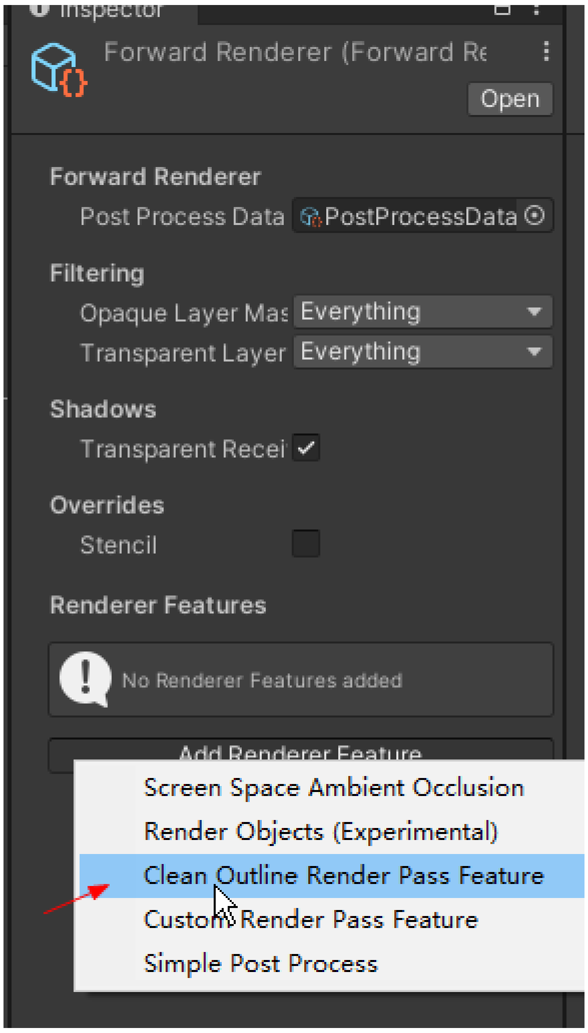


Inspector
Universal RP-High Quality (Universal Rer
Open
General
Renderer List
0 ForwardRenderer (Fc — Default
Depth Texture: ✓
Opaque Texture
Opaque Downsa: 2x Bilinear
Terrain Holes: ✓

Toggle the "Depth Texture" of the render pipeline asset, because we need the depth texture to sample for outline.

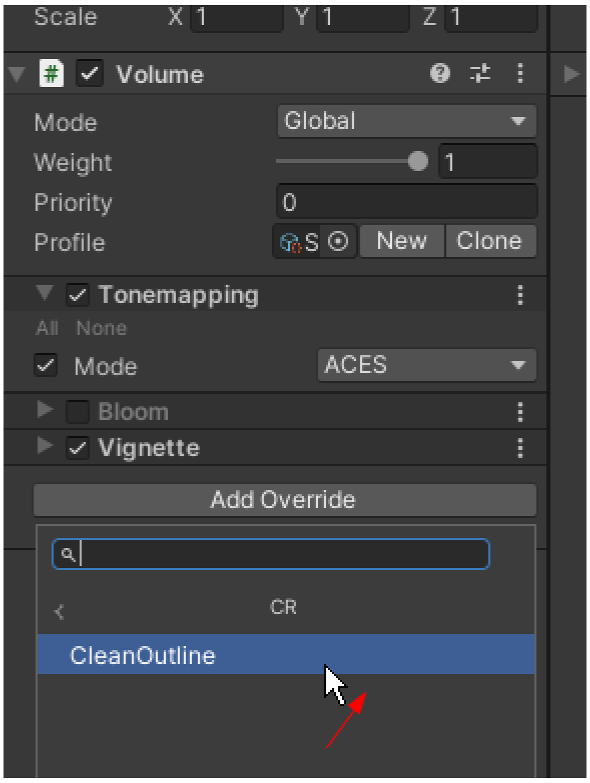

Inspector
Universal RP-High Quality (Universal Render Pip
General
Renderer List
0 ForwardRenderer (Forward Re
Default
Depth Texture ✓
Opaque Texture
Opaque Downsam 2x Bilinear
Terrain Holes ✓

And for the render pipeline asset, there will be a renderer data asset with it.


Inspector
Forward Renderer (Forward Re
Open
Forward Renderer
Post Process Data PostProcessData
Filtering
Opaque Layer Mas Everything
Transparent Layer Everything
Shadows
Transparent Recei ✓
Overrides
Stencil
Renderer Features
No Renderer Features added
Add Renderer Feature
Screen Space Ambient Occlusion
Render Objects (Experimental)
Clean Outline Render Pass Feature
Custom Render Pass Feature
Simple Post Process

Select the renderer data asset Click “Add Renderer Feature”.

The Clean Outline URP package already provides a custom RendererFeature for the outline post, so click “Clean Outline Render Pass Feature”.



Scale X 1 Y 1 Z 1

Volume

Mode: Global

Weight: 1

Priority: 0

Profile: S | New | Clone

Tonemapping

All None

✓ Mode: ACES

Bloom

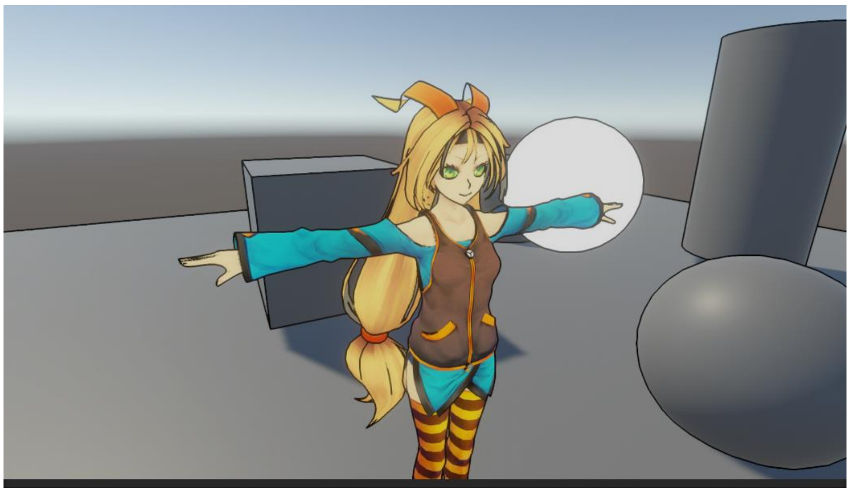
✓ Vignette

Add Override

CR

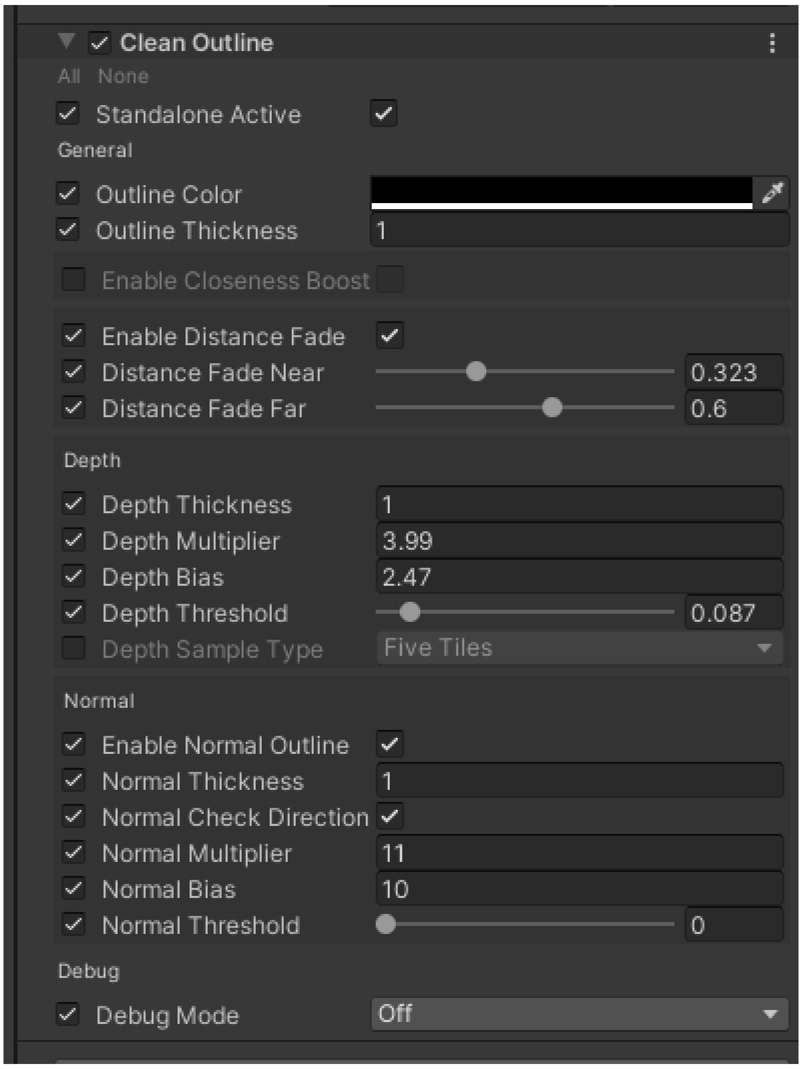
CleanOutline

Then open a game scene, find the volume of the scene.
Click “Add Override” and select “CR->CleanOutline” to add the outline volume to the volume.



Then you’d have a post outline effect.

Tuning The Parameters:

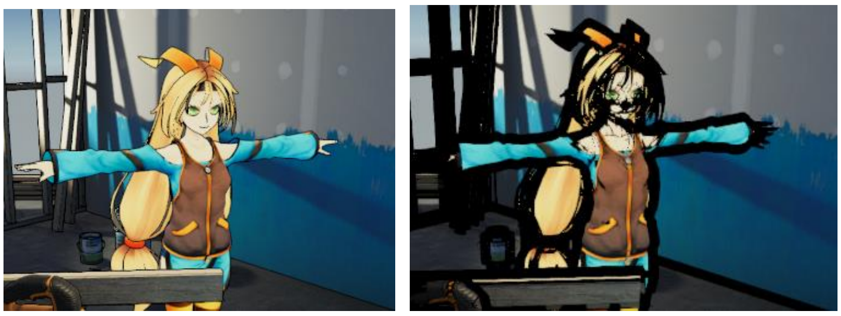

Clean Outline

All None

✓ Standalone Active	✓
General	
✓ Outline Color	■
✓ Outline Thickness	1
Enable Closeness Boost	
✓ Enable Distance Fade	✓
✓ Distance Fade Near	0.323
✓ Distance Fade Far	0.6
Depth	
✓ Depth Thickness	1
✓ Depth Multiplier	3.99
✓ Depth Bias	2.47
✓ Depth Threshold	0.087
Depth Sample Type	Five Tiles
Normal	
✓ Enable Normal Outline	✓
✓ Normal Thickness	1
✓ Normal Check Direction	✓
✓ Normal Multiplier	11
✓ Normal Bias	10
✓ Normal Threshold	0
Debug	
✓ Debug Mode	Off

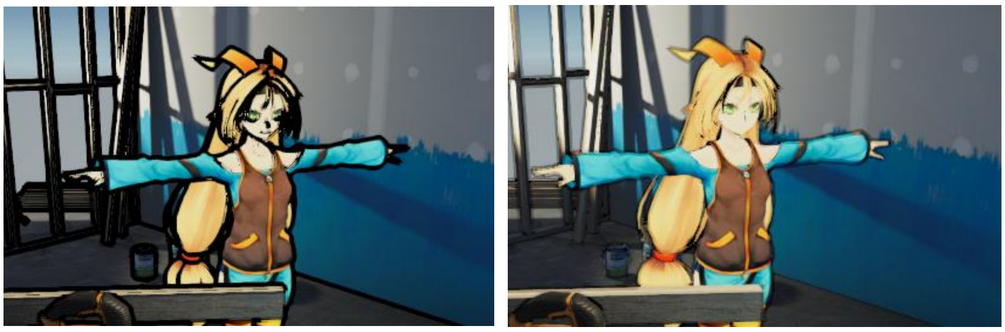
Standalone Active : A standalone bool value to enable/disable the outline effect.

First off to tune the parameters, you should have a general concept of thickness and strength. When outline shader shades a pixel on the screen, it will sample the pixel data and some other pixels around it.
So thickness means how much distance it will reach other pixels from the original pixel.

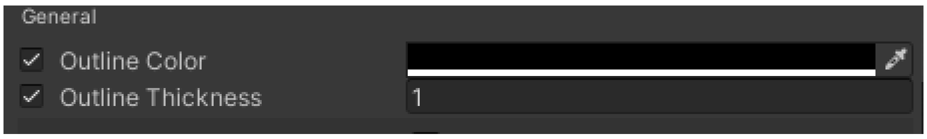


With larger thickness, outline shader will samples pixels more far away like the picture on the right. So it doesn’t mean that larger thickness is alwasy better, for the best value you should find a balance.

And strength means how much affect the outline result to apply to the final result.



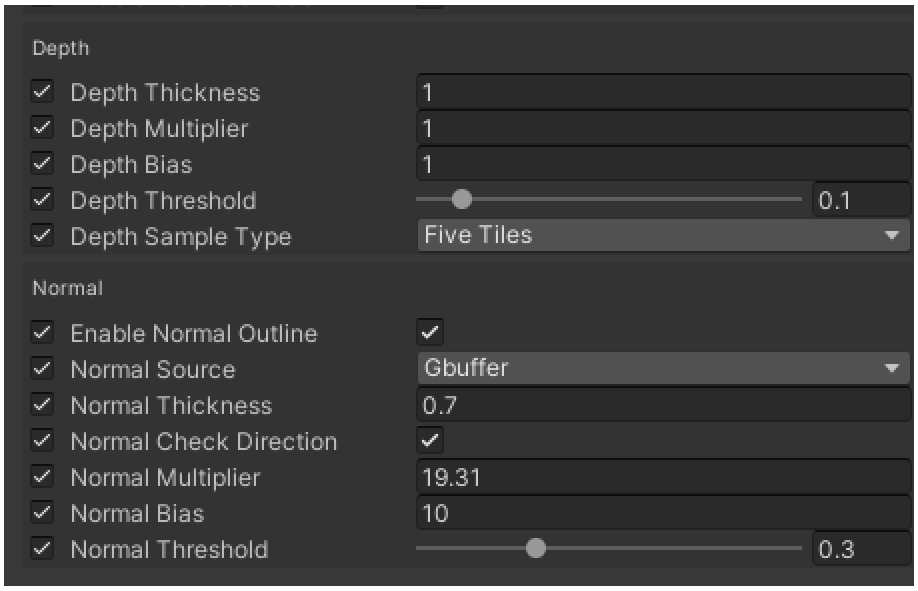
The compare of bigger and smaller strengths in the pictures. The outline becomes lighter with smaller strength. It appears to be thinner as the strength is calculated by exponentiation. So a curve strength would be give different result from different strength value by power value. There's no a global strength of outlines, but they can be modified separately and might be auto calculated in the shader through other parameters.



General

✓ Outline Color	■
✓ Outline Thickness	1

Outline Color : The color of outlines.
Outline Thickness : The global thickness of outlines, affecting both depth and normal outlines.



Depth

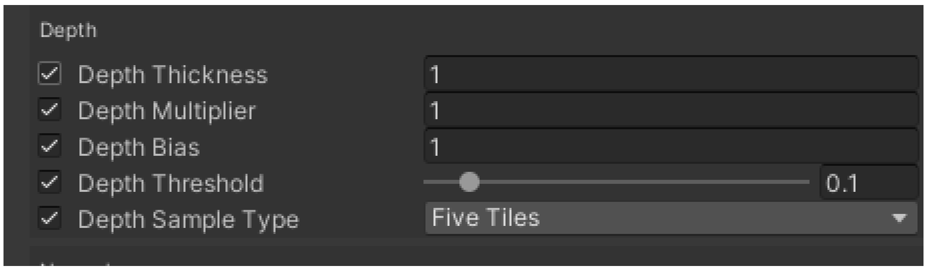
✓ Depth Thickness	1
✓ Depth Multiplier	1
✓ Depth Bias	1
✓ Depth Threshold	0.1
✓ Depth Sample Type	Five Tiles

Normal

✓ Enable Normal Outline	✓
✓ Normal Source	Gbuffer
✓ Normal Thickness	0.7
✓ Normal Check Direction	✓
✓ Normal Multiplier	19.31
✓ Normal Bias	10
✓ Normal Threshold	0.3

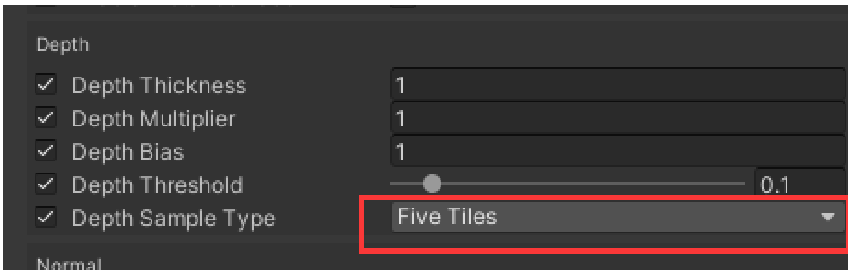
The outline effect include a depth calculation and a normal calculation.
Depth will always be calculated to create outline.
Normal can be optional because it only work good in lowpoly scenes, and for PBR objects with complicated normal maps, the normal outline will look dirty and messy.

Depth


Depth

- ✓ Depth Thickness: 1
- ✓ Depth Multiplier: 1
- ✓ Depth Bias: 1
- ✓ Depth Threshold: 0.1
- ✓ Depth Sample Type: Five Tiles

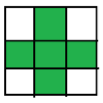
Depth Thickness : Thickness of the depth outline
Depth Multiplier/Bias : Affect the strength of depth outline.
Depth Threshold : After samples of depth difference, there will be a check if the result is under the threshold, and if it is it will be ignored. Depth will only consider the result over the threshold.


Depth

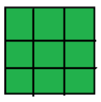
- ✓ Depth Thickness: 1
- ✓ Depth Multiplier: 1
- ✓ Depth Bias: 1
- ✓ Depth Threshold: 0.1
- ✓ Depth Sample Type: Five Tiles

Normal

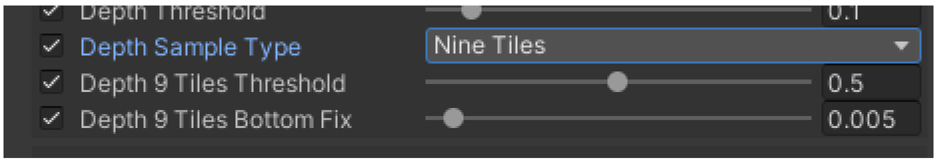
Depth calculation has 2 ways of sampling.



Five Tiles means it will sample the middle of the pixel with 4 other pixels by the side.

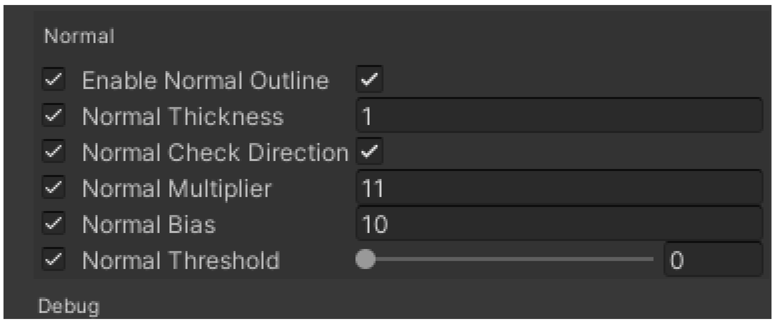


Nine Tiles will sample the middle and 8 other pixels around, and will be more costly.


- ✓ Depth Threshold: 0.1
- ✓ Depth Sample Type: Nine Tiles
- ✓ Depth 9 Tiles Threshold: 0.5
- ✓ Depth 9 Tiles Bottom Fix: 0.005

And in Nine Tiles way, there will be another threshold for checking, and a fix value to fix a weird effect which can be left as it is (0.005).
Best practice is to try to tune the values to see how they take effect, or read the outline shader to see how they work.

Normal


Normal

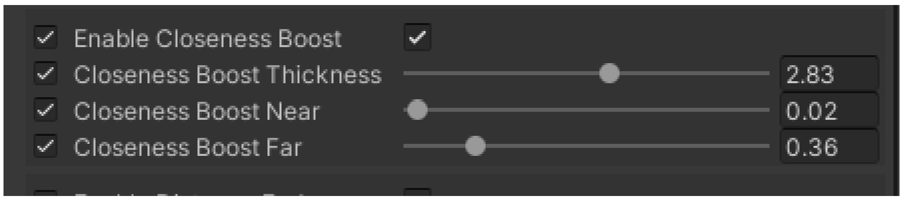
✓	Enable Normal Outline	✓
✓	Normal Thickness	1
✓	Normal Check Direction	✓
✓	Normal Multiplier	11
✓	Normal Bias	10
✓	Normal Threshold	0

Debug

Enable Normal Outline : You can use normal outline as an option, it can work good with lowpoly scenes. But for scenes with PBR normal maps, normal outline might give weird result so you might need to turn if off.
Normal Thickness : Thickness of normal outline.
Normal Multiplier/Bias : Affect the strength of the normal.
Normal Threshold : A threshold to ignore the result value lower than it for normal outline.
Normal Check Direction: If not toggled, and normal values will just be compared as raw vector values. And if toggled, normal will be seen as direction vectors. Different samples of normal vectors will be calculated into an angle between them to calculate further result.

Normal outline will always sample 5 tiles of pixels.

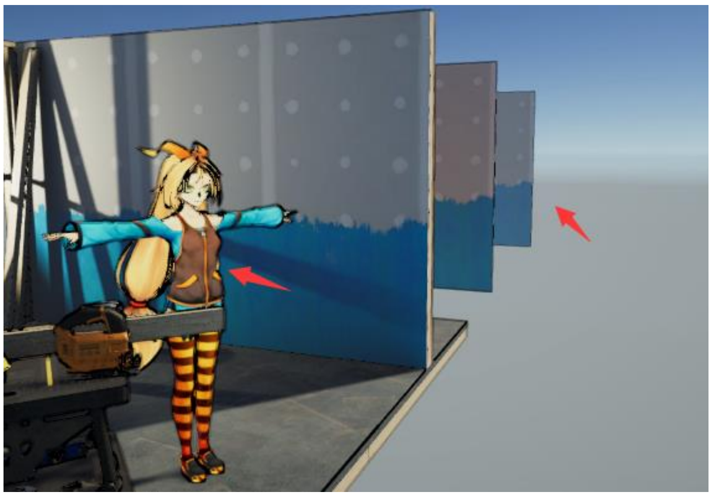
Some modifiers



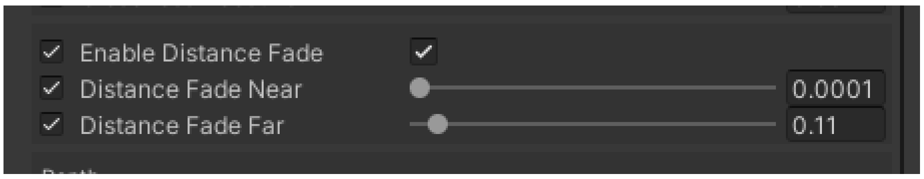
✓	Enable Closeness Boost	✓
✓	Closeness Boost Thickness	2.83
✓	Closeness Boost Near	0.02
✓	Closeness Boost Far	0.36

Closeness boost will add more thickness if the pixel is close enough. So in cases like that you want further distance has small thickness, and close distance has larger.
But please note that the thickness means how far distance it will sample near by pixels, and it cannot sample less than 1 pixels. So boost thickness might not often work good, you might turn it off if you can find a good result.

No Closeness Boost for thickness.

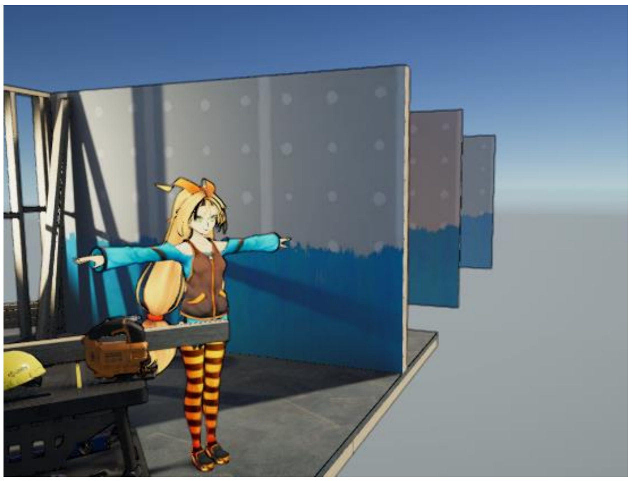


Turn on Closeness Boost, the thickness of close objects appear thicker.

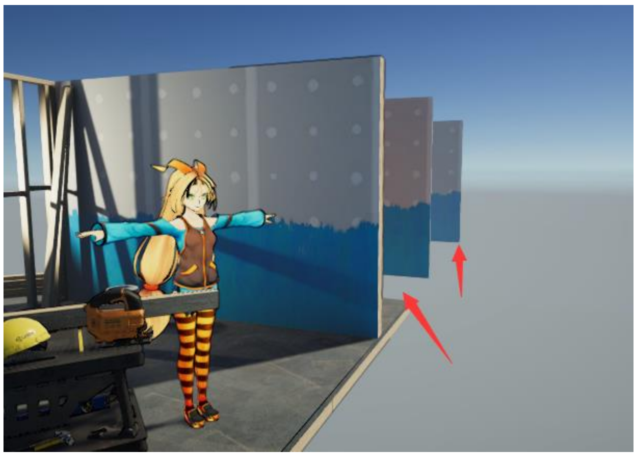


✓ Enable Distance Fade	✓	
✓ Distance Fade Near		0.0001
✓ Distance Fade Far		0.11

Distance fade will consider that if the pixel's distance (depth) is too far, it's strength will be seen as 0. So that distanced objects will have no or less outline effect.

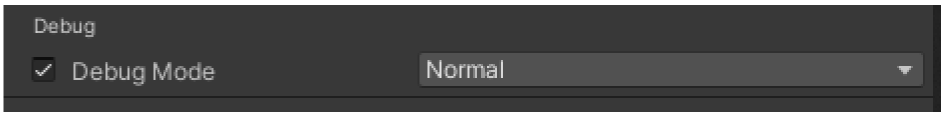


No Distance Fade outline.



Turn on Distance Fade, the outline strength almost fade to 0 at a distance.

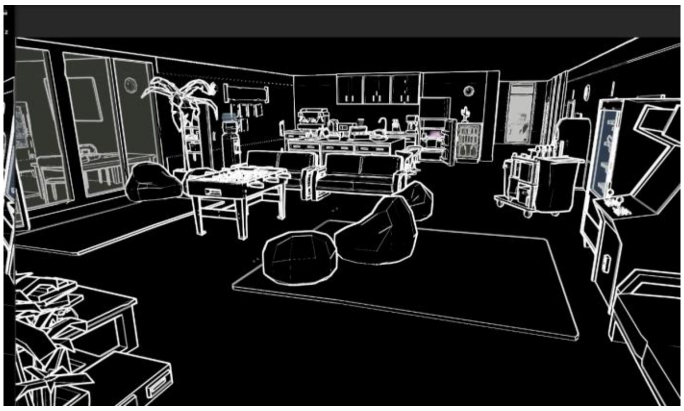
Debug Mode



Debug

☑ Debug Mode   Normal

Turn on debug mode to see how depth and normal outline work standalone in the game view.



Best Practice and known issues :

Renderer Feature

In practical cases, you probably should have your own custom renderer feature to draw custom post processing effects.

So you might need to implement the outline custom pass to your own render feature. There's a CleanOutlineRenderPassFeature.cs script in the package that can be used as reference. It's not suggested to use CleanOutlineRenderPassFeature.cs directly in a final project as with more custom passes, there will be more overhead for your game.

And if you need reference of making a custom renderer feature for custom post effects all together. You could check the link below

https://github.com/togucchi/urp-postprocessing-examples/tree/e120c442a24f79acea6d0ed26a2924e3dc724f58

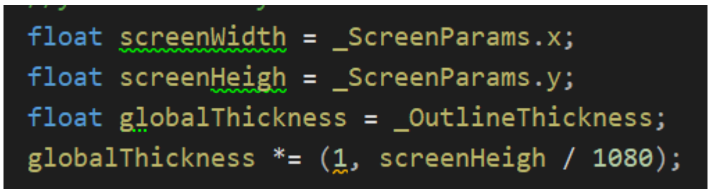
Scene View

Clean Outline URP only shows result in Game View, and not affecting Scene View as I haven't figured out how to let it work correctly in Scene View. So you might need to focus more on Game View to tune the effect.

Screen Resolution

You might also find it that with smaller resolution, the outline will look very thick which might be inevitable as distance to sample around each pixel can't be lower than 1. While with larger resolution outline will look too thin.

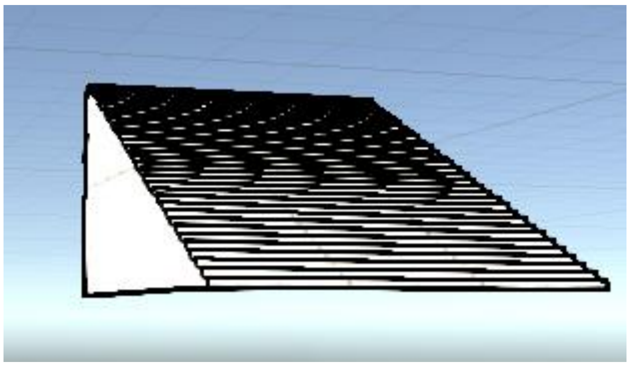
I've add some workaround try to match all resolution in CleanOutline.shader. You can have more custom control for different resolution so that large resolution will get more thickness.



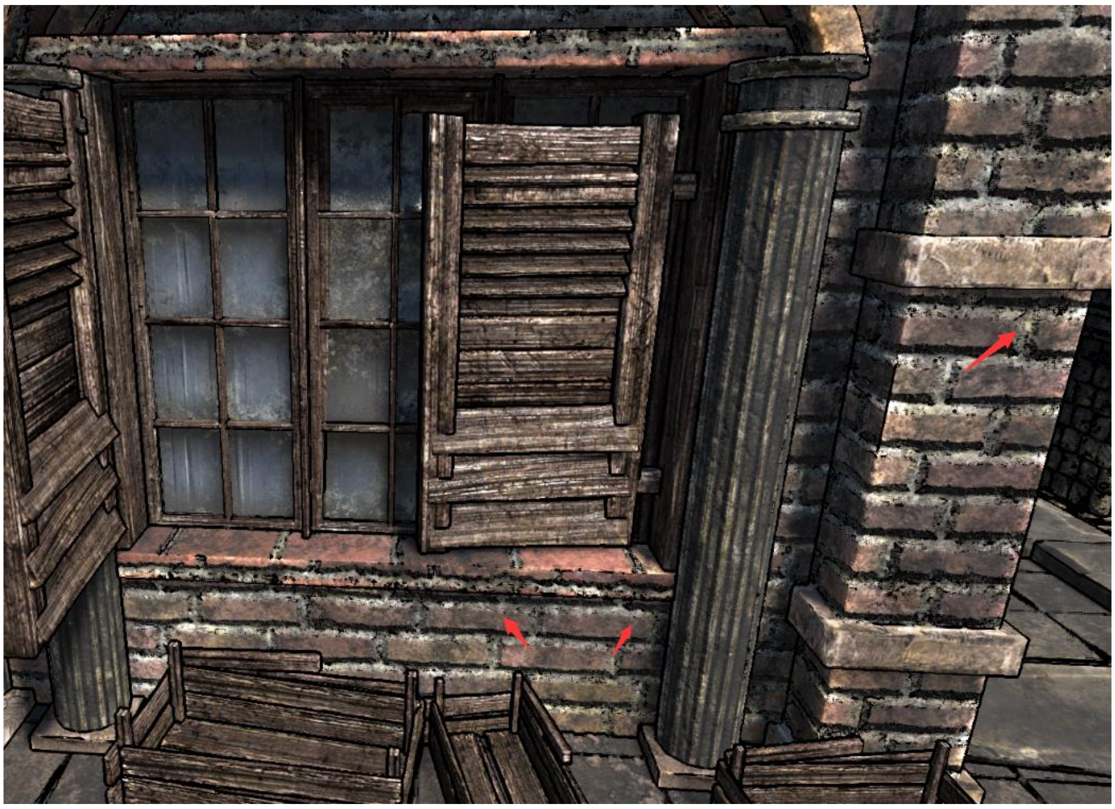
```
float screenWidth = _ScreenParams.x;
float screenHeigh = _ScreenParams.y;
float globalThickness = _OutlineThickness;
globalThickness *= (1, screenHeigh / 1080);
```

Clustered Outlines

Sometimes, clustered sudden changes of surfaces and distant objects might give a dense result of outlines.



For example the stairs with too many steps. So you might need to use less steps stair, or use LOD to reduce objects complicated surfaces at range if you want to use fullscreen outline.



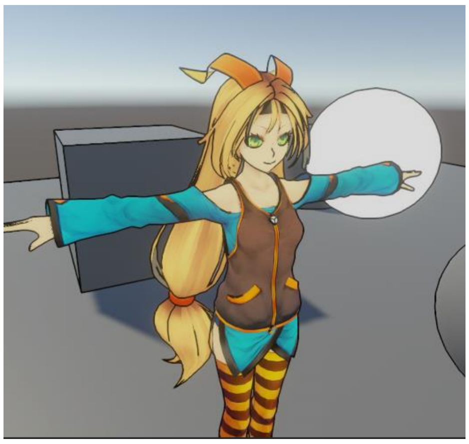
Normal maps

Normal map of object's materials might give more normal outline effect on the surface of objects. So if you don't want this fuzzy effect just disable normal outline especially with PBR objects.

"DepthNormals" pass of model shaders

To make outline works correct. The fullscreen shader needs a _CameraNormalsTexture and a _CameraDepthTexture.

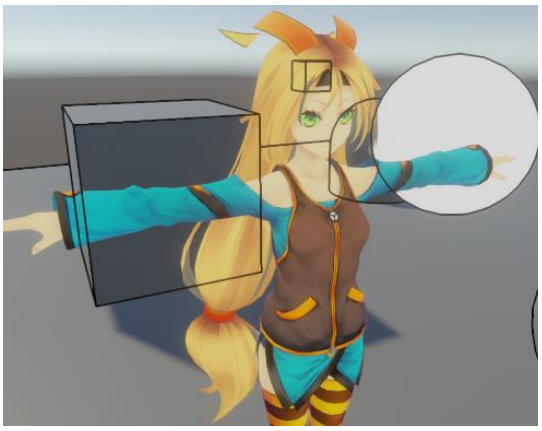
And URP needs your models in the scene to have a "DepthNormals" pass to correctly draw the textures on the screen.



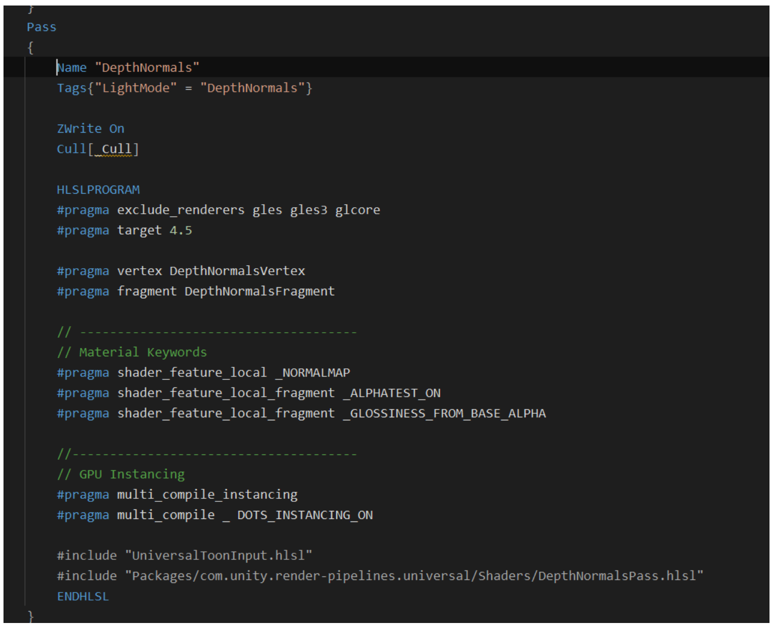
For example ,this Unity Chan's URP shader comes from this link.

https://github.com/EddiePrince/UTS3ToonShaderURP

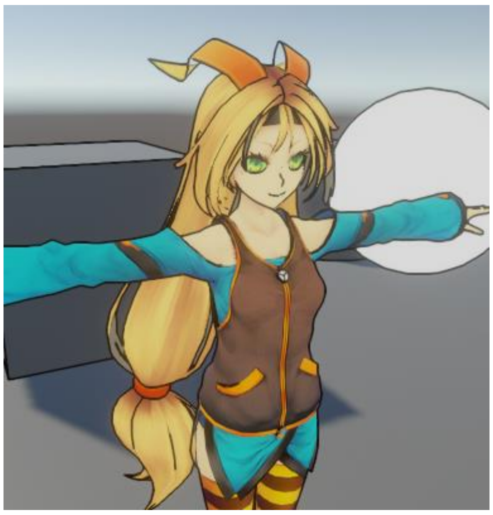
The shader UniversalToon.shader of it lacks of a "DepthNormals" which makes the outline result like this



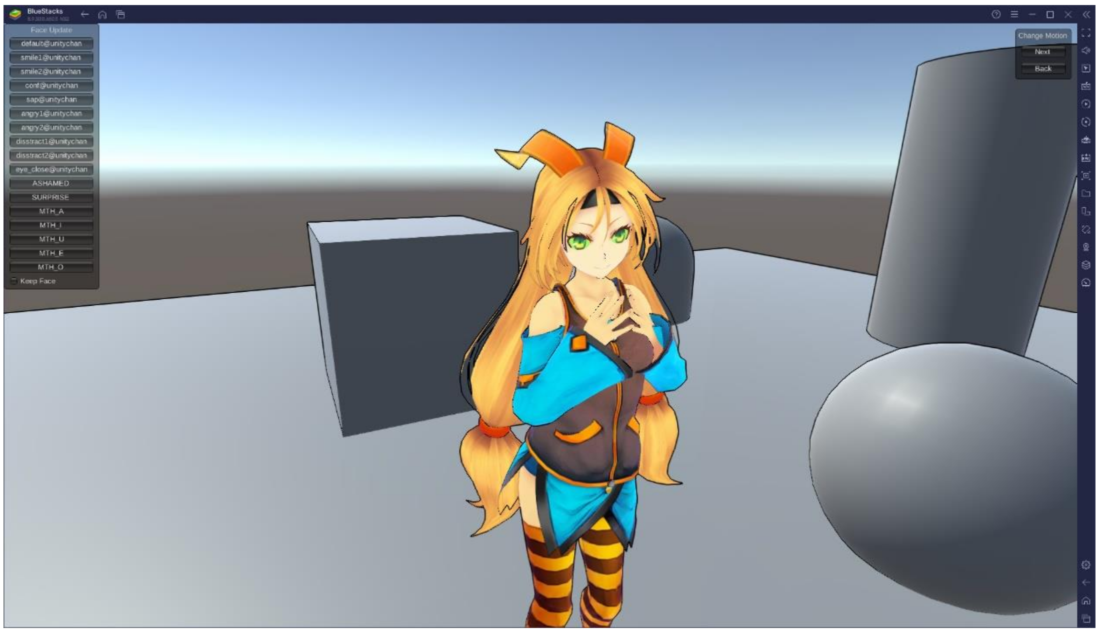
I manually added an extra pass to it to make it work correctly.



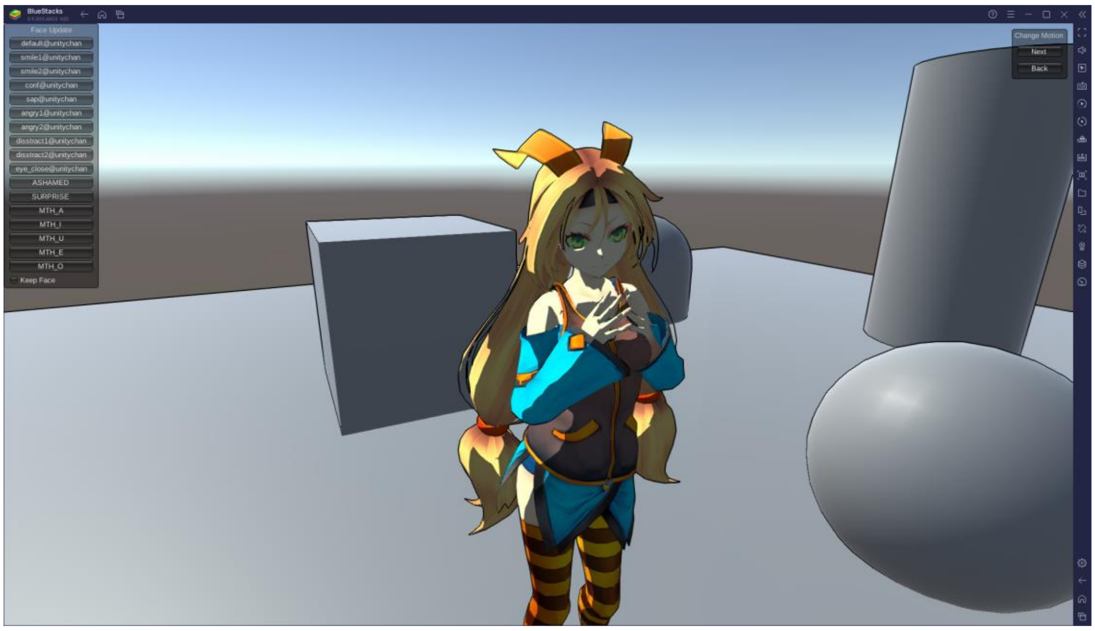
```
}
Pass
{
    Name "DepthNormals"
    Tags{"LightMode" = "DepthNormals"}

    ZWrite On
    Cull[_Cull]

    HLSLPROGRAM
    #pragma exclude_renderers gles gles3 glcore
    #pragma target 4.5

    #pragma vertex DepthNormalsVertex
    #pragma fragment DepthNormalsFragment

    // -------------------------------------
    // Material Keywords
    #pragma shader_feature_local _NORMALMAP
    #pragma shader_feature_local_fragment _ALPHATEST_ON
    #pragma shader_feature_local_fragment _GLOSSINESS_FROM_BASE_ALPHA

    //--------------------------------------
    // GPU Instancing
    #pragma multi_compile_instancing
    #pragma multi_compile _ DOTS_INSTANCING_ON

    #include "UniversalToonInput.hlsl"
    #include "Packages/com.unity.render-pipelines.universal/Shaders/DepthNormalsPass.hlsl"
    ENDHLSL
}
```



So if you are making a custom coded model shader, make sure it has a DepthNormals pass otherwise it won’t work correctly with post outline, and make sure the DepthNromals supports the platform you need.



UnityChan using UnityChan's UniversalToon.shader tested on Android



UnityChan Realtoon shader(Realtoon's outline is off) on Android

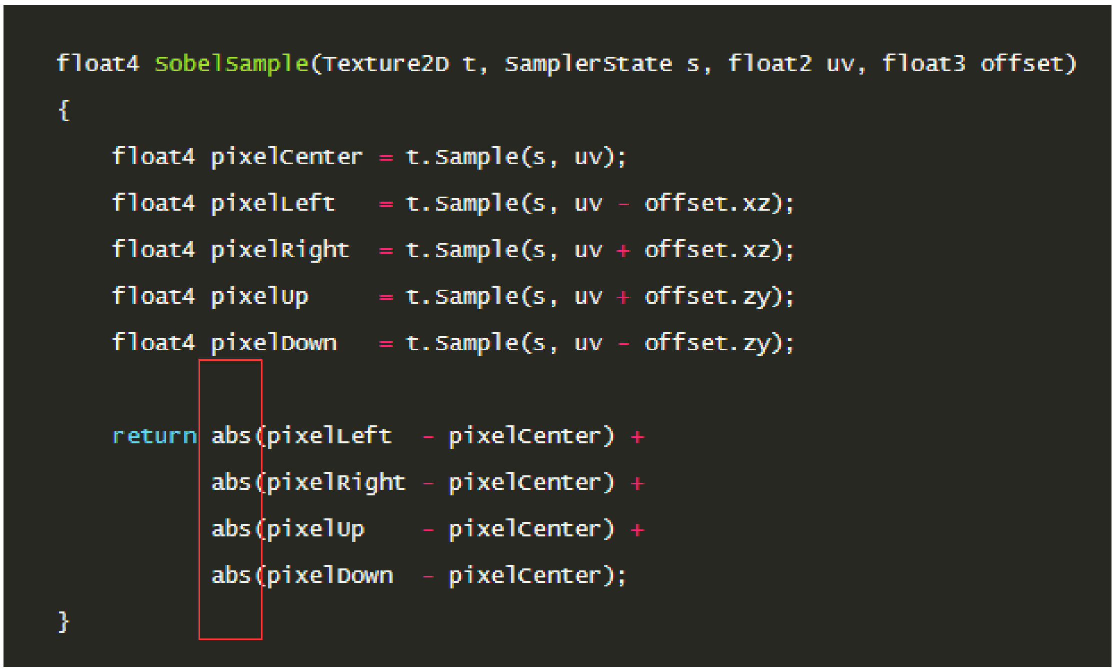
Credits:

Example contains Unity Chan models which is from Unity Japan.

Uniy Chan's toon shader came from this github link.
https://github.com/EddiePrince/UTS3ToonShaderURP

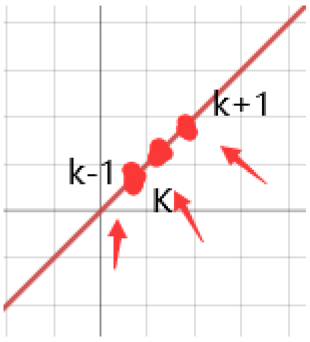
https://www.vertexfragment.com/ramblings/unity-postprocessing-sobel-outline/
https://github.com/Daodan317081/reshade-shaders
The shader got some ideas from Steven Sell and Alexander Federwisch with links above and thanks to them..

Please not that if you want to take a look at Steven Sell's sobel outline guide. He uses absolute value of difference to do the sample.



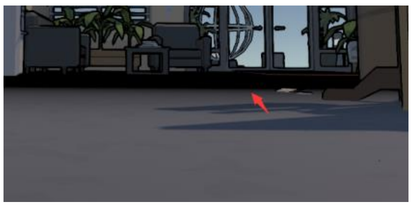
```
float4 SobelSample(Texture2D t, SamplerState s, float2 uv, float3 offset)
{
    float4 pixelCenter = t.Sample(s, uv);
    float4 pixelLeft   = t.Sample(s, uv - offset.xz);
    float4 pixelRight  = t.Sample(s, uv + offset.xz);
    float4 pixelUp     = t.Sample(s, uv + offset.zy);
    float4 pixelDown   = t.Sample(s, uv - offset.zy);

    return abs(pixelLeft  - pixelCenter) +
           abs(pixelRight - pixelCenter) +
           abs(pixelUp    - pixelCenter) +
           abs(pixelDown  - pixelCenter);
}
```

Which might cause problems as abs means that the a curve data of the pixels will be giving result out of nowhere from the middle.

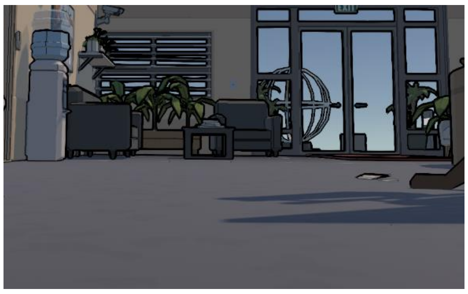


For example in a curve ,you want to get the sampled delta value of each point.
D(k-) = f(k) – f(k-1), is the sampled value at point k to the left, let's say it's 0.1 in the graph.
D(k+) = f(k) – f(k + 1), is the sampled value at point k to the right, then it's -0.1 because the curve is linear. So D(k-) and D(k+) will cancel each out.
Then D(k) = D(k-) + D(k+) = 0.1 + (-0.1) = 0
Which is like a linear surface of depth should have no outline effect, only non-linear sudden changes of depth will cause outline.
But if you add "abs" to D(k-) and D(k+), the delta will be
D(k) = abs(D(k-)) + abs(D(k+)) = abs(0.1) + abs(-0.1) = 0.2

There's no point to add outline effect if the depth just changes linearly.



And the depth outline will be dirty when you look from a close angle by the surface in Steven's guide like in the picture above. Which I fixed in Clean Outline shader.



There will be no dirty effect for depth samples.

If you have any problems, you could send me email
ryanflees@hotmail.com
ryanbai2008@163.com