Retro Scanline Post-Processing

Welcome to the documentation for Retro Scanline Post-Processing, a customizable post-processing plugin that brings the nostalgic feel of CRT displays, retro arcades, and analog televisions directly into your Unity projects.

Getting Started:

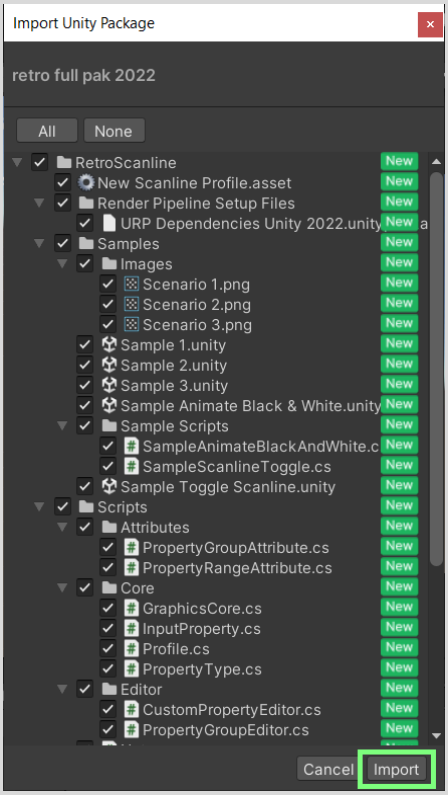
To set up Retro Scanline Post-Processing, We need to follow this simple four steps listed below:

- **Import Retro Scanline Post-Processing in your project.**
- **Enable Retro Scanline depending on your Render Pipeline.**
- **Create a Retro Scanline Profile.**
- **Assign the Scanline Profile to the required field.**

Now let's begin to set up everything step by step.

Step 1: Import Retro Scanline Post-Processing in your project

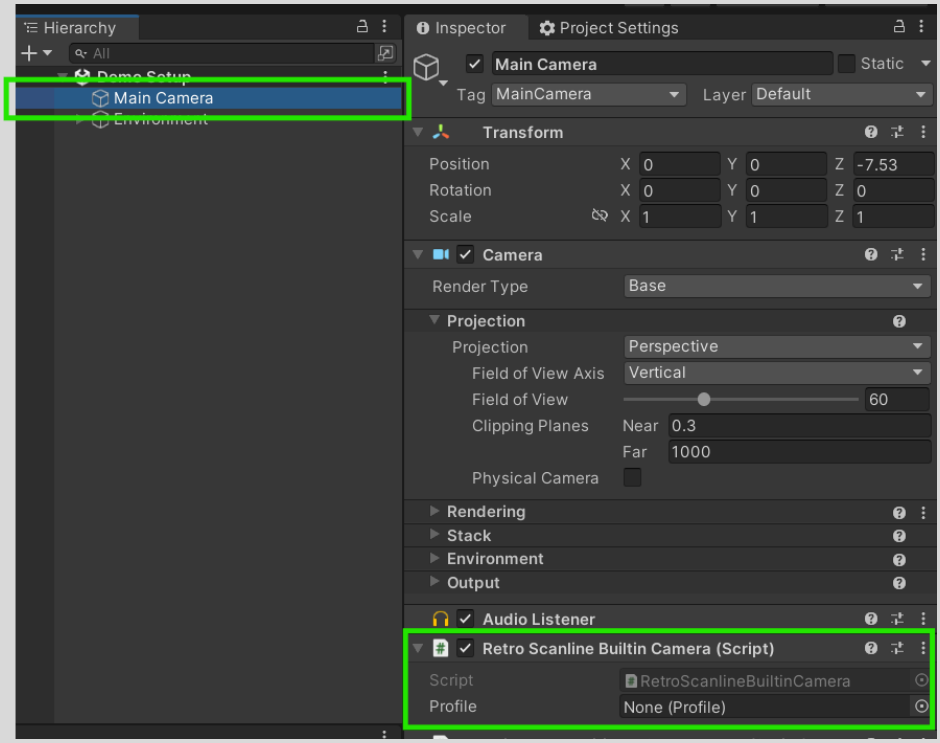
Import Retro Scanline Post-Processing package from Unity Package Manager. Make sure to import all files that come from the package.



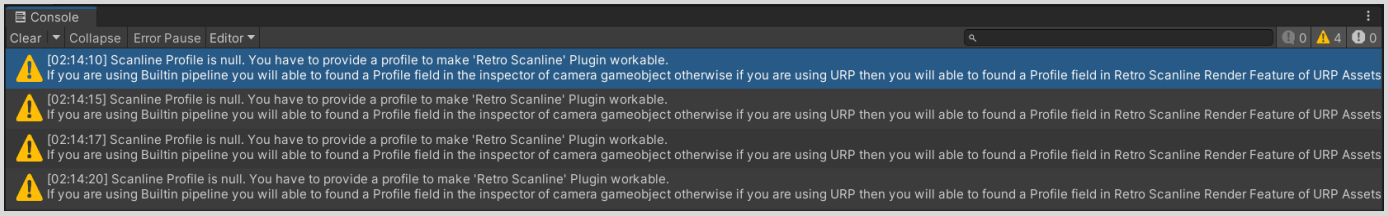
Step 2: Enable Retro Scanline depending on your Render Pipeline

Built-in Render Pipeline Setup: Select the Main Camera from the scene and add “RetroScanlineBuiltinCamera” Component. Now you have to create a Profile asset to make it workable, please follow step 3 to do that.

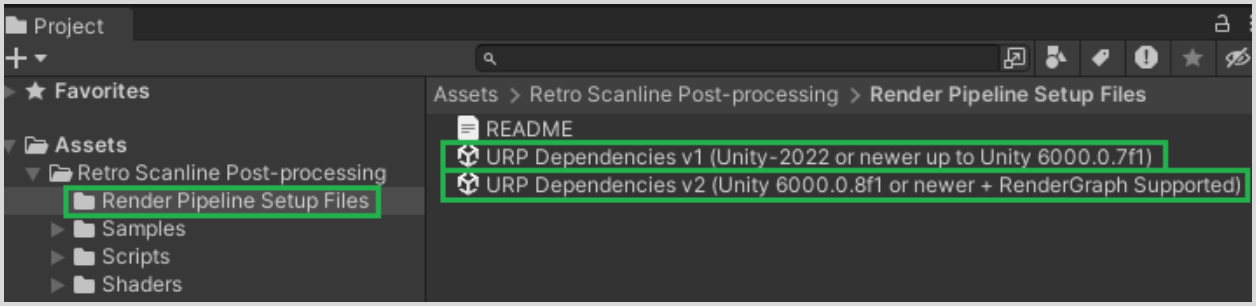
Note: The “RetroScanlineBuiltinCamera” component works only with the Built-in render pipeline.



After adding the “Retro Scanline Builtin Camera” component, you might notice a warning in the console like in the image below. This is expected and nothing to worry about it. We will resolve the warning by assigning Retro Scanline profile to the required field in Steps 3 and 4.

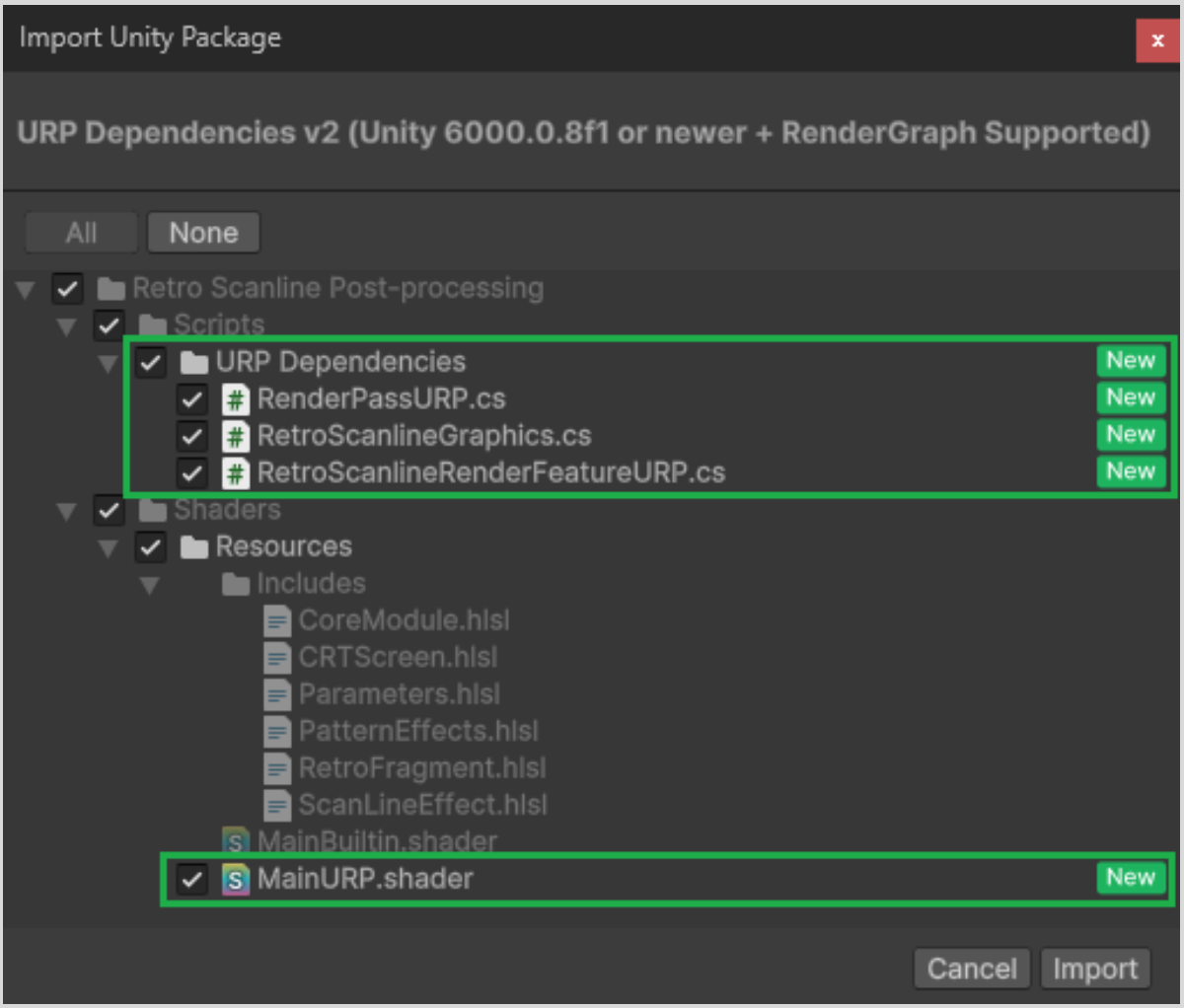


Universal Render Pipeline Setup: If you are using Universal Render Pipeline Please make sure the Unity Universal Render Pipeline package is already imported in your project.
Please Import a URP Dependencies setup file from the following directory below.
Retro Scanline Post-Processing>Render Pipeline Setup Files>

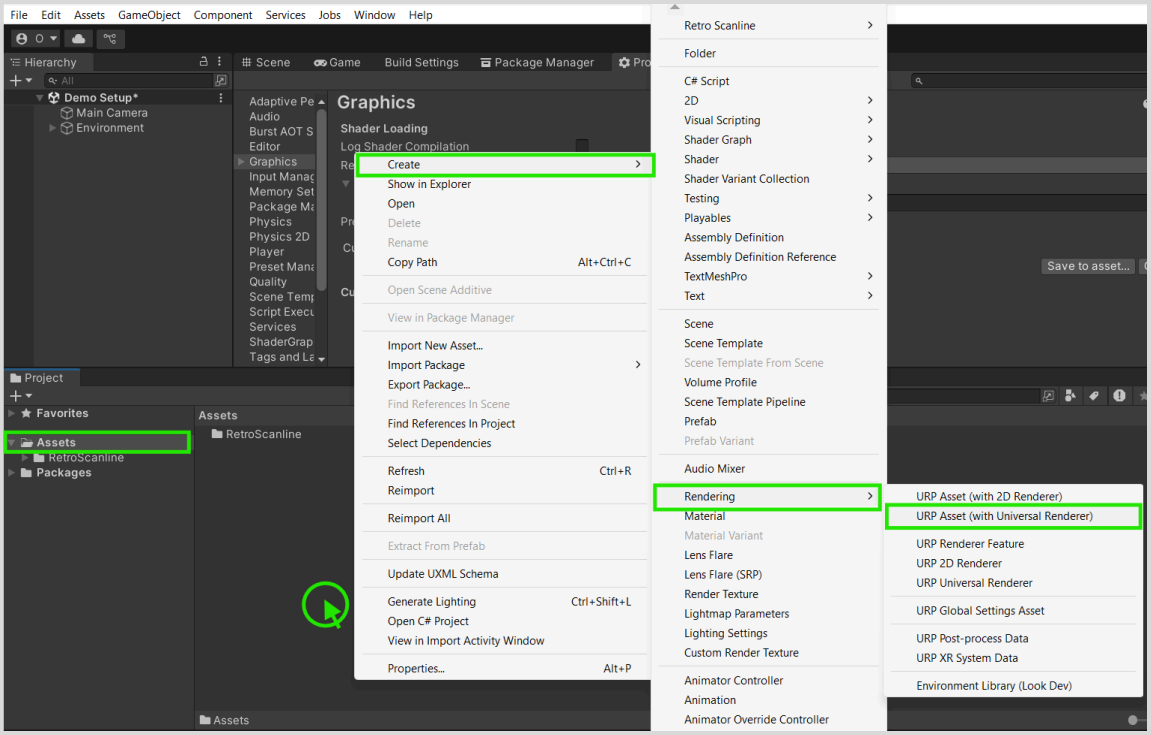


Please note there are multiple setup files available there, you have to import only one file from there. Please make sure to import the right version of the setup file based on your Unity version and URP version. For more clarification there is a README file available, Please have a look.
To import Double-click on a setup file something like URP Dependencies v-.unitypackage, then click on the “Import” button.

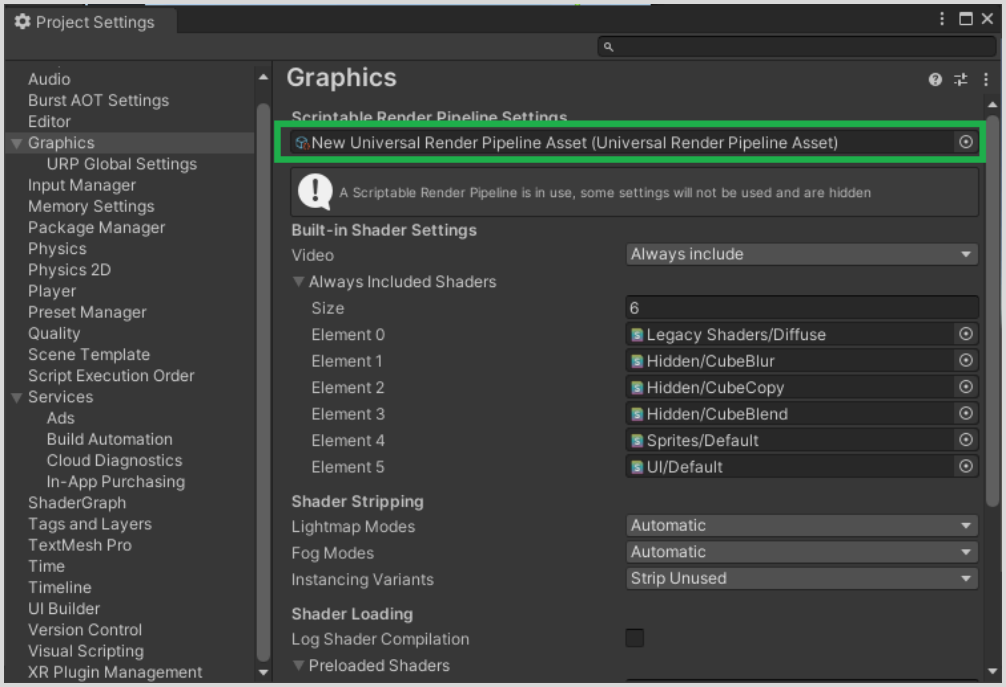
Note: Ensure all listed files are selected before pressing on the import button. Also, make sure the Unity Universal Render Pipeline (URP) package is already imported in your project otherwise, you may encounter few errors.



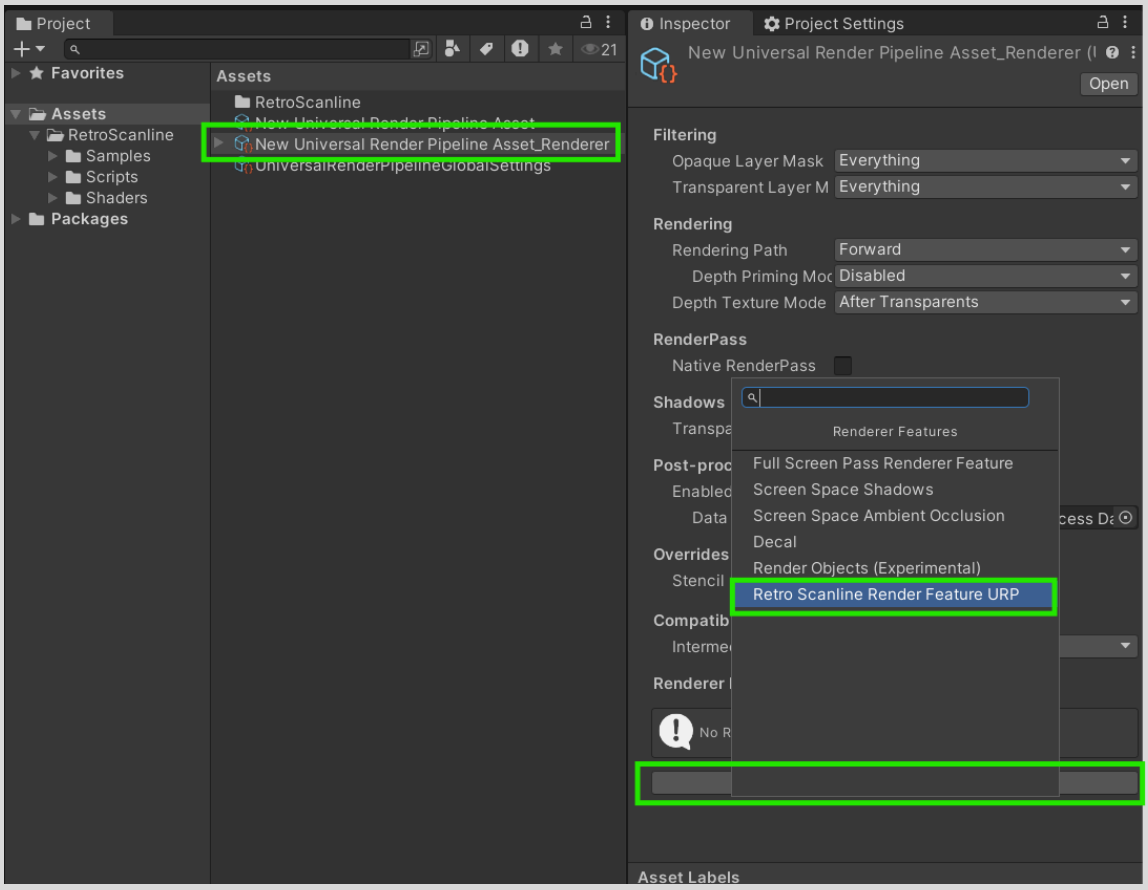
If you have not create any URP Assets, Please right click on empty place of “Assets” folder and navigate to Create>Rendering>URP Asset (with Universal Rendering) to create new URP assets



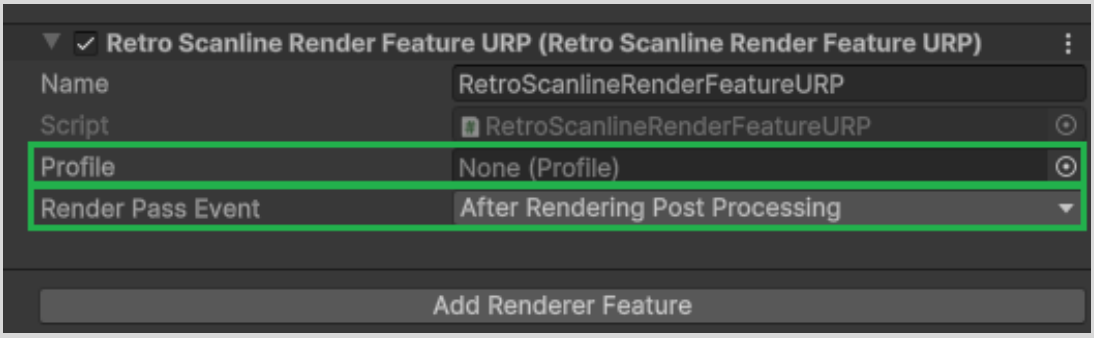
Open Project Settings from Edit>Project Settings then select “Graphics” and assign Render Pipeline Asset into the “Scriptable Render Pipeline Settings” field.



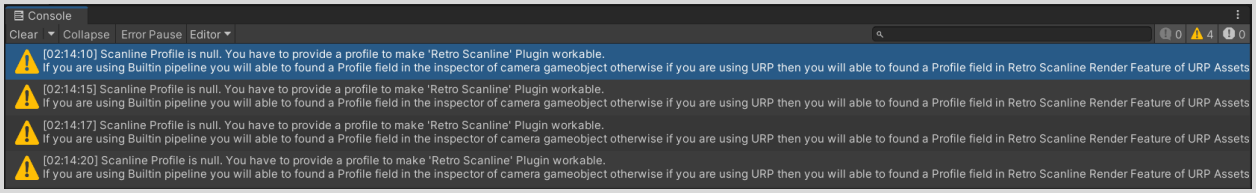
Please select the Universal Render Pipeline Asset_Renderer scriptable object from the asset folder then click on the “Add Render Feature” button from the inspector and choose “Retro Scanline Render Feature URP”.



Now we are almost done. After adding the render feature you will be able to see two important fields in the inspector Profile and Render Pass Event. The Render Pass Event field allows you to change the injection point of the render pass event. You can also see a Profile field which is currently empty, now we need to create a profile asset and assign it to the profile field, to do that please follow step 3 and step 4.

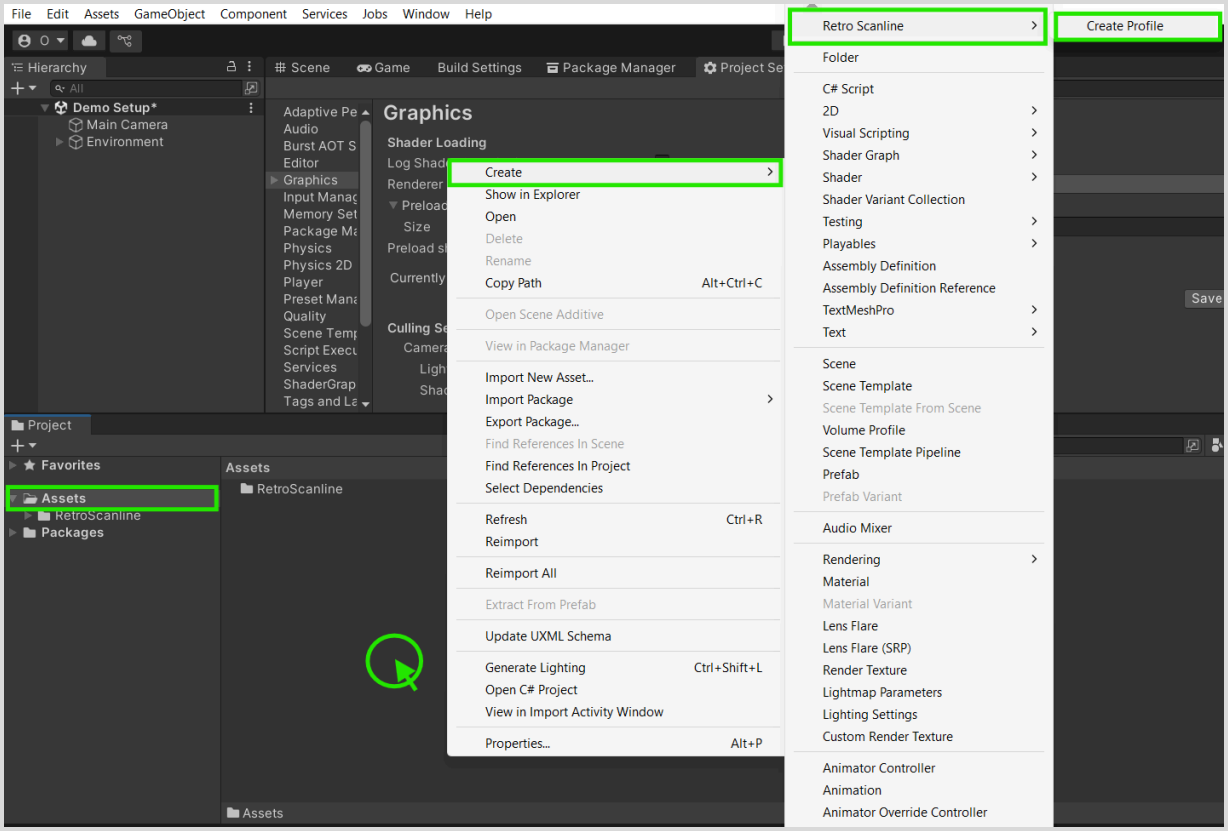


Note: After adding the "Retro Scanline Render Feature URP", you might notice a warning in the console like image below. This is expected and nothing to worry about it will be resolved in Steps 3 and 4, once the Retro Scanline profile is properly assigned to the required field.



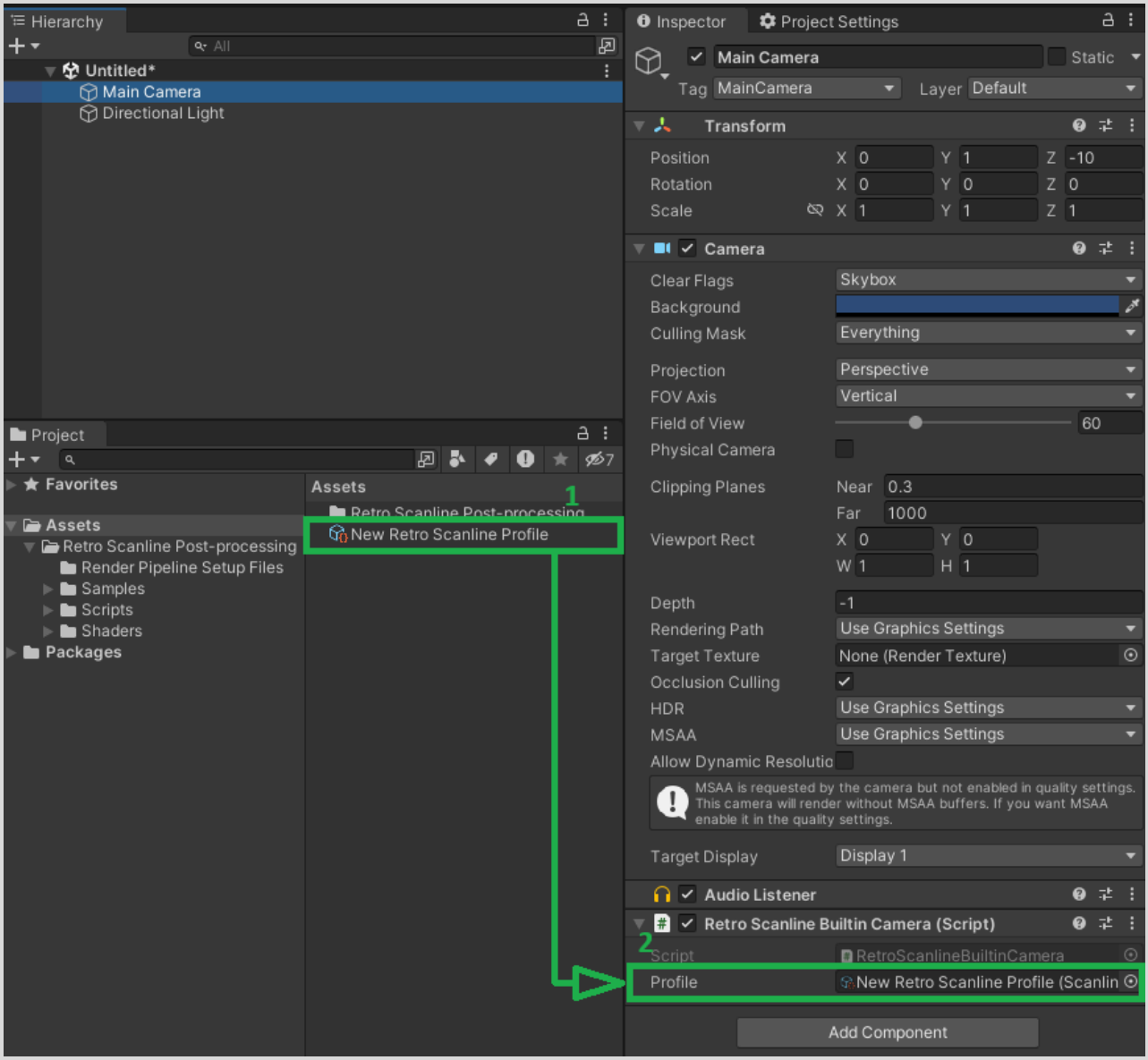
Step 3: Create a Retro Scanline Profile

Please right click on the empty place of the Asset folder and navigate "Create>Retro Scanline>Create Profile" to create a new retro scanline profile asset.

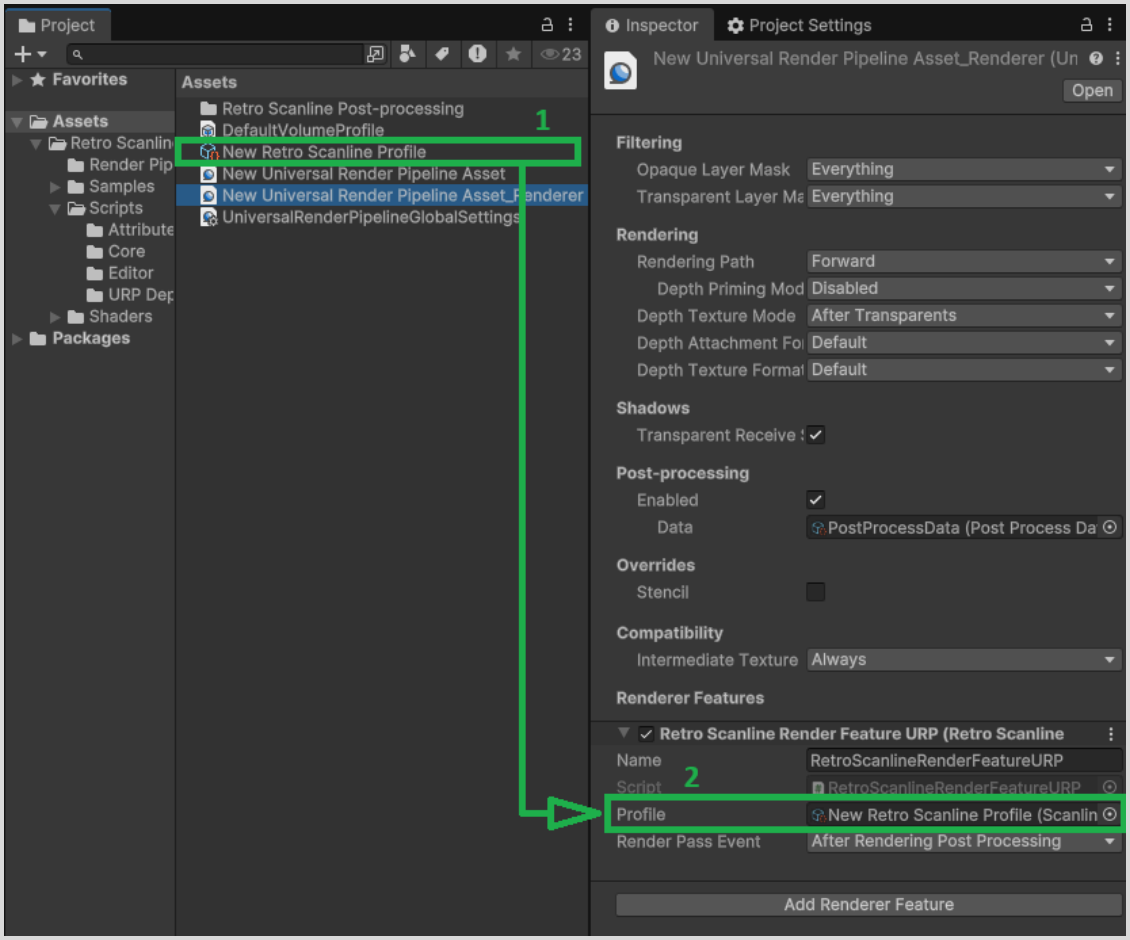


Step 4: Assign the Scanline Profile to the required field

If you are using the Builtin Render Pipeline please select the Main Camera and look for Profile field of RetroScanlineBuiltinCamera component in the inspector, then assign the newly created Retro Scanline Profile asset in the profile field.



If you are using Universal Render Pipeline URP please select Universal Render Pipeline Asset_Renderer file from project window and look for the Profile field of Retro Scanline Render Feature URP in the inspector, then assign the newly created Retro Scanline Profile asset in the profile field.



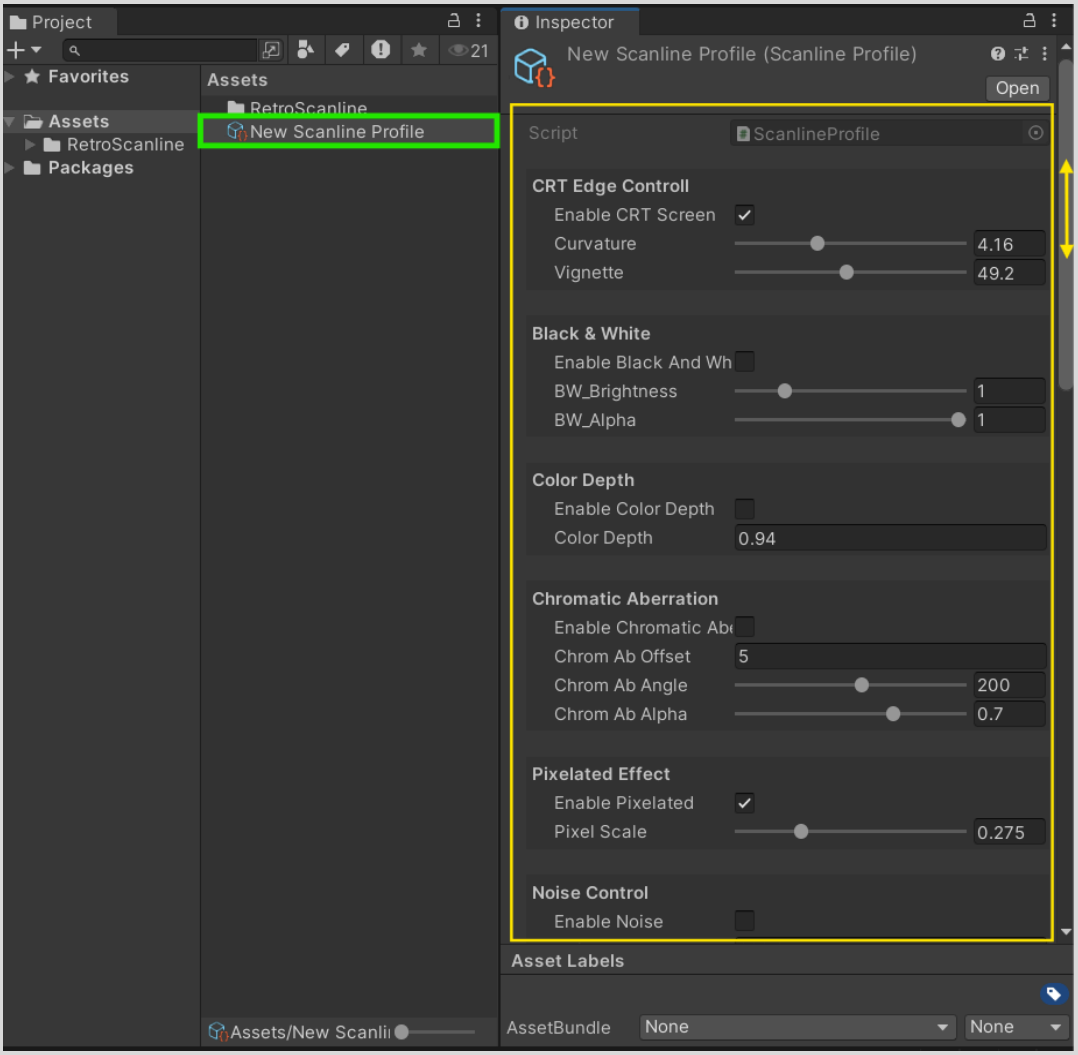
If you follow the above instructions successfully, I hope at this point you are able to see Retro Scanline Post-Processing effect is enabled on your game view window.

Manual

Retro Scanline Profile:

Retro Scanline Profile allows you full control of this plugin, you can enable or disable specific filter effects from there. You can also adjust & tweak Retro Scanline Profile parameters from the Inspector and the scripting API both.

Let's try to learn how we can adjust & tweak parameters from the Inspector.
Select your Retro Scanline Profile from your Assets browser window and have a look to the Inspector, You will be able to see all of the available parameters. Now you can change values of available parameters as you need.



API Documentation

ScanlineProfile

ScanlineProfile actually contains all of the available effects parameters to adjust and tweak the post-processing effect. It's inherited from Profile, Profile is a base class which is actually handled internally to make this plugin workable.

Public Property:

Name	Type	Remarks
enableCRTScreen	BoolProperty	Enable or disable CRT screen effect.
curvature	FloatProperty	Control the CRT screen curvature amount.
curveSmoothness	FloatProperty	Allow to smother the CRT screen curvature from center of edge to each corner.
vignette	FloatProperty	Vignette amount of CRT screen
enableBlackAndWhiteColor	FloatProperty	Enable or disable Black & White Color effect. Alpha channel will allow you to control transparency.
bW_Brightness	FloatProperty	Brightness strengths the amount of black & white effect.
bW_Alpha	FloatProperty	Alpha of black and white effect to blend color with original.
enableColorDepth	BoolProperty	Enable or disable Color Depth effect.
colorDepth	FloatProperty	Amount of depth of color.
enableChromaticAberration	BoolProperty	Enable or disable chromatic aberration effect.
chromAbOffset	FloatProperty	Offset of chromatic aberration effect.
chromAbAngle	FloatProperty	Angle to move the direction of chromatic aberration effect.
chromAbAlpha	FloatProperty	Alpha of chromatic aberration effect to blend color with original.
enablePixelated	BoolProperty	Enable or disable Pixelated effect.
pixelSize	FloatProperty	Size of pixels combined together.
enableNoise	BoolProperty	Enable or disable Noise effect.
noiseScale	FloatProperty	Size of each noise pixel.
noiseSpeed	FloatProperty	Update noise pixel flickering.

noiseIntensity	FloatProperty	Alpha of noise effect to blend color with original.
enableScanline	BoolProperty	Enable or disable Scanline effect.
lineColore	ColorProperty	Scanline effect color. Alpha channel will allow you to control transparency.
numberOfLine	FloatProperty	The amount of line will be visible on the screen for Scanline effect.
lineSharpness	FloatProperty	Control the thickness and soft edge of each line.
lineSpeed	FloatProperty	Line scrolling speed, negative value will change the scroll direction.
angleOfLine	FloatProperty	Allows you to change the line alignment to different directions.
enablejitterEffect	BoolProperty	Enable or disable the Jitter effect.
jitterStrengthWidth	FloatProperty	Strength of jittering effect to the horizontal direction.
jitterStrengthHeight	FloatProperty	Strength of jittering effect to the vertical direction.
jitterFrequency	FloatProperty	Amount the number of jittering waves.
jitterSpeed	FloatProperty	Speed of jittering waves.
enableAdvanceJitterEffect	BoolProperty	Enable or disable Advance Jitter effect.
advanceJitterFrequency	FloatProperty	Amount the number of jittering waves.
advanceJitterStrengthWidth	FloatProperty	Strength of advance jittering signal effect in horizontal direction.
advanceJitterStrengthHeight	FloatProperty	Height of advance jittering signal effect in vertical direction.
advanceJitterSpeed	FloatProperty	Speed of advance jittering wave.
advanceJitterNoiseScale	FloatProperty	Size of noise pixel inside the advance jittering wave only.

advanceJitterNoiseAlpha	FloatProperty	Alpha amount of noise pixel inside the advance jittering wave only.
enableSubpixelRGBMask	BoolProperty	Enable or disable the Sub Pixel RGB Mask effect.
subpixelScale	FloatProperty	Scale of sub pixel mask to allow resize it.
subpixelAlpha	FloatProperty	Alpha of Sub Pixel RGB Mask effect to blend color with original.
enableHalftoneDithering	BoolProperty	Enable or disable Halftone Dithering effect.
halftoneColor	ColorProperty	Color of halftone dithering filter. Alpha channel will allow you to control transparency.
halftoneScale	FloatProperty	Control the pattern of halftone dithering.
enableColorBasedThreshold	BoolProperty	The effect will apply a different threshold based on the current color of the pixel.
halftoneThreshold	FloatProperty	Control the threshold of effect between whiter and darker color area based on current pixel color.
enableAsterisk	BoolProperty	Enable or disable the Asterisk effect.
asteriskColor	ColorProperty	Color of asterisk symbol on the screen. Alpha channel will allow you to control transparency.
asteriskFrequency	FloatProperty	The density of the asterisk on the screen.
asteriskLineThickness	FloatProperty	Thickness with smooth line edge of asterisk symbol on the screen.
asteriskAngleInDeg	FloatProperty	Alignment of the asterisk pattern orientation.
enableCheckerboard	BoolProperty	Enable or Checkerboard effect.
checkerColor	ColorProperty	Color of the checkerboard effect. Alpha channel will allow you to control transparency.
checkerFrequency	FloatProperty	Allows to resize the checkerboard density.

checkerAngleInDeg	FloatProperty	Alignment of the checkerboard pattern orientation.
enableGridLine	BoolProperty	Enable or disable Grid Line effect.
gridLineColor	ColorProperty	Color of grid line pattern. Alpha channel will allow you to control transparency.
gridLineFrequency	FloatProperty	Density of grid line pattern.
gridLineThickness	FloatProperty	Line thickness with soft edge of each line.
gridLineAngleInDeg	FloatProperty	Alignment of the grid line pattern orientation.

InputProperty

InputProperty is an abstract class which is used as base class to store input property data. This property is used for encapsulating input data to pass into the material shader property with a unique ID.

Constructor:

InputProperty(string ID, PropertyType type)

Public Property:

Name	Type	Remarks
GetID {get;}	string	Contain unique id for input property. It indicates the material shader property name which allows to put value to the correct material shader property value.
GetPropertyType {get;}	PropertyType	Return a property type which type of data containing this InputProperty's instance.
GetValue {get;} : virtual	Object	Return the exact value which data contains this property, it will be defined by the child class who will inherit this class.

StringProperty

StringProperty is a class which is used to store string data with a unique ID and data type. It's inherited from InputProperty.

Constructor:

StringProperty(string defaultValue, string ID)

Public Property:

Name	Type	Remarks
value	string	Contain string value of actual data responsible for this property type.
GetValue {get;} : *override*	string	Return the string value that represents this property value.

FloatProperty

FloatProperty is a class which is used to store float data with a unique ID and data type. It's inherited from InputProperty.

Constructor:

FloatProperty(float defaultValue, string ID)

Public Property:

Name	Type	Remarks
value	float	Contain float value of actual data responsible for this property type.
GetValue {get;} : *override*	float	Return the float value that represents this property value.

IntProperty

IntProperty is a class which is used to store int data with a unique ID and data type. It's inherited from InputProperty.

Constructor:

IntProperty(int defaultValue, string ID)

Public Property:

Name	Type	Remarks
value	int	Contain int value of actual data responsible for this property type.
GetValue {get;} : *override*	int	Return the int value that represents this property value.

BoolProperty

BoolProperty is a class which is used to store bool data with a unique ID and data type. It's inherited from InputProperty.

Constructor:

BoolProperty (int defaultValue, string ID)

Public Property:

Name	Type	Remarks
value	bool	Contain bool value of actual data responsible for this property type.
GetValue {get;} : *override*	bool	Return the bool value that represents this property value.

ColorProperty

ColorProperty is a class which is used to store Color data with a unique ID and data type. It's inherited from InputProperty.

Constructor:

ColorProperty(int defaultValue, string ID)

Public Property:

Name	Type	Remarks
value	Color	Contain color value of actual data responsible for this property type.
GetValue {get;} : *override*	Color	Return the color value that represents this property value.

Vector2Property

Vector2Property is a class which is used to store Vector2 data with a unique ID and data type. It's inherited from InputProperty.

Constructor:

Vector2Property(int defaultValue, string ID)

Public Property:

Name	Type	Remarks
value	Vector2	Contain vector2 value of actual data responsible for this property type.
GetValue {get;} : *override*	Vector2	Return the vector2 value that represents this property value.

Thanks For Reading