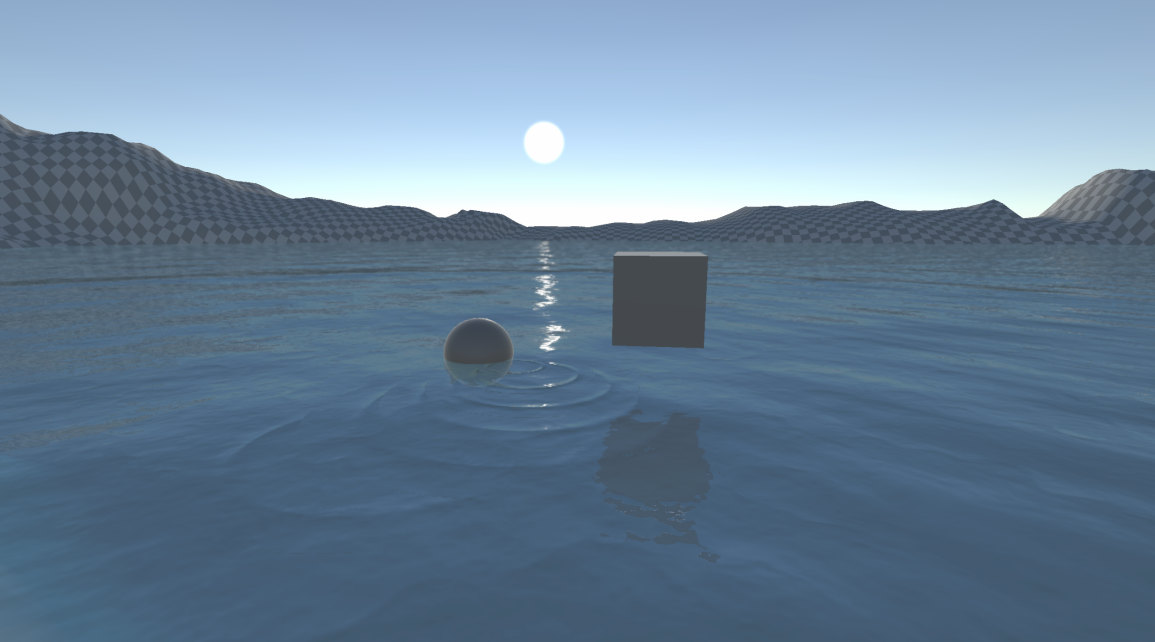
# Interactive Water System and Under Water Effect



Introduction

Dive into the depths of realism with our Interactive Water System and Underwater Effect! This asset package offers a comprehensive solution for creating stunning water surfaces and immersive underwater experiences in your Unity projects.

**Key Features:**

**PBR-Based Water Surface Rendering:** Achieve photorealistic water surfaces that respond dynamically to lighting and environmental conditions.

**Gerstner Waves:** Simulate realistic wave patterns with Gerstner wave technology for natural movement and interaction.

**SSR Reflections:** Utilize Screen Space Reflections to create stunning, mirror-like water effects that enhance visual fidelity.

**Diverse Water Surfaces:** Choose from multiple water surface styles to suit different environments, from tranquil lakes to raging oceans.

**Interactive Water Ripples:** Add interactivity with water ripple effects that respond to player interactions and objects within the water.

**Underwater Rendering:** Experience beautifully rendered underwater environments, complete with visual effects that mimic the distortion and color changes of being submerged.

**Caustics:** Enhance the realism with accurate underwater caustics that simulate light patterns on the seabed.

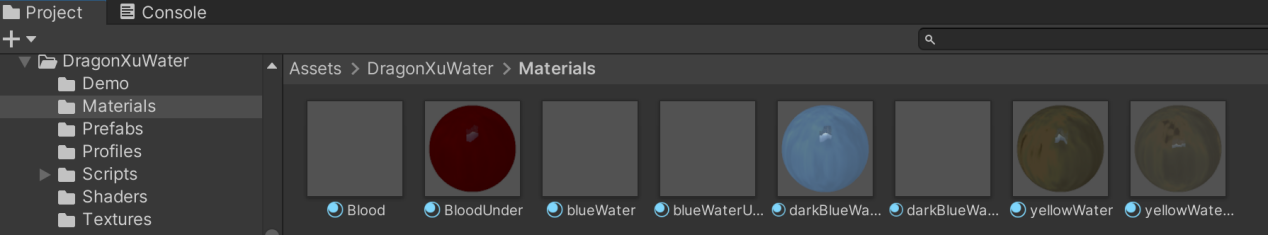
**Distance Blur:** Address moiré issues with distance blur, ensuring seamless visual transitions and enhanced depth perception.

This resource package is perfect for developers seeking to elevate their aquatic scenes, whether for games, simulations, or educational experiences. Unleash your creativity and bring your water environments to life!

How To Use

**Water Surface Material Parameters**

You can find the materials following the path DragonXuWater/Materials.



**Under Water Surface**

Is the current material for the for the underwater perspective of the water? We utilize two materials for rendering the water surface: one for the view from above the water, capturing the surface appearance, and another for the underwater perspective, highlighting the unique visual effects and characteristics of the water below.

The **Under Water Surface** parameter for the material rendering the view from above water is marked as TRUE.



(view of water surface from above the water)

The **Under Water Surface** parameter for the material rendering the underwater perspective is marked as FALSE.



(view of water surface from the underwater perspective)

**Shallow Color**

Color of the water surface in the shallow area.

**Deep Color**

Color of the water surface in the deep area.

**Surface Roughness**

Controlling the roughness of the water surface. The surface seems rougher when the parameter is set larger.

**Depth Factor**

If the **Under Water Surface** parameter is set false, **Depth Factor** controls how deep you can see into the water and the depth of the shallow area of the water.

If the **Under Water Surface** parameter is set true, **Depth Factor** controls transparent the water surface from the under water perspective is. It is set between 0 to 1. the surface is transparent when Depth Factor is set to 0, and is opaque when set to 1.

**Water Normal Tex**

The normal texture controlling the normal map of the surface. Specifies the normal map texture for simulating surface details.It is given in the folder DragonXuWater/Textures.

**Normal Intensity**

Adjusts the intensity of the normal map effects.

**Normal Tiling**

Sets the tiling scale for the normal map.

**Max Wave Length**

The wave length of the largest gerstner wave.

**Wave Count**

Specifies the number of waves to simulate.

**Wave Amplitude**

Controls the height of the waves.

**Wind Direction**

Sets the direction of wind affecting the wave motion.

**Direction Randomization**

Adjusts how much the gerstner wave direction can vary randomly.

**Floating Speed**

Controls the speed at which the waves float and move.

**Reflection Strength**

Sets the intensity of Screen Space Reflection (SSR) on the water surface.

**Ripple Height Map**

Texture used for generating ripple effects on the water surface. This is also given in the folder DragonXuWater/Textures.

**Ripple Strength**

Determines the intensity of the ripple effects.

**SkyBox**

Specifies the skybox texture for environmental reflections.

**Env Rotation**

Controls the rotation of the environment reflections.

**Noise Texture**

Texture used for adding noise to the sampling of normal map for a more dynamic appearance. This is also given in the folder DragonXuWater/Textures.

**Noise Scale**

Adjusts the scale of the noise texture.

**Noise Strength**

Controls the intensity of the noise effect.

**DIFFUSE OFF**

Toggle to disable diffuse lighting.

**SPECULAR OFF**

Toggle to disable specular highlights.

**SH OFF**

Toggle to disable shadowing.

**IBL OFF**

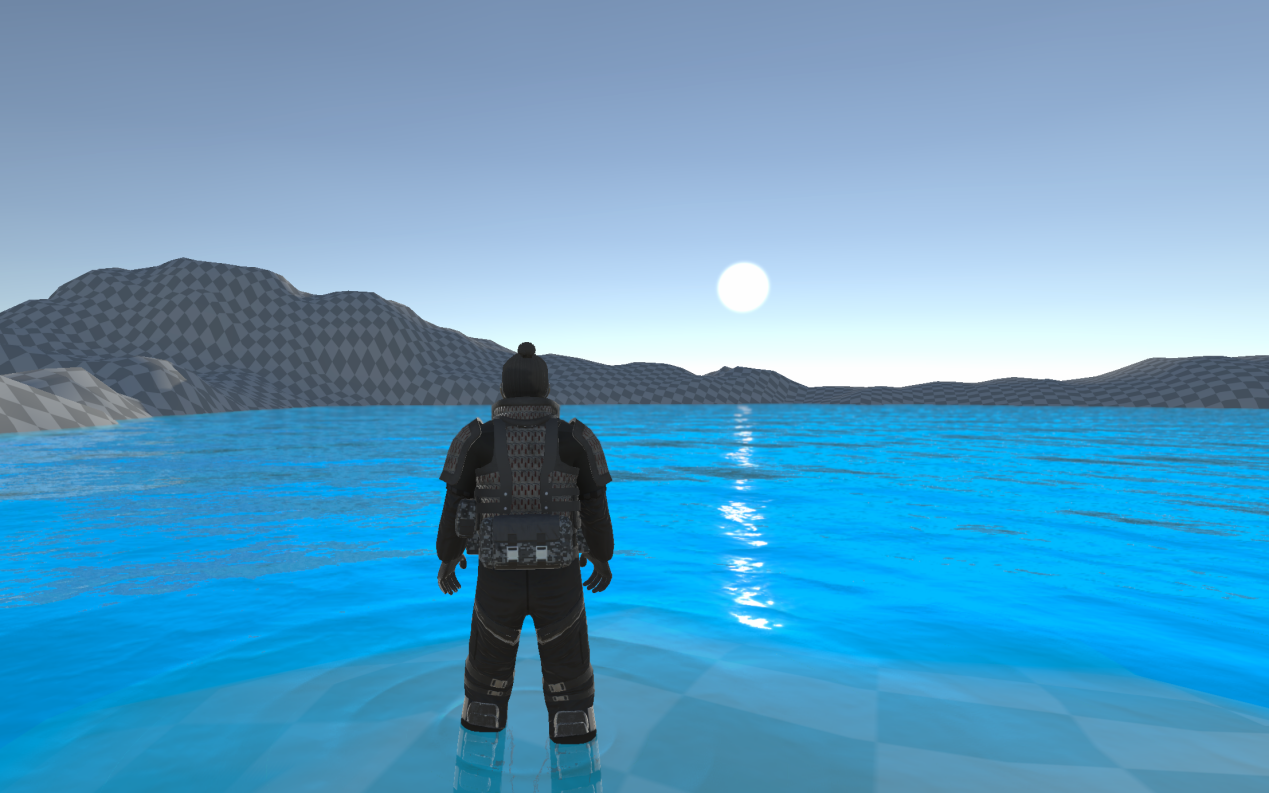
Toggle to disable Image-Based Lighting.

**SSR OFF**

Toggle to disable Screen Space Reflections.

**VERTEX ANIM OFF**

Toggle to disable vertex animation.

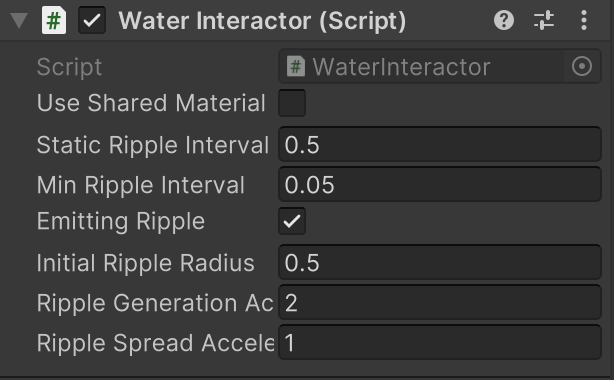


**Ripple Interactions**

To enable the ripple interaction, there **MUST** be a **Water Manager** in your scene. The prefab of it can be found in DragonXuWater/Prefabs. Simply drag and place it into your scene.

The object you want to interact with the water must have a **Collider** component and a **Water Interactor** component. The **Water Interactor** script can be found in the file DragonXuWater/Scripts.

The Water Mesh must have a **Water** component. This can be found in the file DragonXuWater/Scripts.

**Parameters of Water Interactor script**

**Use Shared Material**

Turned off by default. If you have only one water instance in your scene, or you simply want all the waters share the same ripple positions, then you can turn this on. If you have more than one water instances in the scene and don’t want them to share the ripple positions, turn it off.

**Static Ripple Interval**

The interval between the emission of the two ripples when the interactor object is not moving. The interval will be shorten once the interactor object moves, and return to this value when the interactor object stops moving.

**Min Water Interval**

The minimum interval between the emission of the two ripples when the interactor object is moving.

**Emitting Ripples**

A toggle controlling whether this object is emitting ripples.

**Initial ripple radius**

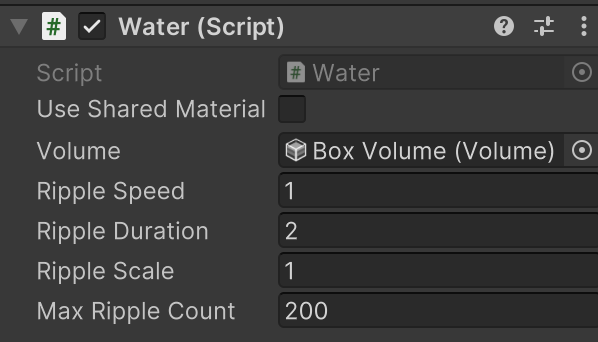
The radius of the ripple when initially generated.

**Ripple Generation Acceleration**

Specifies the acceleration of the generating speed when the interactor object is moving.

**Ripple Spread Acceleration**

Specifies the acceleration of the spreading speed when the interactor object is moving.



**Parameters of Water script**

**Use Shared material**

Turned off by default. If you have only one water instance in your scene, or you simply want all the waters share the same properties, then you can turn this on. If you have more than one water instances in the scene and don’t want them to share the properties, turn it off.

**Ripple Speed**

Specifies the spreading speed of the ripples.

**Ripple Duration**

Specifies the how long time one ripple stays on the water surface.

**Ripple Scale**

Specifies the scale of the ripple.

**Min Ripple Count**

Specifies the maximum count of the ripples on this water surface. Note that this CAN NOT be changed dynamically in runtime. This is because we pass the information of the ripples to the shader via arrays, and in Unity, the length of the array passed to the shaders can not be changed in run time. Say your Min Ripple Count is set to 10 by default, and you changed it to 100 after entering play mode. Then the shader will only receive the first 10 elements of the ripple array, the rest 90 elements will be ignored.

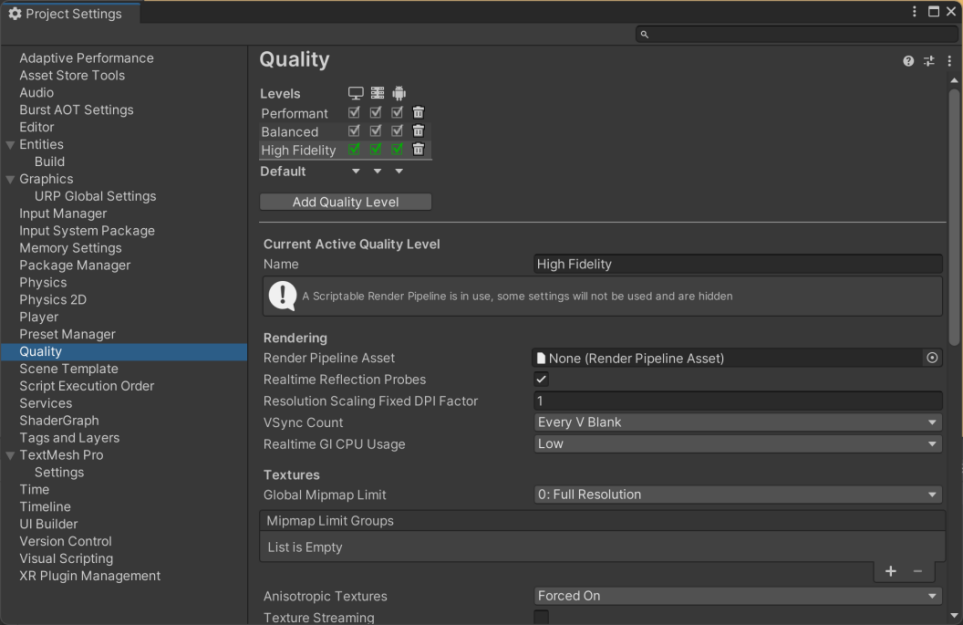
**Post Processing Effects**

We simulate the under water effect and under water caustics using post processing. To enable the post processing effects, you need to add the rendering feature into your render pipeline.

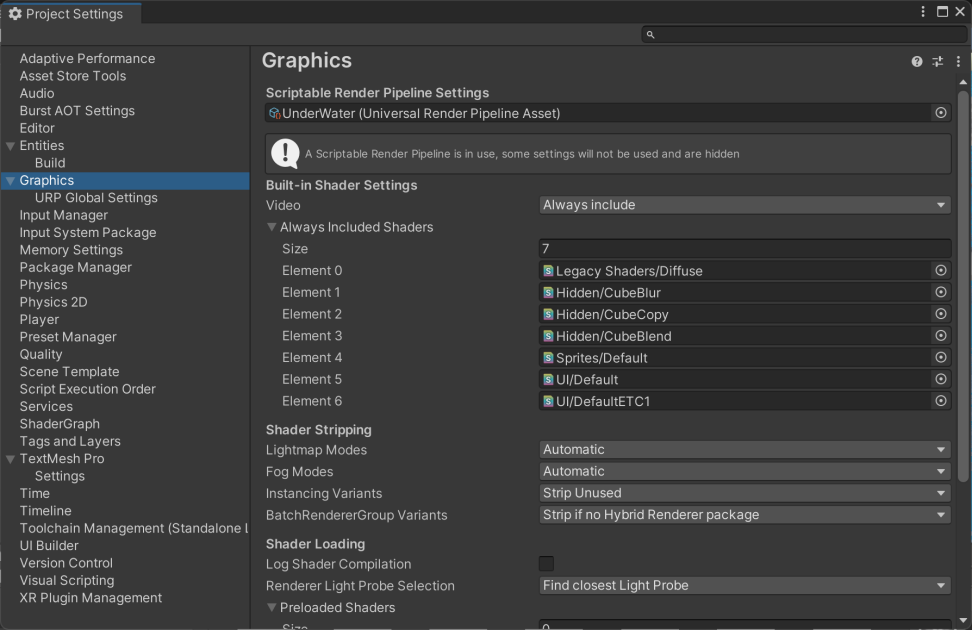
In this package, we created a new render pipeline asset and a renderer data asset for showcasing the under water post processing effects.



If you enter the play mode in our demo scenes, your render pipeline is automatically switched to our showcasing render pipeline asset (if your pipeline asset in the Quality settings is NOT set to none, this won’t happen).



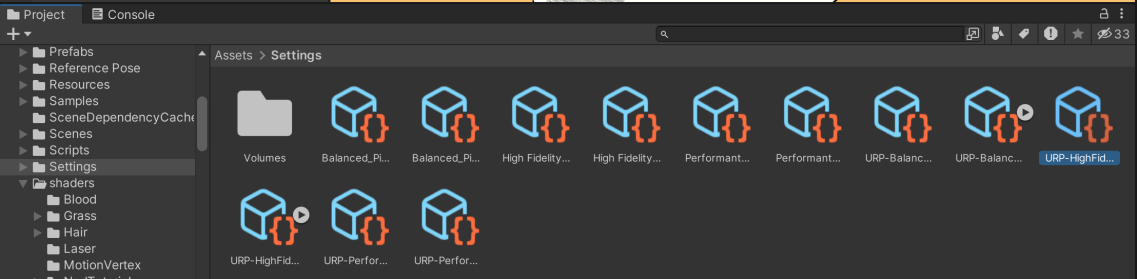
(As is shown in the above picture, the Render Pipeline Asset in the Quality settings is set to none)



(After you play the demo scenes, the pipeline will switch to the showcasing Outline render pipeline asset, as is shown in the above picture. This won’t happen if your pipeline asset in the Quality settings is NOT set to none)

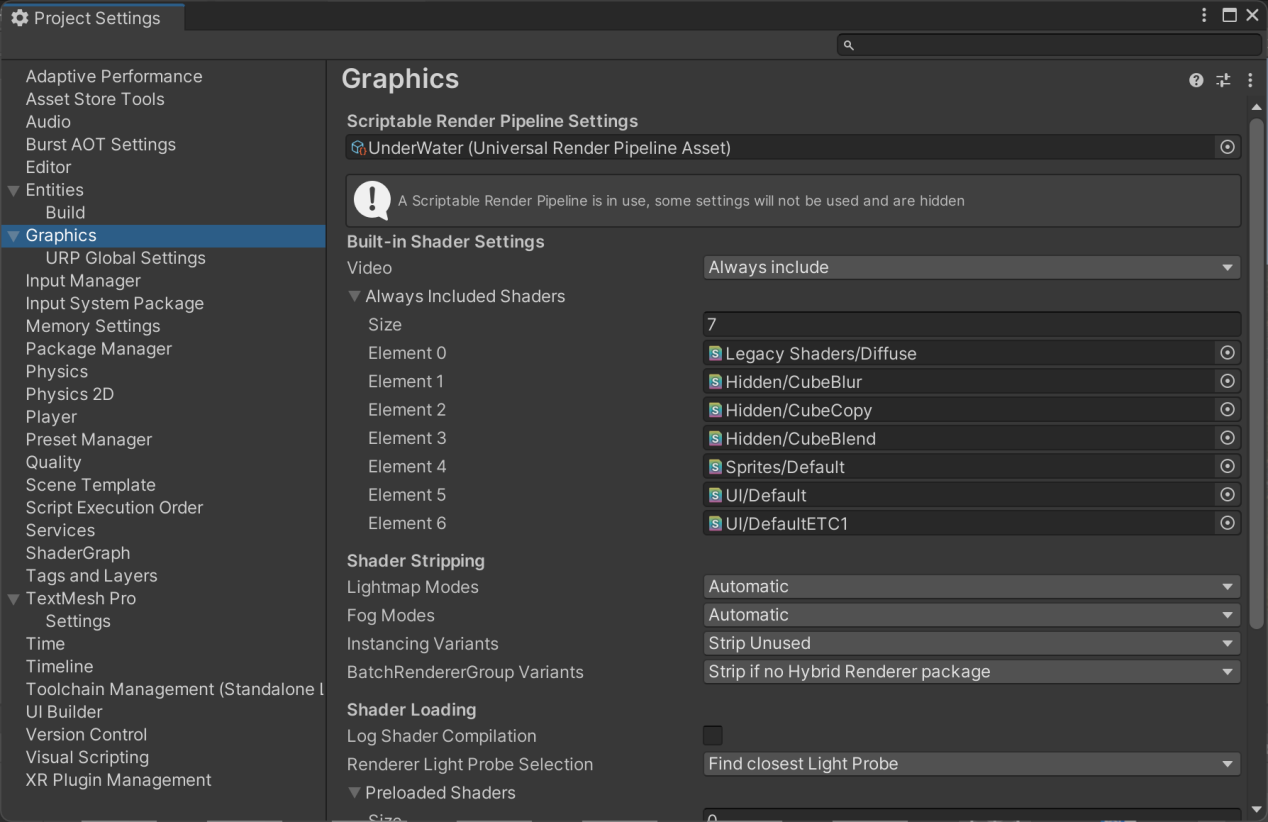
If you want to add the outline effect into your own render pipeline in your game, you need to add the render feature into your render data.

First, you should find your render data that your render pipeline is using. It’s often located in the Settings file of your project.

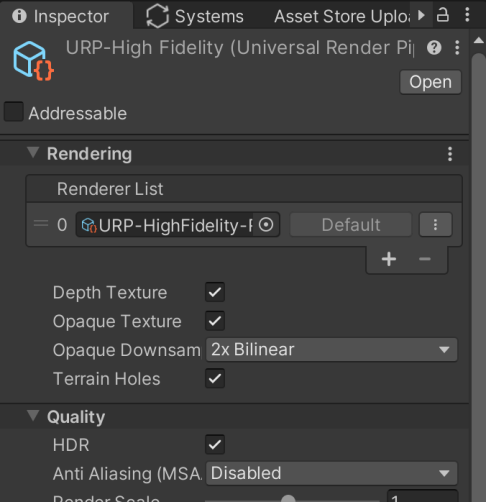


(In this picture, I present one of my own projects as an example. Your own pipeline assets can be found under the Settings file by default)

Or you can go to Graphic Setting in your Project Setting. Your pipeline asset is located in the Scriptable Render Pipeline Settings slot.

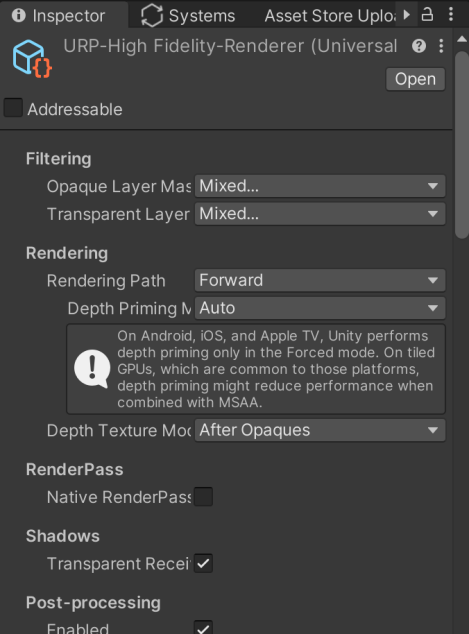


Find your render pipeline asset, then you can see your renderer data asset in the Renderer List.

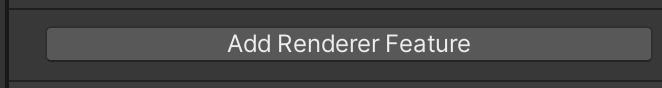


(In this picture, I present one of my own projects as an example. The Renderer data is listed in the Renderer List of your pipeline asset)

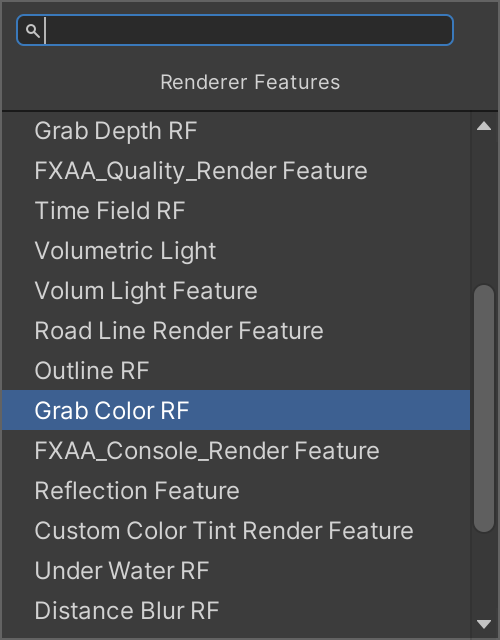
Then, select the renderer data



In the Inspector, click on the “Add Renderer Feature” Button at the bottom of the page

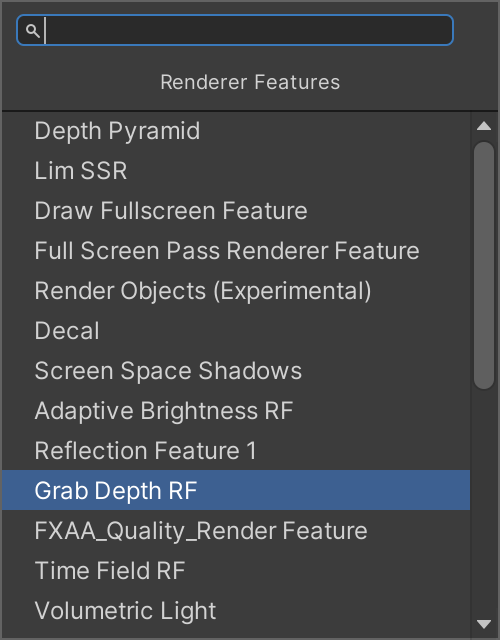


Click on ‘Grab Color RF’ to add it.



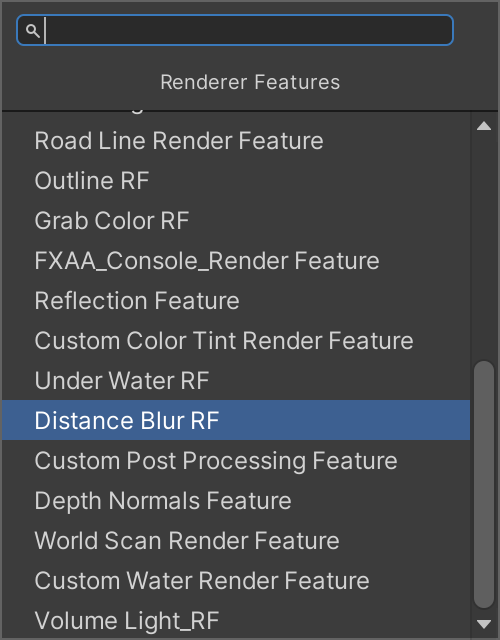
Click on the “Add Renderer Feature” Button again.

This time, select ‘Grab Depth RF’ to add it.



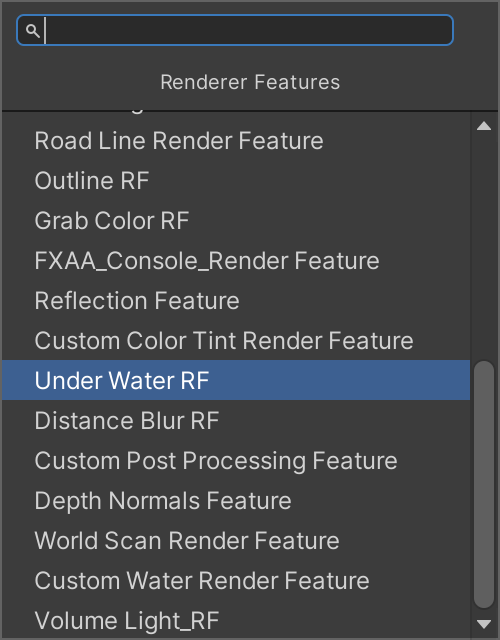
Click on the “Add Renderer Feature” Button again.

This time, select ‘Distance Blur RF’.



Click on the “Add Renderer Feature” Button again.

This time, select ‘Under Water RF’ to add it.



Now you can see four Renderer Features are added.

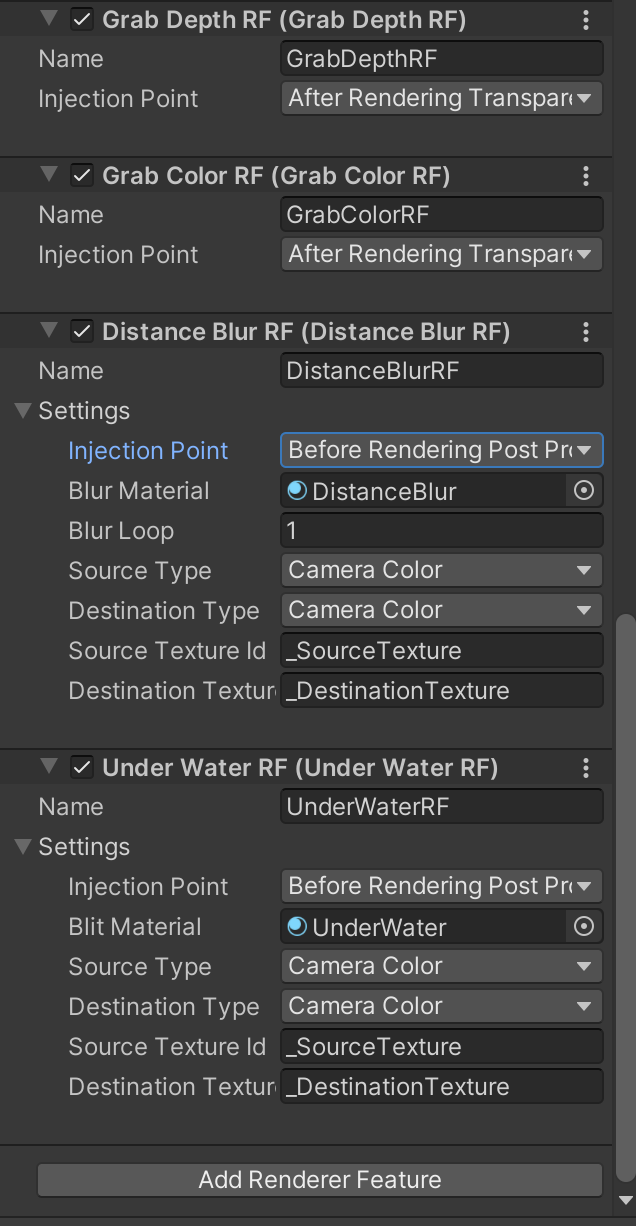
Change the Injection Point of the Grab Passes to ‘After Rendering Transparent’. Change Injection Point of the other two to ‘Before Rendering Post Processing’.

Find the blurring material in DragonXuWater/Materials/DistanceBlur.mat, click and drag it to the Blur Material slot of Distance Blur RF.

Find the under water material in DragonXuWater/Materials/UnderWater.mat, click and drag it to the Blit Material slot of Under Water RF.

Then it is all set, you don’t need to change any other parameters.

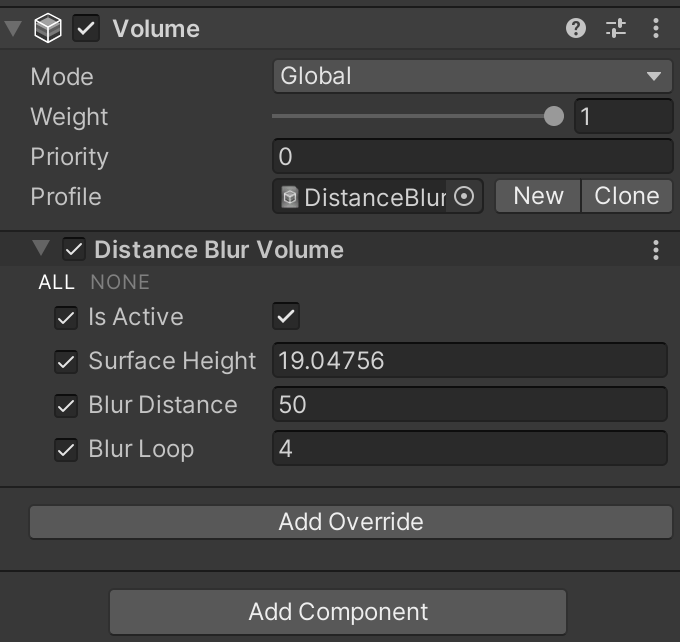
As is shown in the picture below.



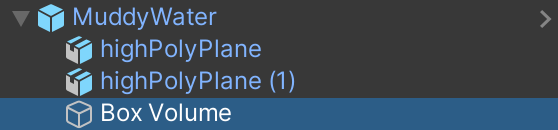
Now go to your scene, make sure the post processing is enabled in your camera component.

Create a global volume, select Distance Blur as the profile of the Volume component.

If you are using your own volume profile, simply click on Add Override and add Distance Blur Volume.



Then, you can select a water prefab in the prefab file and drag it into the scene. You can see a Box Volume as a child object.



The collider component of it defines the area where under water effects should be rendered. The Volume component of it renders the post processing under water effect.



**Volume Parameters**

**Under Water Volume**

**isActive**

is this volume component active.

**SurfaceHeight**

the height of water surface, you **don’t** have to manually set this. This is set by the script Water.cs.

**WaterColor**

The color of the water.

**CausticColor**

The color of the caustics.

**Visibility**

The farthest distance you can see under water.

**DistancePow**

Controls the smoothing curve of visibility.

**CausticScale**

The scale of the caustic texture.

**CausticStrength**

Intensity of the caustic effect.

**CausticMaxDepth**

The maximum depth where you can see caustic.

**FloatSpeed**

Floating speed of the caustics.

**FlowDirection**

The flowing direction of the caustics.

**NoiseScale**

The scale of the noise texture. The noise texture adds randomness to the caustic.

**NoiseStrength**

The intensity of the noise.

**BlurDistance**

Deprecated, just ignore this.



**Distance Blur Volume**

**isActive**

Is this volume component active.

**SurfaceHeight**

the height of blurring surface, you **DO** have to manually set this. We use the blurring effect to solve the moire issue of the water surface. All the pixels around the surface height will be blurred.

**BlurDistance**

Controls the distance factor of distance blur effect. When set with a smaller value, the blurring will be more intense.

**BlurLoop**

Controls how many times to iterate when blurring.