

Water System Guide

Game 2D Water Kit v1.4

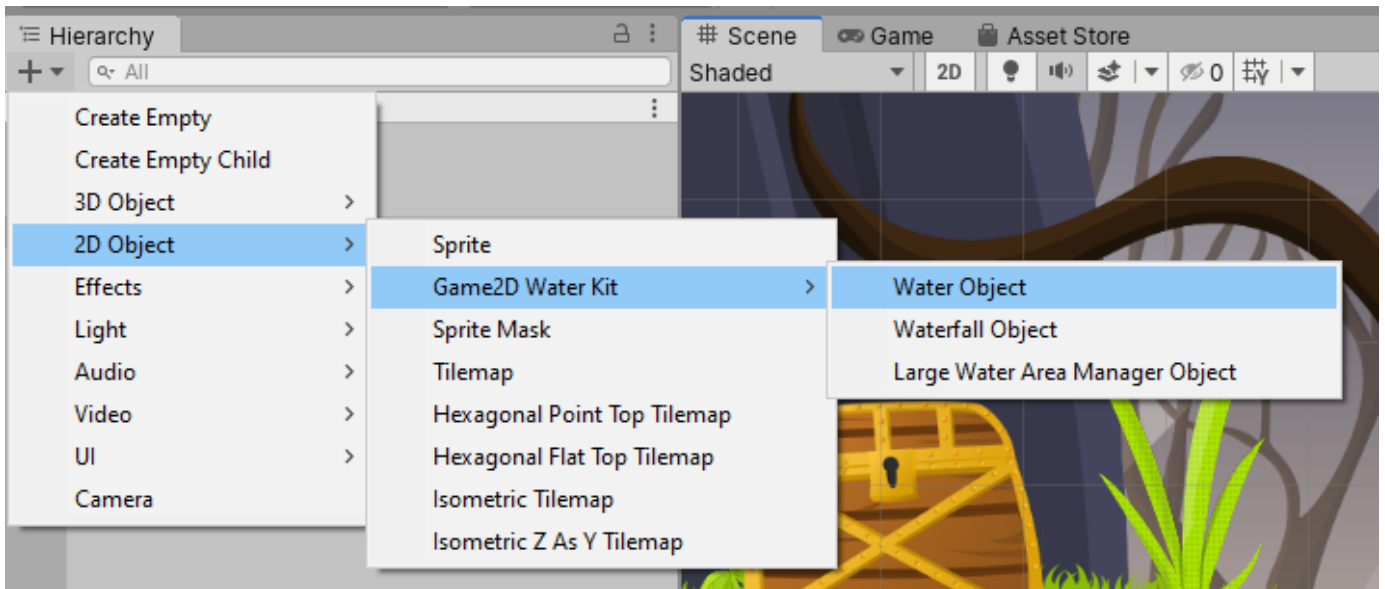
This is an auto-generated pdf file of the online guide
www.game2dwaterkit.com/water-system

Getting Started With The Water System

Creating A Water Object

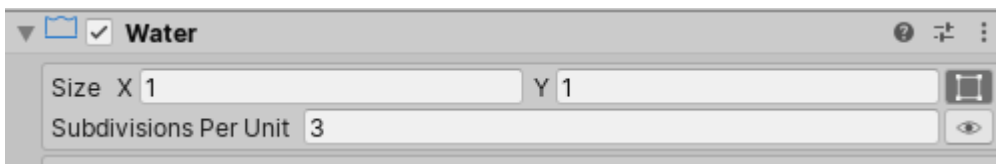
We create a water object from the Hierarchy's Create menu:

2D Object → Game2D Water Kit → Water Object



Resizing The Water Object

We resize the water object right in the scene view using the Rect Tool, or we can just provide the width and the height in the water component inspector.



Script Reference

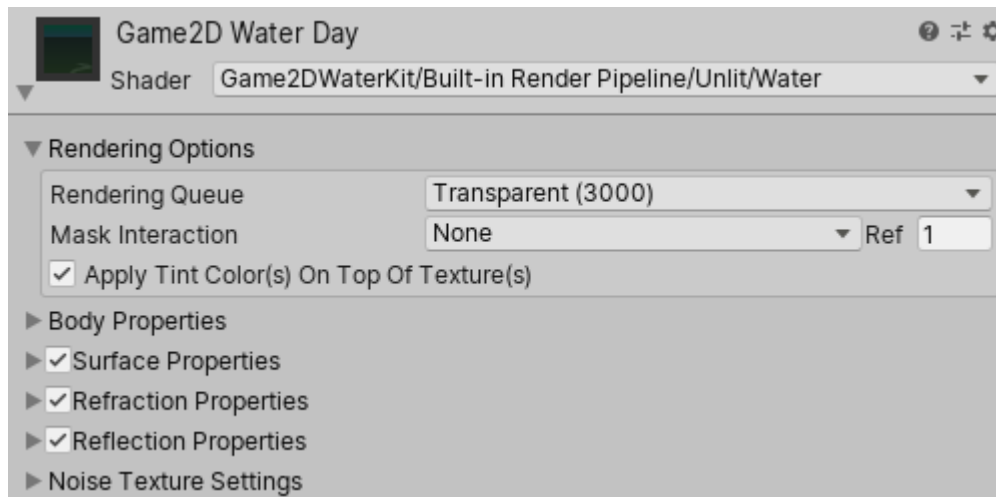
```
waterObject.MainModule.SetSize(new Vector2(width, height));
```

Sorting The Water Object Relative To Sprites

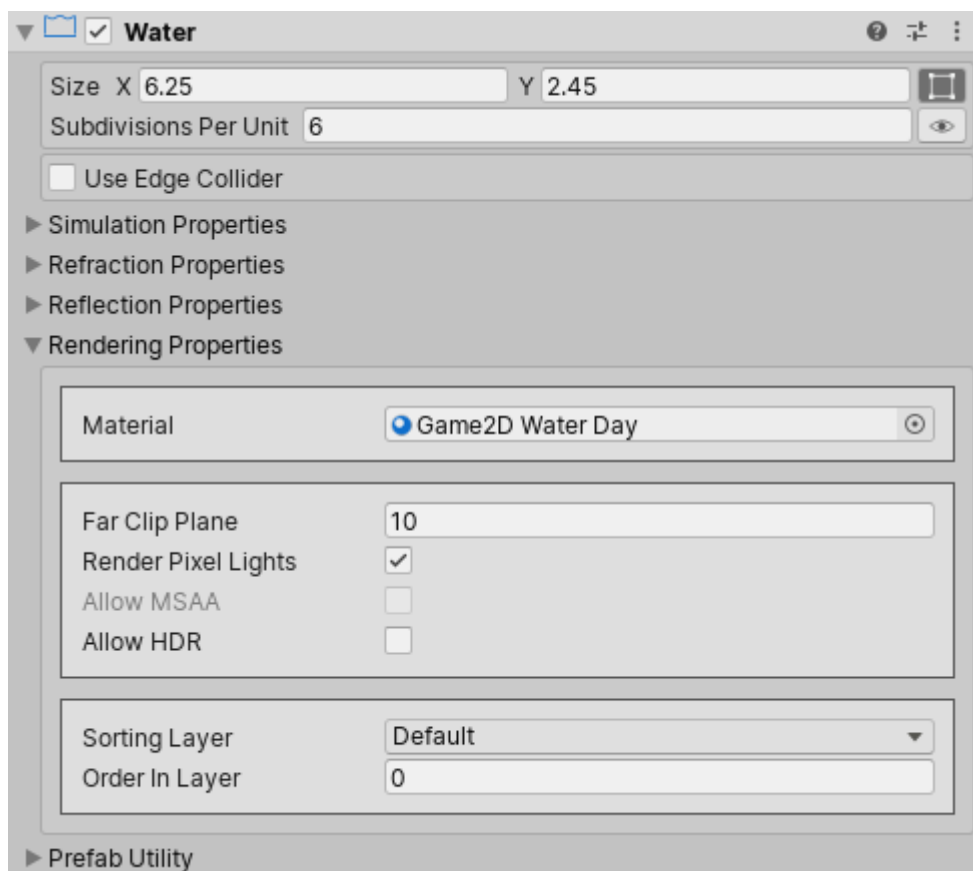
Before trying to sort the water object relative to sprites, we first need to make sure that the **Rendering Queue** property, under **Rendering Options** in the water material inspector, is set to Transparent.

Note

The **Rendering Queue** property is set to **Transparent** by default.



Then, under the **Rendering Properties** in the water component inspector, we specify the sorting layer as well as the order within that layer.



Script Reference

```
waterObject.RenderingModule.SortingLayerID =  
SortingLayer.NameToID("Default");  
waterObject.RenderingModule.SortingOrder = 0;
```

Info

We will look into the other rendering properties later in this guide.

Tweaking The Water Visuals



Water Body Properties

Game2D Water Day

ShaderGame2DWaterKit/Built-in Render Pipeline/Unlit/Water

▶ Rendering Options

▼ Body Properties

Color Properties

Color ModeGradient Color

Gradient Start

Gradient End

Gradient Offset0

Texture Properties

☐ Is A Texture Sheet

Columns & RowsC1R1

Frames Per Second0Lerp0.1

Opacity

Scrolling SpeedX0Y0

Tiling Properties

Tiling ModeRepeat

ScaleKeep Aspect Ratio☐

X2.5Auto☐

Y2.5Auto☐

OffsetX0Y0

☒ Distortion Effect

ScaleX4Y15

OffsetX0Y0

Strength0.246

Speed1.5

TilingX:1Y:1

▶ ☐ Surface Properties

▶ ☒ Refraction Properties

▶ ☒ Reflection Properties

▶ Noise Texture Settings

Body Color

Color Properties

Color ModeGradient Color

Gradient Start

Gradient End

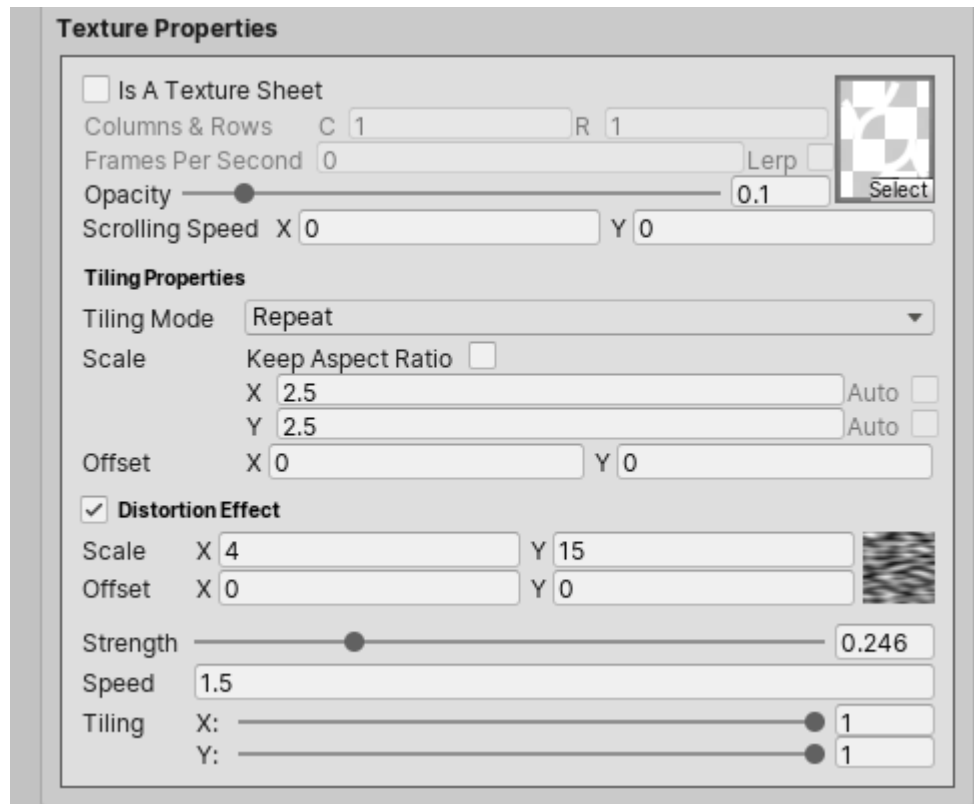
Gradient Offset0

We can set the water body color to either a *Solid Color* or a *Gradient Color*. If we choose to use a *Gradient Color*, the **Gradient Offset** property controls how much to shift the gradient-line midpoint position (where the middle of the color transition should be).

Info

The **Gradient Offset** property range: **-0.5** → **0.5**

Body Texture



We can apply a texture across the water body.

Warning

The texture should have its wrap-mode set to *Repeat* or *Mirror* in the texture import settings.

Body Texture Sheet Properties

This texture could be a *regular* texture, or a texture-sheet (a texture consisting of many frames) by toggling the **Is A Texture Sheet** property on, and then specifying the number of columns and rows and also setting how many frames to play per second.

Body Texture Opacity

The **Opacity** property controls the visibility of the texture.

Body Texture Scrolling Speed

We can make the texture scroll, in the X and/or the Y directions, by tweaking the **Scrolling Speed** property.

Body Texture Tiling Mode

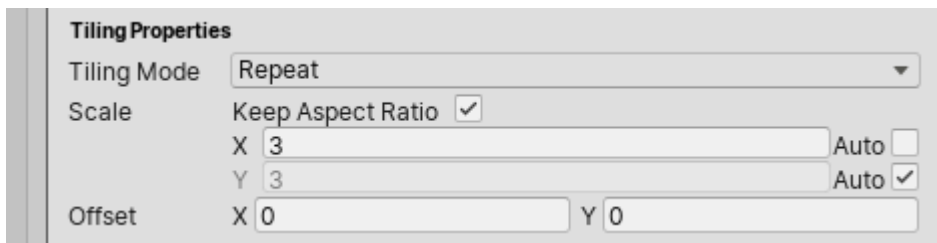
Regarding the texture tiling properties, there are two tiling modes:

- **Stretch:** The texture stretches when the water object size changes, always keeping the same number of tiles we specify for the X and Y directions.



The screenshot shows the 'Tiling Properties' panel with 'Stretch' selected in the 'Tiling Mode' dropdown. The 'Tiling' section has 'Keep Aspect Ratio' unchecked. The 'X' and 'Y' input fields are set to 3 and 6 respectively, with 'Auto' checkboxes to their right. The 'Offset' section has 'X' and 'Y' input fields both set to 0.

- **Repeat:** The texture repeats when the water object size changes. In this mode, we specify the scale in units of a single tile.

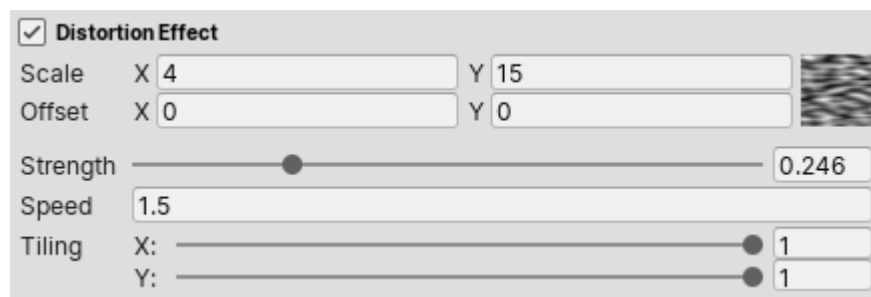


The screenshot shows the 'Tiling Properties' panel with 'Repeat' selected in the 'Tiling Mode' dropdown. The 'Scale' section has 'Keep Aspect Ratio' checked. The 'X' and 'Y' input fields are set to 3, with 'Auto' checkboxes to their right. The 'Offset' section has 'X' and 'Y' input fields both set to 0.

Body Texture Offset

We can provide a texture offset regardless of the selected texture **Tiling Mode**.

Body Texture Distortion Effect



The screenshot shows the 'Distortion Effect' panel with the 'Distortion Effect' checkbox checked. The 'Scale' section has 'X' and 'Y' input fields set to 4 and 15 respectively. The 'Offset' section has 'X' and 'Y' input fields set to 0. The 'Strength' section has a slider and a numeric input field set to 0.246. The 'Speed' section has a numeric input field set to 1.5. The 'Tiling' section has 'X' and 'Y' input fields both set to 1.

DISTORTION SCALE - OFFSET

The `Mathf.PerlinNoise(x,y)` function is used to *sample* the Perlin noise texture values. The **Scale** and **Offset** properties controls the sampled area size and origin, respectively.

Warning

The **Scale** And **Offset** properties are used to generate the noise texture (which actually happens only in the editor), and as such they are not animatable.

DISTORTION STRENGTH

The **Strength** property, as the name suggests, controls how strong the distortion effect is.

DISTORTION SPEED

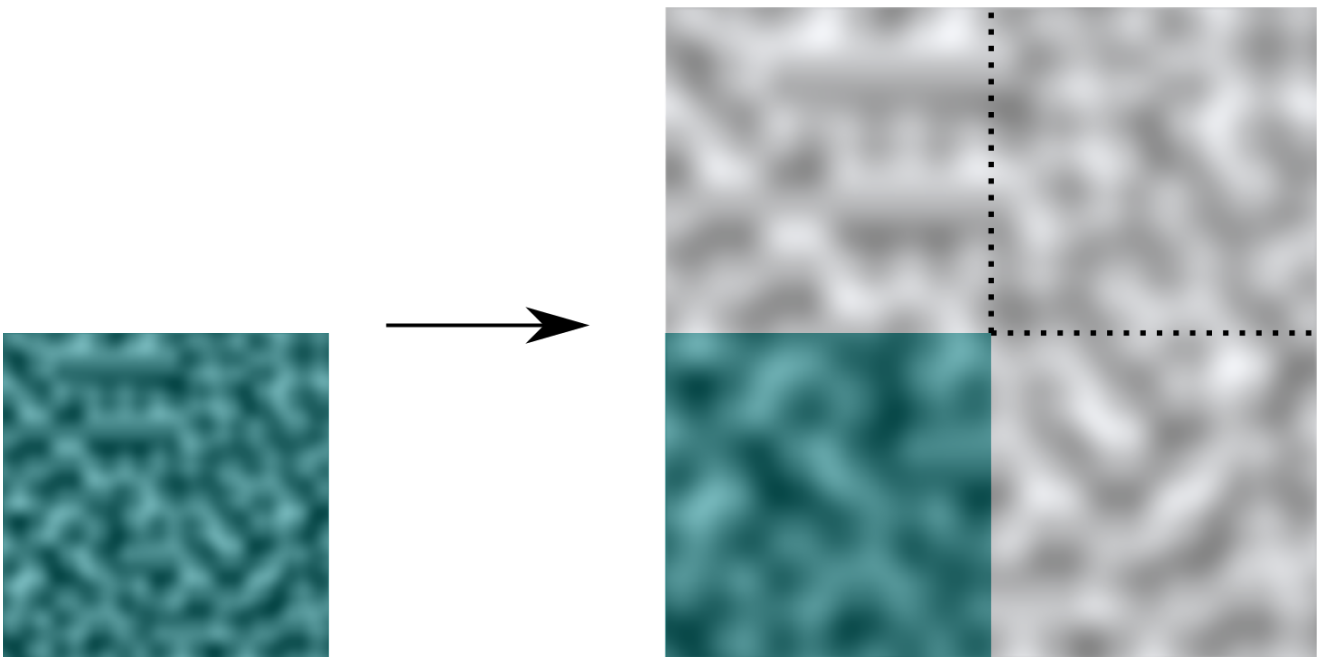
The **Speed** property controls the noise texture scrolling speed.

DISTORTION TILING

The **Tiling** property controls the scale of a noise texture tile relative to a water body texture tile.

Example

A value of **0.5**, for both the X and Y axis, will apply the noise texture across **4** water body texture tiles.



Tip

We can use the **Tiling** property to lower the distortion strength across one axis, and keep the full strength across the other.

Water Surface

Game2D Water Day

Shader Game2DWaterKit/Built-in Render Pipeline/Unlit/Water

▶ Rendering Options

▶ Body Properties

▼ ☒ Surface Properties

Thickness 0.02496

☐ Submerged Level 0

Color Properties

Color Mode Solid Color

Color

Texture Properties

☐ Is A Texture Sheet

Columns & Rows C 1 R 1

Frames Per Second 0 Lerp ☐

Opacity 0.5

Scrolling Speed X 0 Y 0

Tiling Properties

Tiling Mode Repeat

Scale Keep Aspect Ratio ☐

X 1 Auto ☐

Y 1 Auto ☐

Offset X 0 Y 0

☐ Distortion Effect

Scale X 1 Y 1

Offset X 0 Y 0

Strength 0.025

Speed 2.5

Tiling X: 1

Y: 1

▶ ☒ Refraction Properties

▶ ☒ Reflection Properties

▶ Noise Texture Settings

Surface Thickness

▼ ☒ Surface Properties

Thickness 0.02496

☐ Submerged Level 0

The **Thickness** property sets the thickness of the water surface line.

Note

We will look into the **Submerge Level** property later when we go through the water refraction properties.

Surface Color and Texture

The screenshot shows a software interface for configuring surface properties, divided into two main sections: Color Properties and Texture Properties.

Color Properties

- Color Mode:** A dropdown menu set to "Solid Color".
- Color:** A horizontal color bar showing a teal/cyan color, with a small icon to its right for color selection.

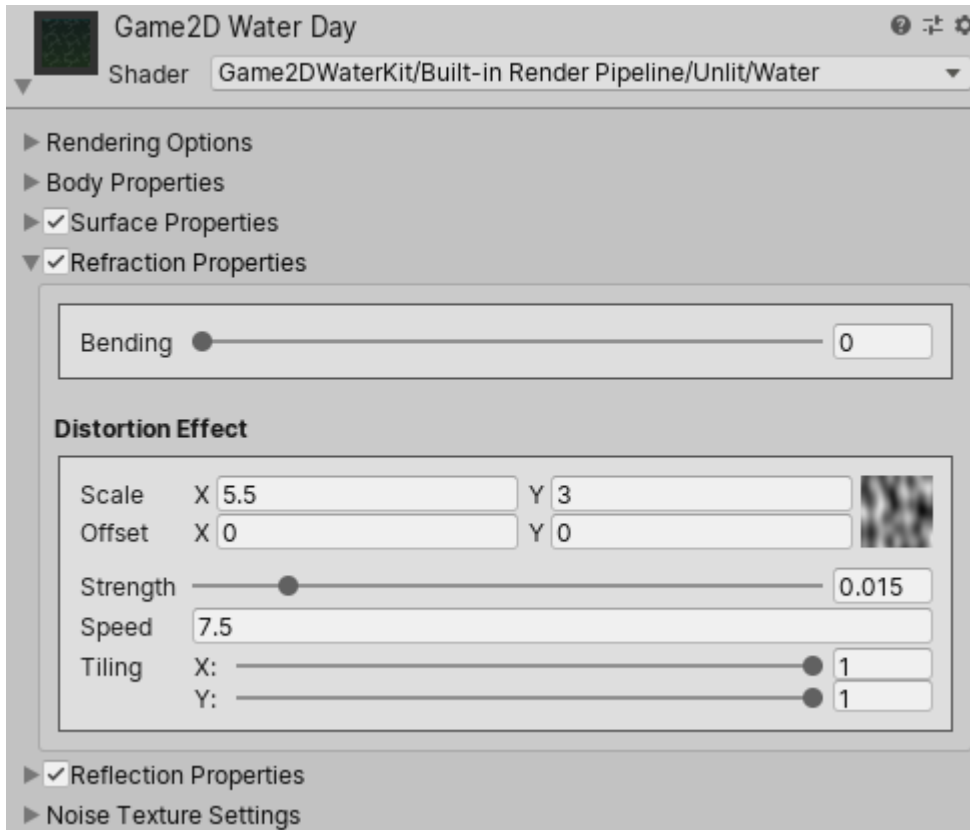
Texture Properties

- Is A Texture Sheet:** An unchecked checkbox.
- Columns & Rows:** Input fields for "C" (1) and "R" (1).
- Frames Per Second:** An input field set to 0.
- Opacity:** A slider bar with a value of 0.5.
- Scrolling Speed:** Input fields for "X" (0) and "Y" (0).
- Texture Preview:** A small square icon showing a black and white checkerboard pattern, with a "Select" button below it.
- Tiling Properties**
 - Tiling Mode:** A dropdown menu set to "Repeat".
 - Scale:** A section with "Keep Aspect Ratio" (unchecked) and input fields for "X" (1) and "Y" (1), each with an "Auto" checkbox.
 - Offset:** Input fields for "X" (0) and "Y" (0).
- Distortion Effect:** An unchecked checkbox.
 - Scale:** Input fields for "X" (1) and "Y" (1).
 - Offset:** Input fields for "X" (0) and "Y" (0).
 - Strength:** A slider bar with a value of 0.025.
 - Speed:** An input field set to 2.5.
 - Tiling:** Input fields for "X" (1) and "Y" (1), each with a slider bar.

Just like the water body, we can tweak the surface tint color and apply a texture across the surface.

The description of the surface color and texture properties is exactly the same as the water body texture properties, as discussed [here](#).

Water Refraction

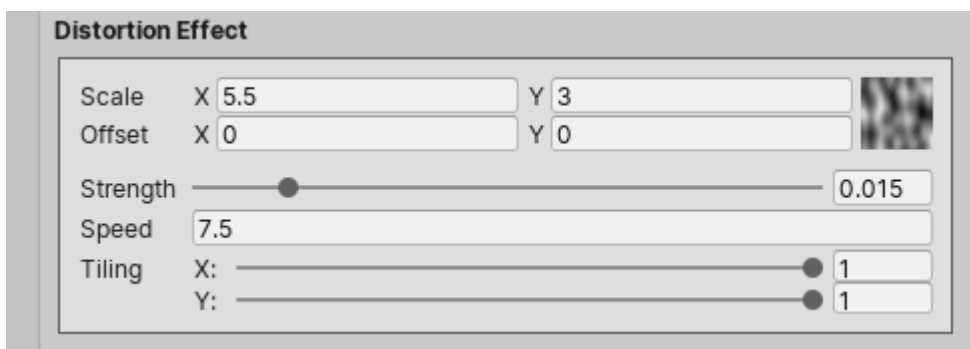


Refraction Bending

The bending property controls how much we would like to shift the submerged portion of an object relative to the other portion above the water.



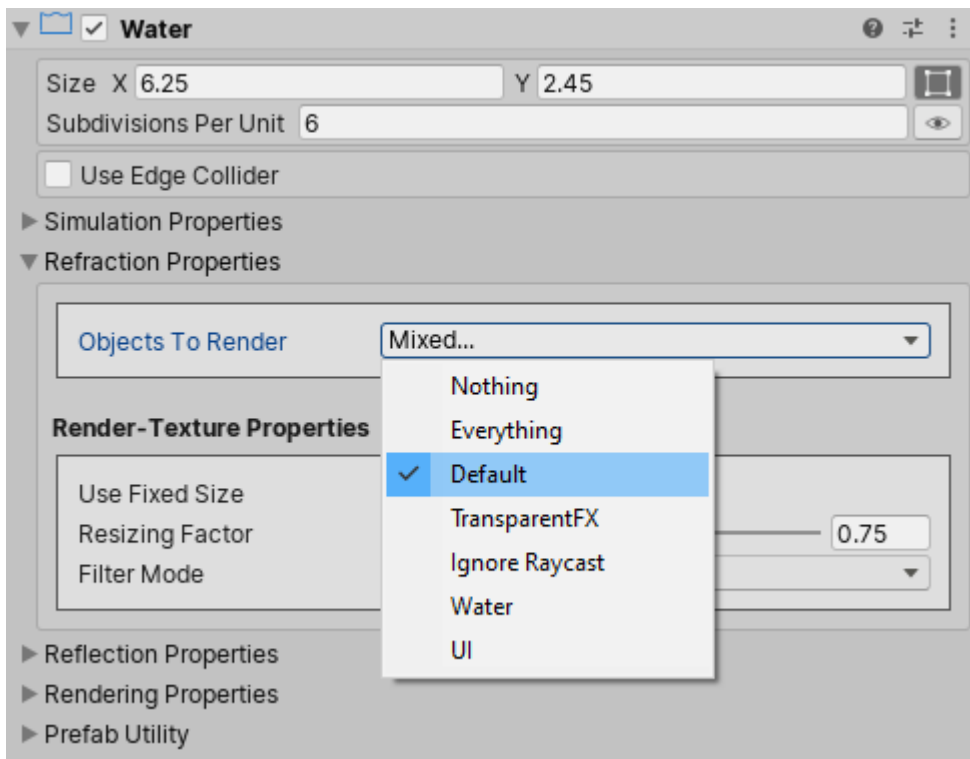
Refraction Distortion Effect



The refraction distortion properties description is the same as the water body distortion effect properties, discussed [here](#).

Refraction Layers

We select which layers to include in the refraction rendering using the ***Objects To Render*** property under the **Refraction Properties** in the water component inspector.

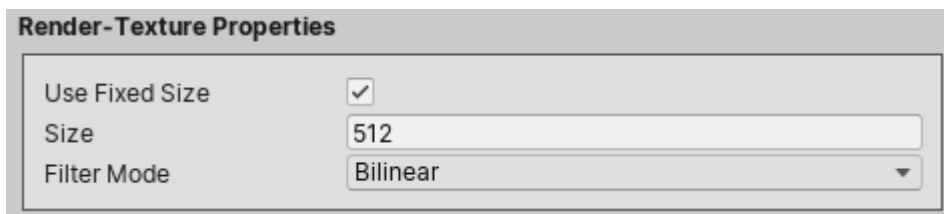


i Script Reference

```
waterObject.RenderingModule.Refraction.CullingMask =
LayerMask.GetMask("Default");
```

Refraction Render-Texture Properties

If the **Use Fixed Size** property is toggled on, the **Size** property sets the refraction render-texture width and height.

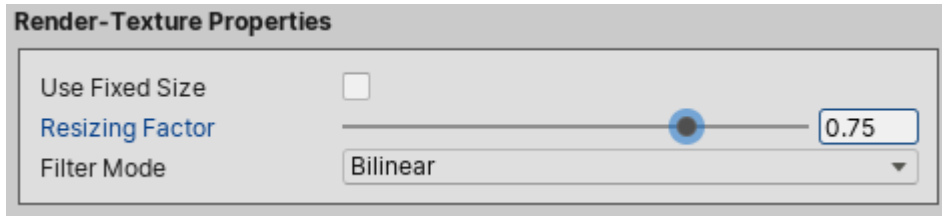


i Script Reference

```
waterObject.RenderingModule.Refraction.RenderTextureUseFixedSize = true;
waterObject.RenderingModule.Refraction.RenderTextureFixedSize = 512;
```

But, if the **Use Fixed Size** property is toggled off, the refraction render-texture will have a dynamic size, and the render-texture width and height are in this case equal to the the water object visible

area on screen width and height. We can even downscale this computed size by lowering the **Resizing Factor** property value.



Script Reference

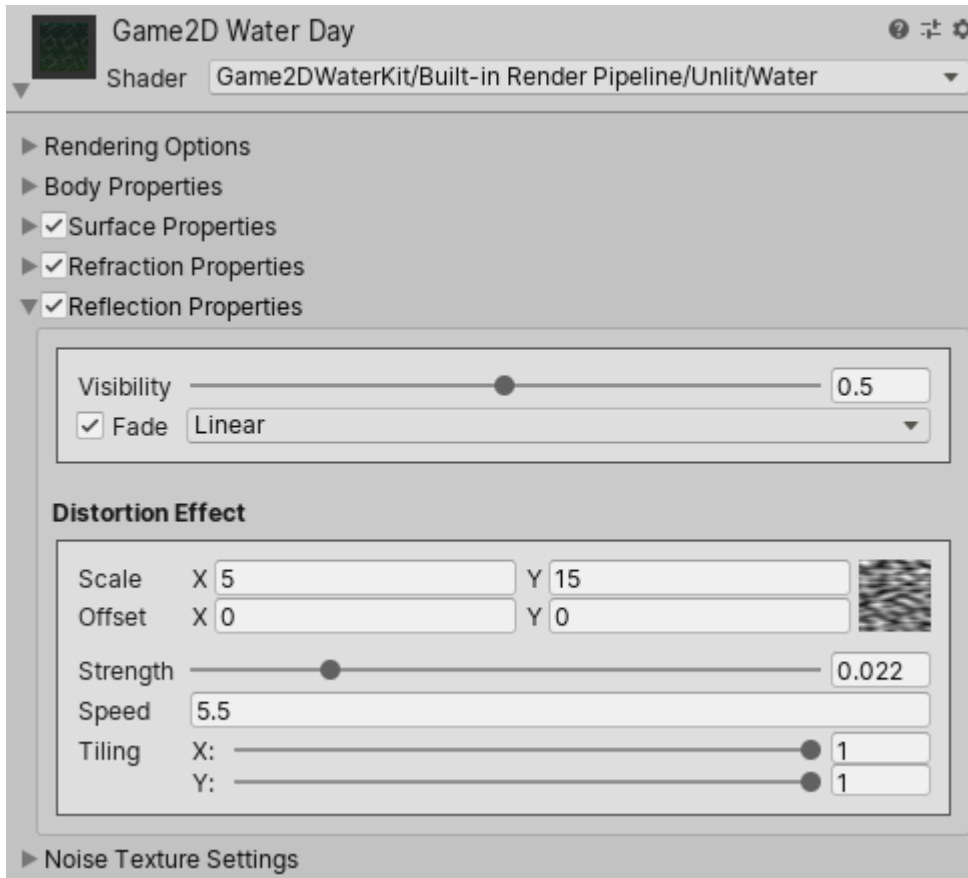
```
waterObject.RenderingModule.Refraction.RenderTextureUseFixedSize = false;  
waterObject.RenderingModule.Refraction.RenderTextureResizingFactor = 0.75f;
```

Lastly, We can set the refraction render-texture **Filter Mode** property to either *Bilinear* or *Point*.

Script Reference

```
waterObject.RenderingModule.Refraction.RenderTextureFilterMode =  
FilterMode.Bilinear;
```

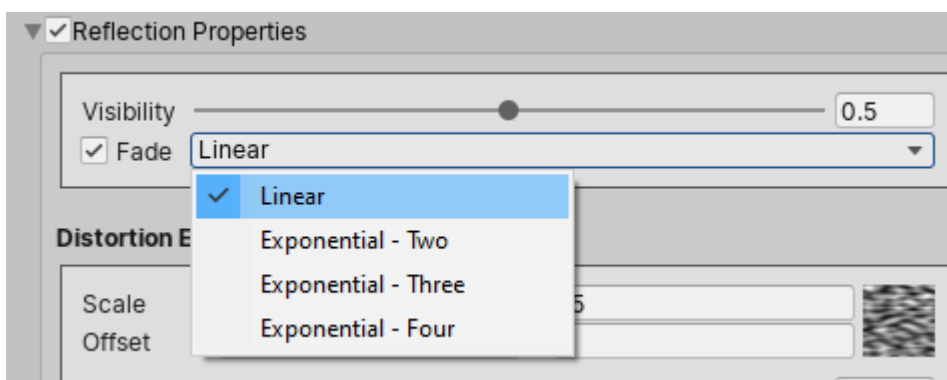
Water Reflection



Reflection Visibility

We control the visibility of the rendered reflection image under the **Reflection Properties** in the water material inspector.

Reflection Fade

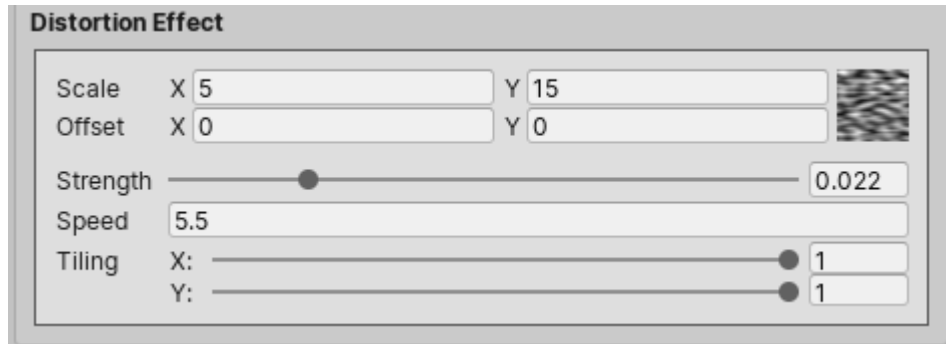


We can make the reflection fade vertically along the water object, starting from the top at full opacity. To do this, we toggle the **Fade** property on, and proceed to select the desired fade *speed*:

- Linear

- Exponential - Two
- Exponential - Three
- Exponential - Four

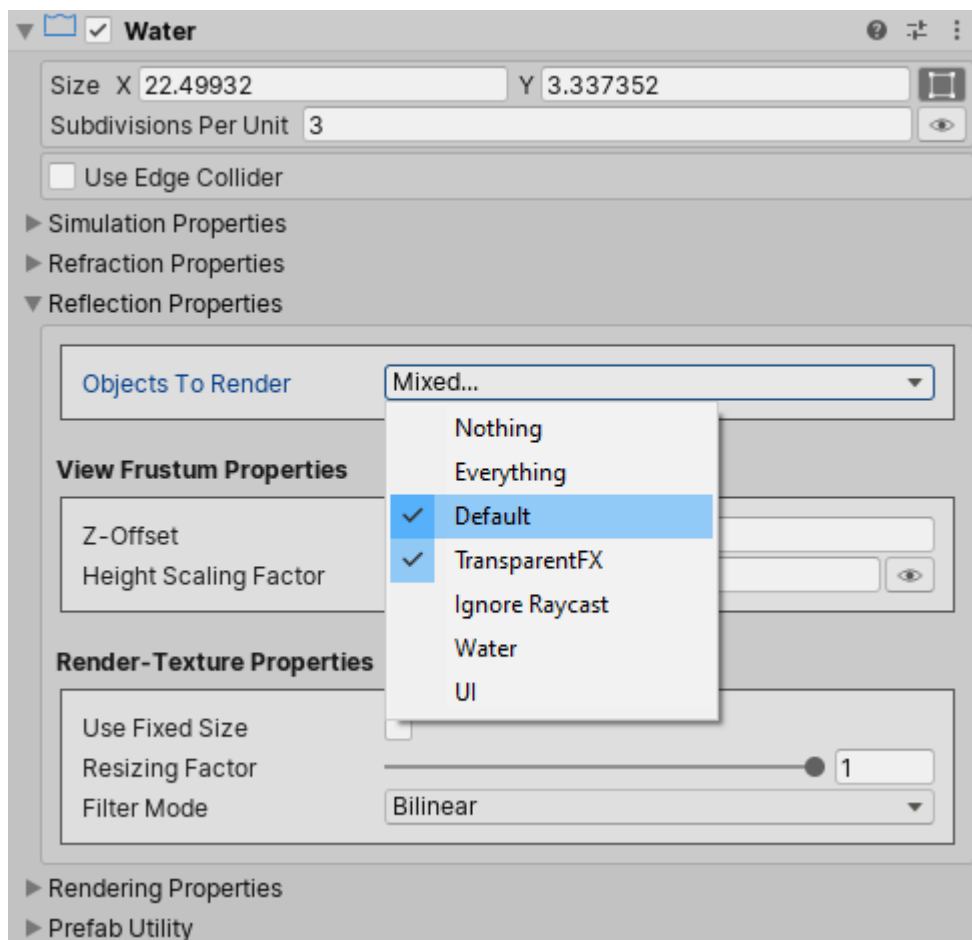
Reflection Distortion Effect



The reflection distortion properties description is the same as the water body distortion effect properties, as discussed [here](#).

Reflection Layers

We use select which layers to include in the reflection rendering using the ***Objects To Render*** property under the **Reflection Properties** in the water component inspector.

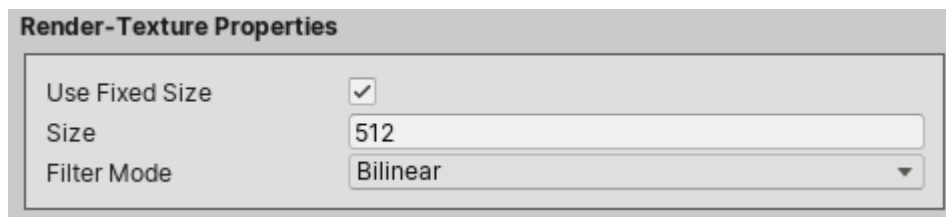


Script Reference

```
waterObject.RenderingModule.Reflection.CullingMask =  
LayerMask.GetMask("Default", "TransparentFX");
```

Reflection Render-Texture Properties

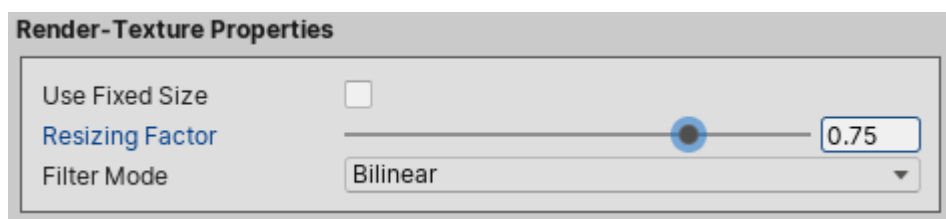
If the **Use Fixed Size** property is toggled on, the **Size** property sets the reflection render-texture width and height.



Script Reference

```
waterObject.RenderingModule.Reflection.RenderTextureUseFixedSize = true;  
waterObject.RenderingModule.Reflection.RenderTextureFixedSize = 512;
```

Otherwise, the refraction render-texture will have a dynamic size, and the render-texture width and height are equal to the the water object visible area on screen width and height. We can even downscale this computed size by lowering the **Resizing Factor** property value.



Script Reference

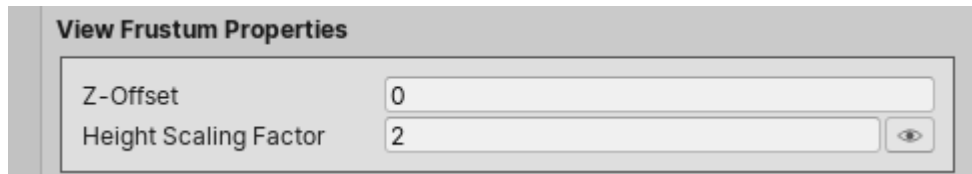
```
waterObject.RenderingModule.Reflection.RenderTextureUseFixedSize = false;  
waterObject.RenderingModule.Reflection.RenderTextureResizingFactor = 0.75f;
```

Lastly, We can set the reflection render-texture **Filter Mode** property to either *Bilinear* or *Point*.

Script Reference

```
waterObject.RenderingModule.Reflection.RenderTextureFilterMode =  
FilterMode.Bilinear;
```

Reflection Frustum Properties



Frustum Z-Offset

The **Z-Offset** property controls where to start rendering the water reflection relative to the water object position. This might be useful if we would like to render the reflection of certain objects that are in front of the water object.

Script Reference

```
waterObject.RenderingModule.ReflectionZOffset = 0f;
```

Frustum Height Scaling Factor

The reflection frustum size is equal to the visible water area size. We can expand this frustum vertically by setting the **Height Scaling Factor** property.

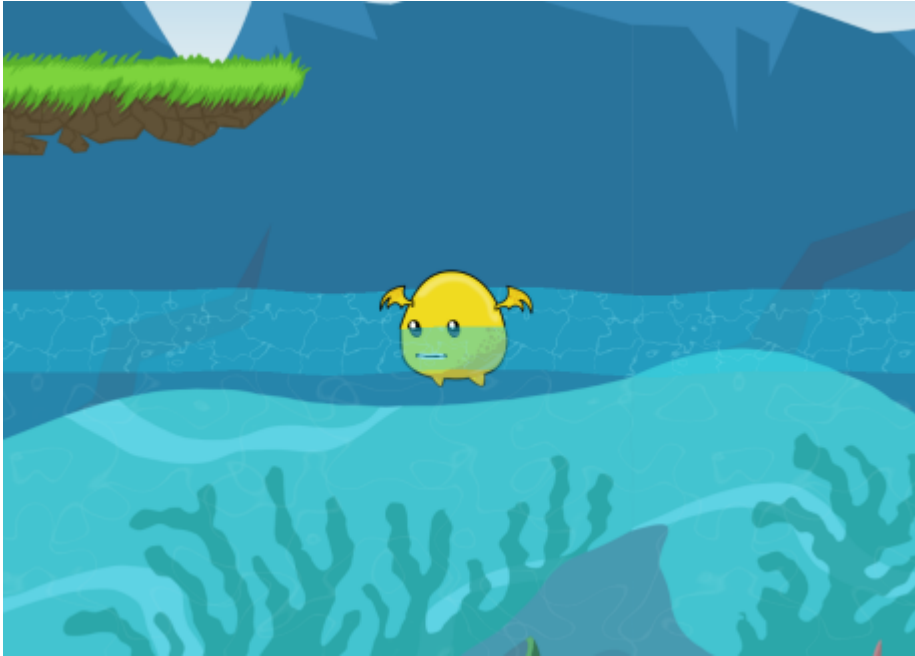
Tip

We can visualize the reflection frustum bounds in the scene-view by toggling the 'eye icon' on.

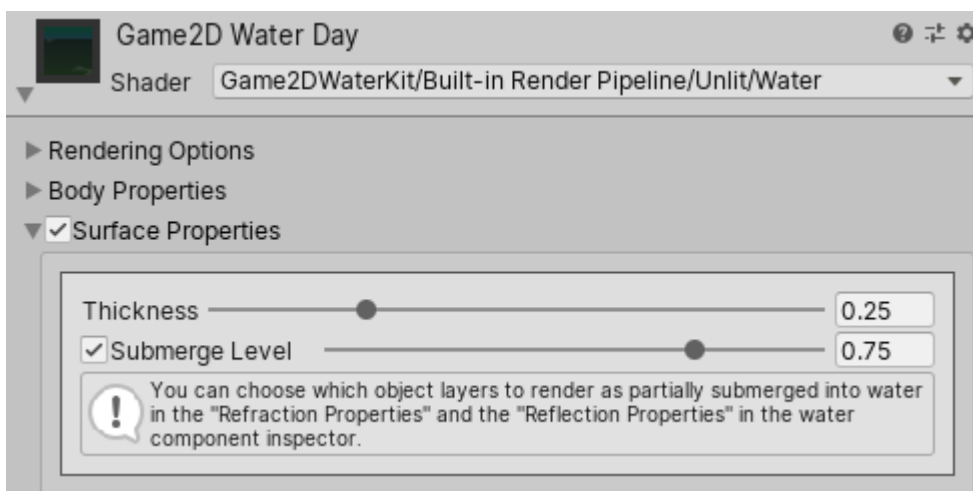
Script Reference

```
waterObject.RenderingModule.Reflection.ViewingFrustumHeightScalingFactor =  
2f;
```

Water Fake Perspective Effect



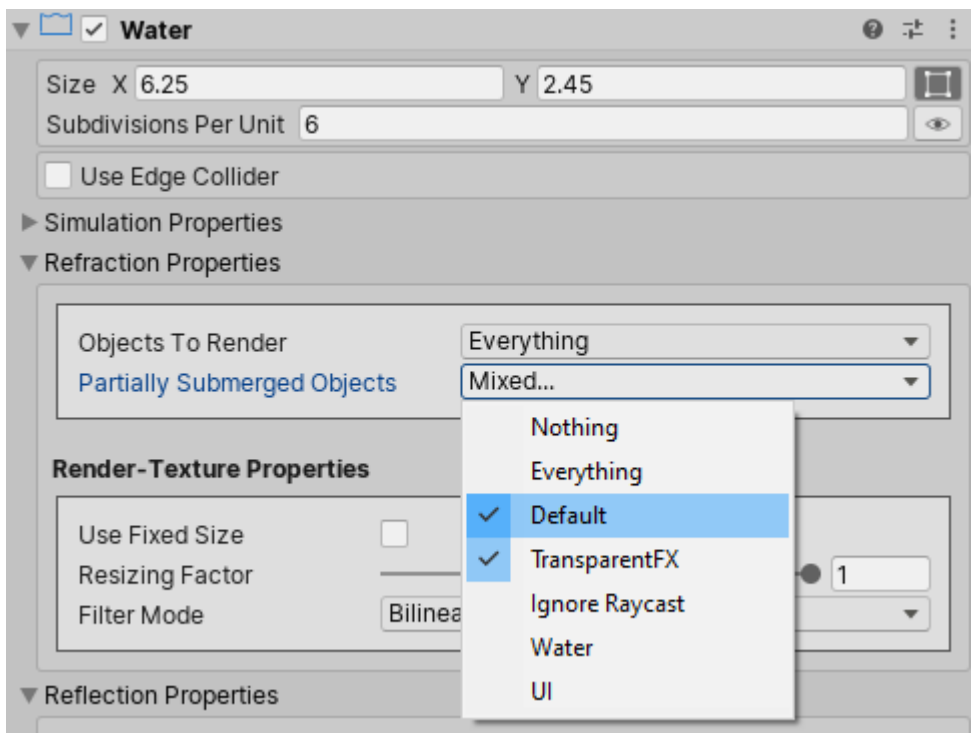
The fake perspective effect consists of rendering certain objects that intersect the submerge level as partially submerged into water, adding a sense of depth to the water.



We can enable this effect under the Surface Properties in the water material inspector by toggling the Submerge Level property on and adjusting the submerge level position.

Fake Perspective Effect Layers

Fake Perspective Effect - Refraction Layers

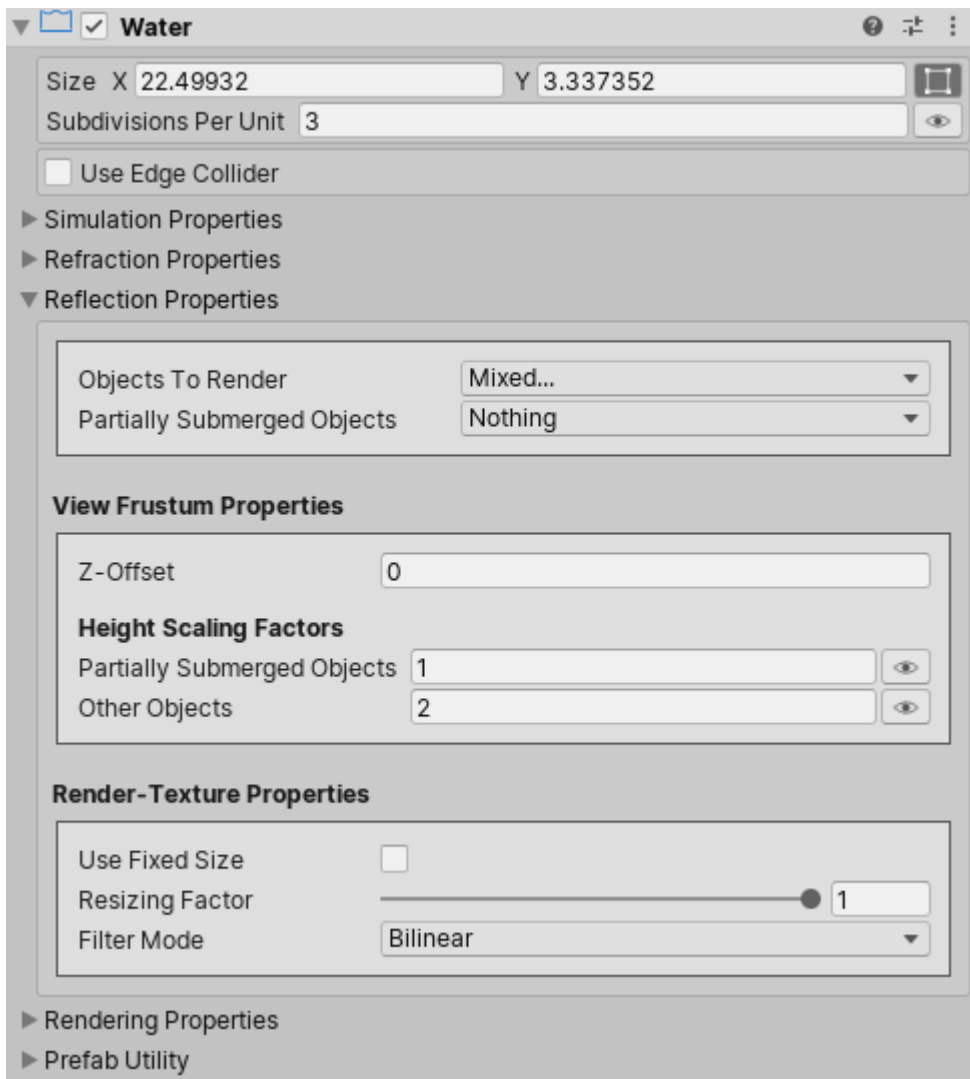


Under the **Refraction Properties** in the water component inspector, we set the ***Partially submerged Objects*** property to select which objects layers to render as partially submerged into water when they intersect the submerge level.

Script Reference

```
waterObject.RenderingModule.RefractionPartiallySubmergedObjects.CullingMask  
= LayerMask.GetMask("Default", "TransparentFX");
```

Fake Perspective Effect - Reflection Layers

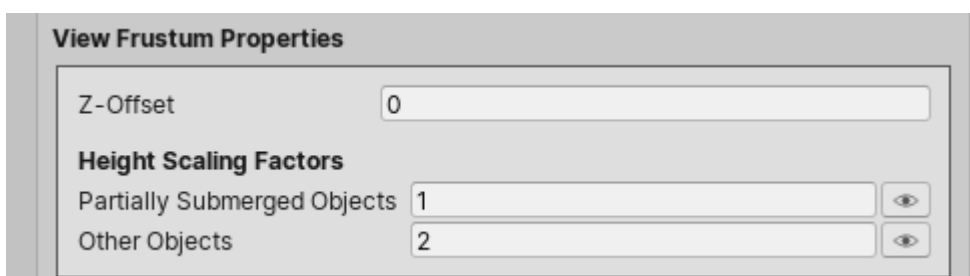


If we would like to render the reflection of certain partially submerged objects, we make sure to select their layers under the **Reflection Properties** in the water component inspector.

Script Reference

```
waterObject.RenderingModule.ReflectionPartiallySubmergedObjects.CullingMask  
= 0;
```

Fake Perspective Effect - Reflection View Frustum Properties



We can expand the partially submerged objects reflection frustum as well as the other objects reflection frustum vertically by settings their respective scaling factors properties.

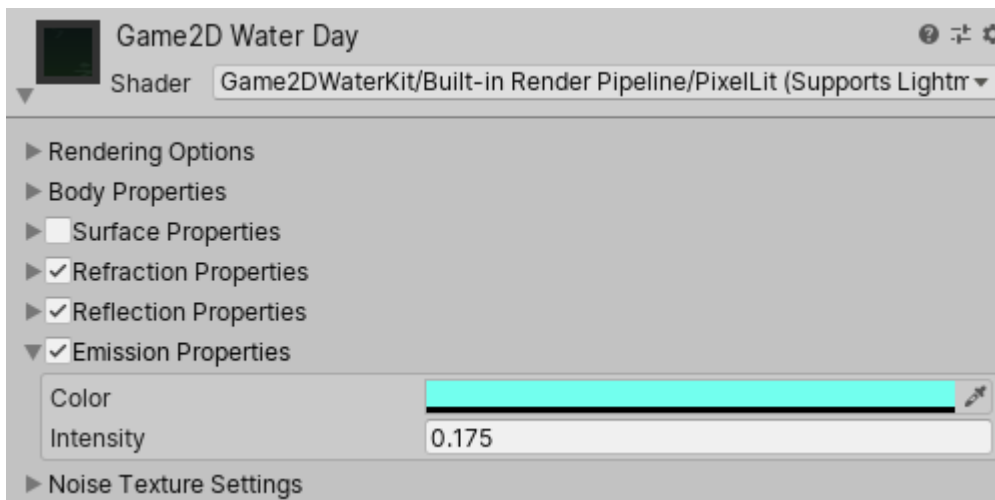
Tip

We can visualize the reflection frustums bounds in the scene-view by toggling the 'eye icon' on.

Script Reference

```
waterObject.RenderingModule.ReflectionPartiallySubmergedObjects.ViewingFrustum  
= 1f;
```

Water Emission Effect



If the water material is using one of the lit shaders, then we can set an **Emission Color** and also tweak its **Intensity** under the **Emission Properties** in the water material inspector.

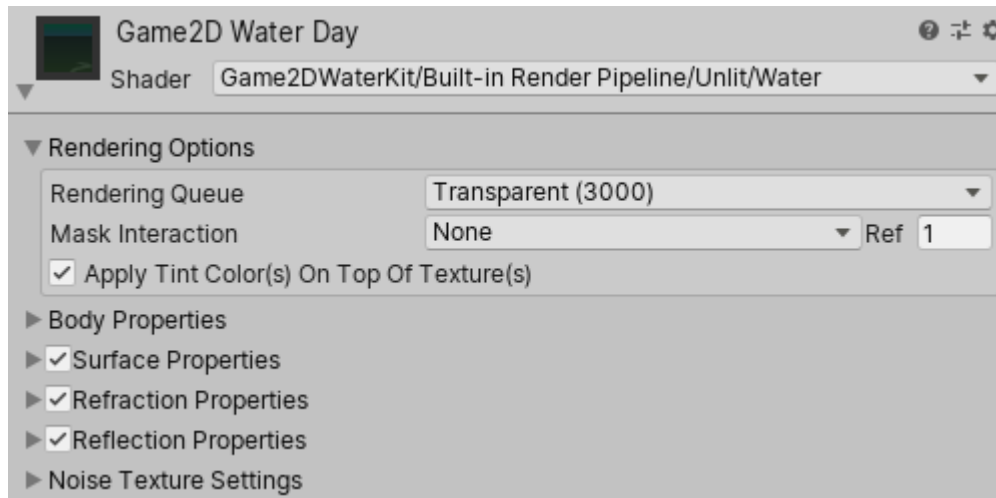
Info

The asset comes with **5** lit shaders, **3** for the **Builtin Render Pipeline** (Pixel-Lit, Vertex-Lit, Vertex-Lit-Only-Directional-Lights), **1** for the **Universal Render Pipeline (URP)**, and another one for the **Lightweight Render Pipeline (LWRP)**.

Rendering Properties

We have already looked into some of the rendering properties in the [previous section of this guide](#). Now, it's time to go through the other properties!

Rendering Properties - Material Inspector

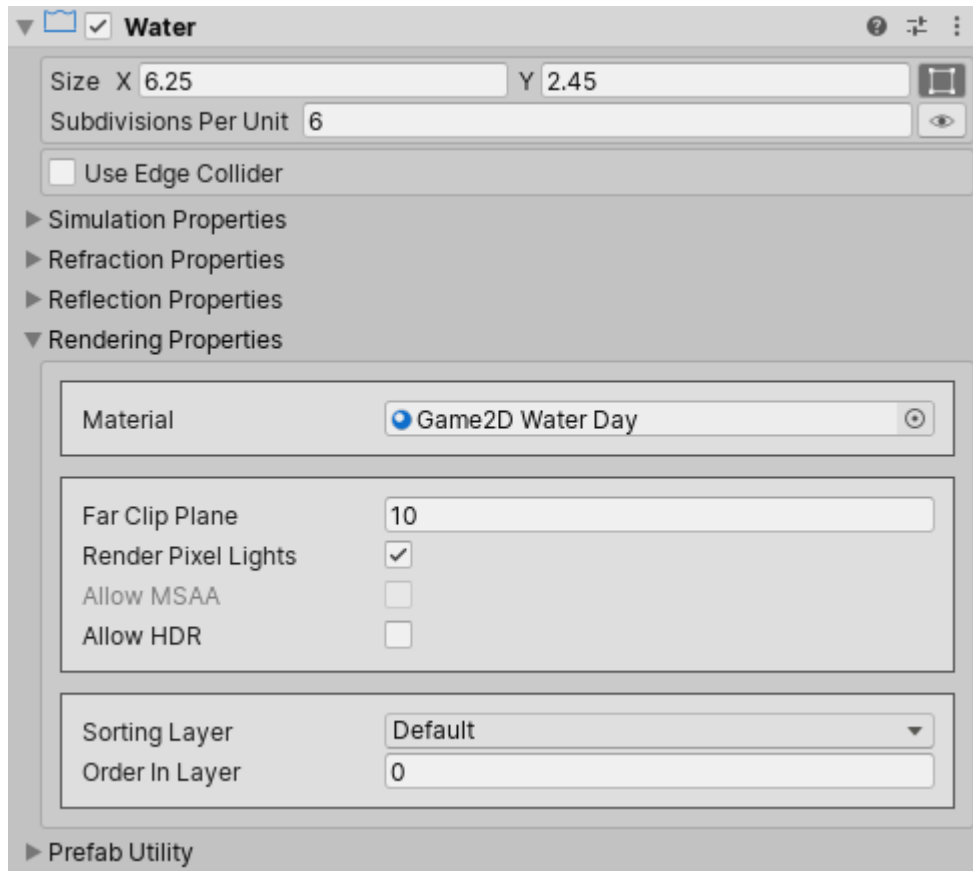


Under the **Rendering Options** in the water material inspector, the **Mask Interaction** property controls how the water object behaves when interacting with a *Sprite Mask*. We can set the mask interaction to one of the following options:

- None : No interaction
- Visible Inside Mask : the water is visible where the Sprite Mask overlays it, but not outside it.
- Visible Outside Mask : the water is visible outside the Sprite Mask, but not inside it.

The **Apply Tint Color(s) On Top Of Texture(s)** property controls in which order, the water body and the water surface, tint colors and textures are applied. If this property is toggled on, the textures are applied first and the tint colors are applied second, otherwise textures are applied last.

Rendering Properties - Component Inspector



Under the **Rendering Properties** in the water component inspector, the **Far Clip Plane** property sets the furthest point relative to the water object position that gets included in the water refraction/reflection rendering.

The **Render Pixel Lights** property controls whether or not the rendered objects are affected by pixel lights.

Info

The **Render Pixel Lights** property is ignored when working with the URP/LWRP.

We can also activate/deactivate the High Dynamic Range (HDR) and the Multisample Anti-Aliasing (MSAA) rendering.

Info

The MSAA rendering is activated only if the Antialiasing is enabled in the project quality settings or in the URP/LWRP settings asset.

Script Reference

```
waterObject.RenderingModule.FarClipPlane = 10f;  
waterObject.RenderingModule.RenderPixelLights = true;  
waterObject.RenderingModule.AllowMSAA = false;  
waterObject.RenderingModule.AllowHDR = false;
```

Tweaking The Water Behavior



We tweak all the water behavior properties in the water component inspector.

Water

Size X 6.25 Y 2.45

Subdivisions Per Unit 6

☐ Use Edge Collider

Simulation Properties

Simulation Preview

▶ Enter Simulation Mode

Buoyancy Effector Properties

Surface Level

0.02

Wave Properties

Stiffness 50

Spread 50

Damping 0.05

Use Custom Boundaries ☐

▶ On-Collision Ripples Properties

▶ ☒ Constant Ripples Properties

▶ Script-Generated Ripples Properties

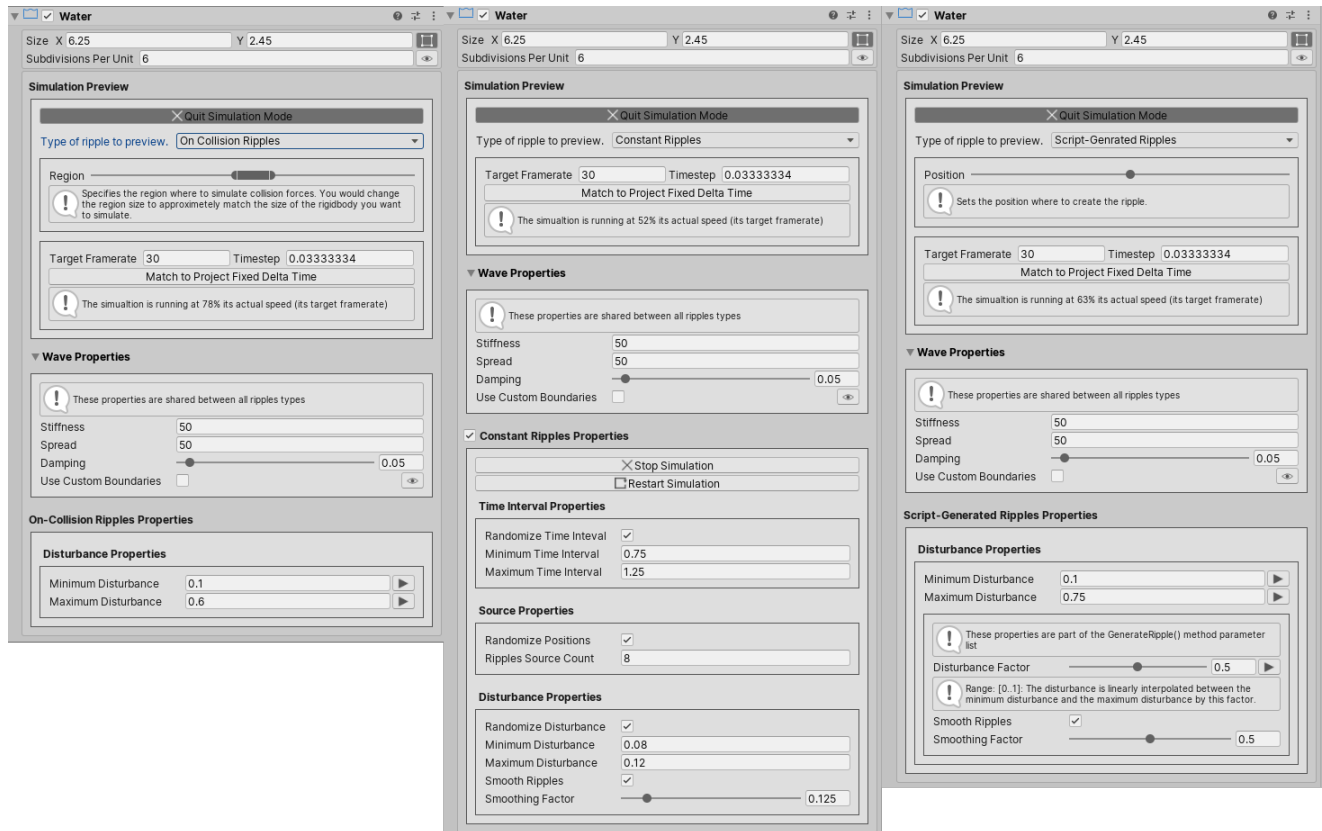
▶ Refraction Properties

▶ Reflection Properties

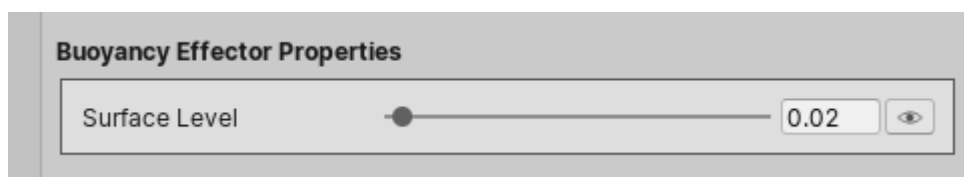
▶ Rendering Properties

▶ Prefab Utility

We can use the simulation mode to quickly and easily tweak and test the water system simulation properties, in realtime, right in the edit mode without the need to switch to play mode. All we need to do is to press the **Enter Simulation Mode** button.



Buoyancy Effector Properties

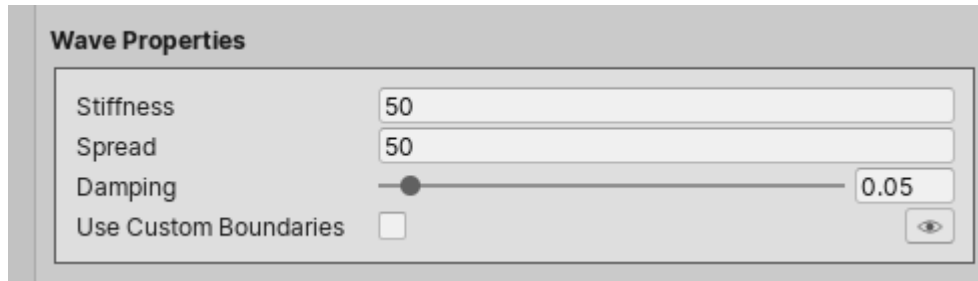


Under the **Simulation Properties**, the **Surface Level** property controls the surface location of the buoyancy effector. When rigidbodies are intersecting or below the surface level, buoyancy forces are applied to them.

Script Reference

```
waterObject.AttachedComponentsModule.BuoyancyEffectorSurfaceLevel = 0.02f;
```

Wave Properties



The image shows a 'Wave Properties' panel with four controls: 'Stiffness' and 'Spread' are text input fields both containing the value '50'; 'Damping' is a slider with a dot in the middle and a numeric field on the right showing '0.05'; and 'Use Custom Boundaries' is an unchecked checkbox. There is an eye icon to the right of the checkbox.

The **Stiffness** property controls the frequency of the wave vibration (how fast the water oscillates). A low value make the water oscillate slowly, while a high value make it oscillate quickly.

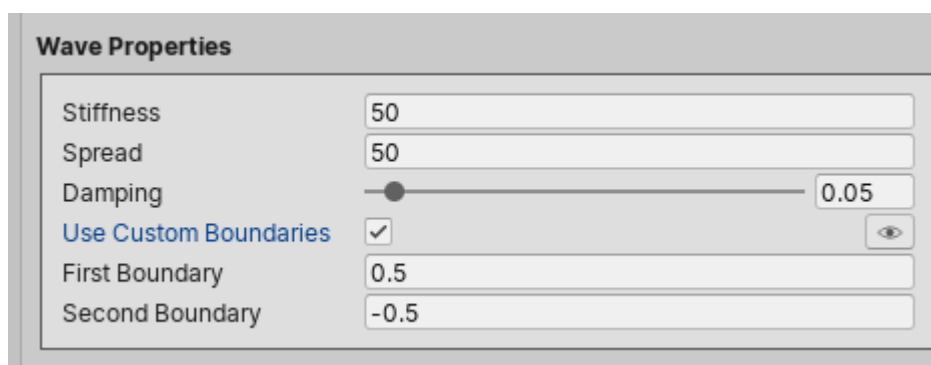
The **Spread** property controls how fast the waves propagate.

The **Damping** property controls how fast the water stops oscillating. The lower the damping value is, the longer the water keeps oscillating.

Script Reference

```
waterObject.SimulationModule.Stiffness = 50f;  
waterObject.SimulationModule.Spread = 50f;  
waterObject.SimulationModule.Damping = 0.05f;
```

When a wave reaches a boundary, it bounces back. And by default, the simulation boundaries are the left and the right water edges. But, we could define our own boundaries by checking the **"Use Custom Boundaries"** property, and defining the custom boundaries locations.



The image shows the 'Wave Properties' panel with additional fields. 'Stiffness' and 'Spread' are '50'. 'Damping' is '0.05'. 'Use Custom Boundaries' is checked. Below it, 'First Boundary' is '0.5' and 'Second Boundary' is '-0.5'. An eye icon is visible to the right of the 'Use Custom Boundaries' checkbox.

Tip

We can visualize the simulation boundaries positions in the scene view by toggling the 'eye icon' on.



Script Reference

```
waterObject.SimulationModule.IsUsingCustomBoundaries = true;  
waterObject.SimulationModule.FirstCustomBoundary = 0.5f;  
waterObject.SimulationModule.SecondCustomBoundary = -0.5f;
```

On-Collision Ripples Properties

On-Collision Ripples are created when a rigidbody falls into or gets out of the water.

▼ On-Collision Ripples Properties

Collision Properties

Collision Mask	Mixed...
Minimum Depth	-10
Maximum Depth	10
Maximum Distance	0.5

Disturbance Properties

Minimum Disturbance	0.1
Maximum Disturbance	0.75
Minimum Velocity	6.25



The "Minimum Velocity" property controls the minimum velocity that a rigidbody hitting the water should have to cause the "Maximum Disturbance" to the water surface.

▼ ☒ On-Water-Enter Ripples Properties

On Water Enter ()

List is Empty

+

-

▼ ☒ Sound Effect

Audio Clip	None (Audio Clip)
Pool Size	10
Can Expand	<input checked="" type="checkbox"/>
Volume	<input type="range" value="1"/>
Constant Pitch	<input type="checkbox"/>
Minimum Pitch	<input type="range" value="0.75"/>
Maximum Pitch	<input type="range" value="1.25"/>

▼ ☐ Particle Effect

Particle System	None (Particle System)
Pool Size	10
Can Expand	<input checked="" type="checkbox"/>
Spawn Offset	X <input type="text" value="0"/> Y <input type="text" value="0"/> Z <input type="text" value="0"/>

Stop Action ()

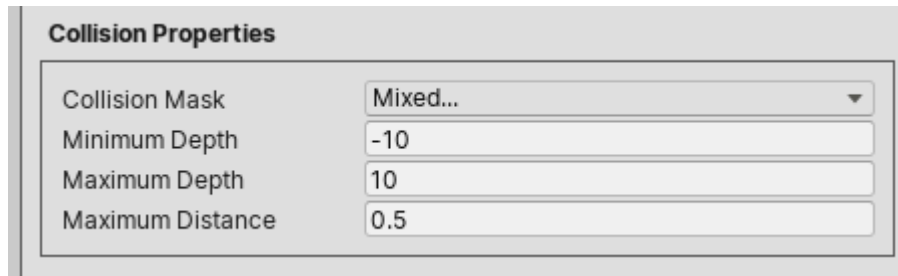
List is Empty

+

-

► ☒ On-Water-Exit Ripples Properties

Collision Properties



The screenshot shows a panel titled "Collision Properties". It contains four settings:

Property	Value
Collision Mask	Mixed...
Minimum Depth	-10
Maximum Depth	10
Maximum Distance	0.5

Under the **Collision Properties**, the **Collision Mask** property controls which objects layers are able to create on-collision ripples.

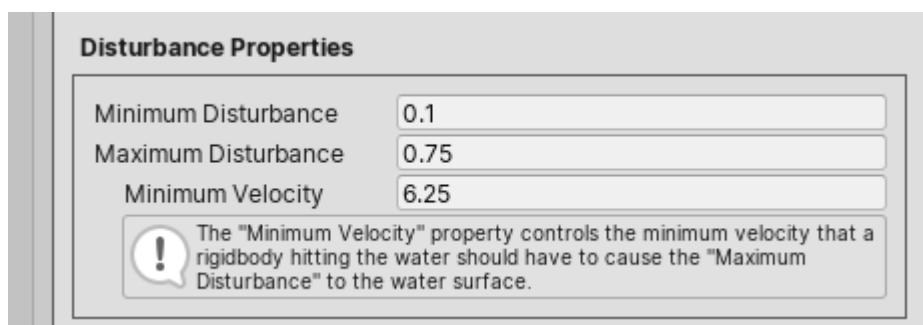
We can also filter the rigidbodies by their depth (z-position), and as such they should have a depth value between the **Minimum Depth** and the **Maximum Depth** properties values to be able to create on-collision ripples.

Lastly, the **Maximum Distance** property controls the maximum distance from the water surface over which to check for collisions.

Script Reference

```
waterObject.OnCollisionRipplesModule.CollisionMask =  
LayerMask.GetMask("Default", "TransparentFX");  
waterObject.OnCollisionRipplesModule.CollisionMinimumDepth = -10f;  
waterObject.OnCollisionRipplesModule.CollisionMaximumDepth = 10f;  
waterObject.OnCollisionRipplesModule.CollisionRaycastMaximumDistance = 0.5f;
```

Disturbance Properties



The screenshot shows a panel titled "Disturbance Properties". It contains three settings:

Property	Value
Minimum Disturbance	0.1
Maximum Disturbance	0.75
Minimum Velocity	6.25

Below the settings is a warning icon and text: "The 'Minimum Velocity' property controls the minimum velocity that a rigidbody hitting the water should have to cause the 'Maximum Disturbance' to the water surface."

The **Minimum Disturbance** and the **Maximum Disturbance** properties control the minimum and the maximum displacement of the water surface, respectively, when a rigidbody gets into or out of water. The *greater* the velocity of the rigidbody, the *greater* the disturbance.

The **Minimum Velocity** property controls the minimum velocity that a rigidbody hitting the water should have to cause the **Maximum Disturbance** to the water surface.

```
waterObject.OnCollisionRipplesModule.MinimumDisturbance = 0.1f;
waterObject.OnCollisionRipplesModule.MaximumDisturbance = 0.6f;
waterObject.OnCollisionRipplesModule.MinimumVelocityToCauseMaximumDisturbance = 5f;
```

On-Water-Enter And On-Water-Exit Ripples Properties

☒ **On-Water-Enter Ripples Properties**

On Water Enter ()

List is Empty

+ -

☒ **Sound Effect**

Audio Clip
None (Audio Clip)

Pool Size
10

Can Expand
☒

Volume
1

Constant Pitch
☐

Minimum Pitch
0.75

Maximum Pitch
1.25

☐ **Particle Effect**

Particle System
None (Particle System)

Pool Size
10

Can Expand
☒

Spawn Offset
X 0 Y 0 Z 0

Stop Action ()

List is Empty

+ -

☒ **On-Water-Exit Ripples Properties**

On Water Exit ()

List is Empty

+ -

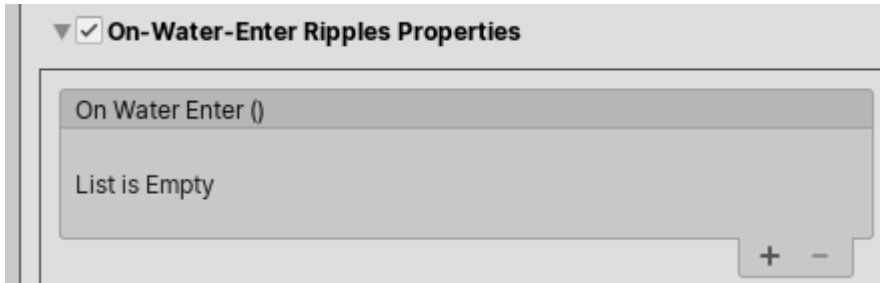
☐ **Sound Effect**

☐ **Particle Effect**

Script Reference

```
waterObject.OnCollisionRipplesModule.IsOnWaterEnterRipplesActive = true;  
waterObject.OnCollisionRipplesModule.IsOnWaterExitRipplesActive = true;
```

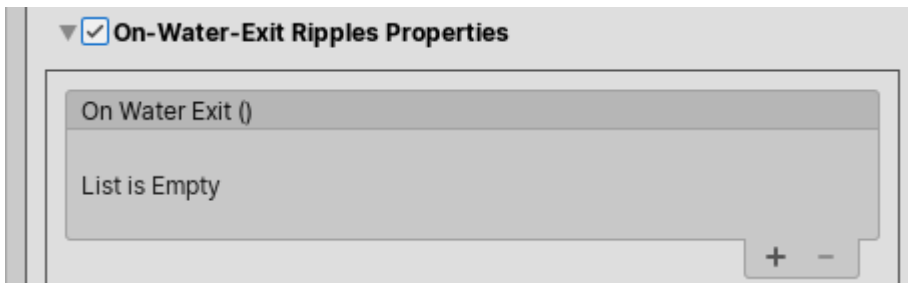
On-Water-Enter And On-Water-Exit Events



The ***OnWaterEnter*** event is triggered when a rigidbody falls into water.

Script Reference

```
waterObject.OnCollisionRipplesModule.OnWaterEnter.AddListener(OnWaterEnterCallb
```

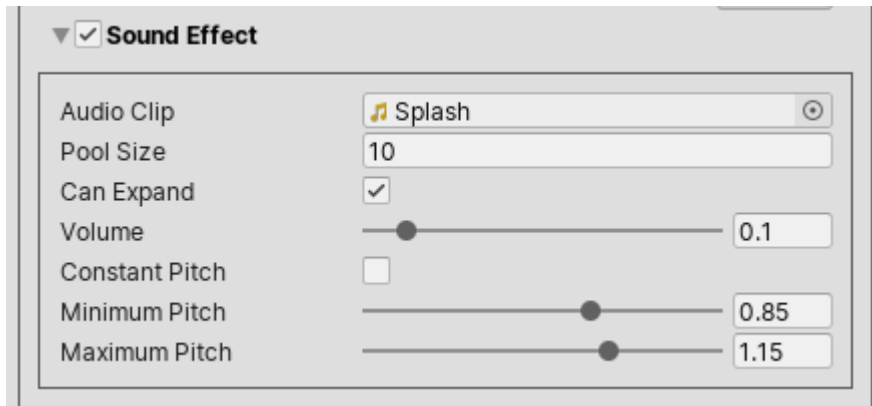


The ***OnWaterExit*** event is triggered when a rigidbody gets out of water.

Script Reference

```
waterObject.OnCollisionRipplesModule.OnWaterExit.AddListener(OnWaterExitCallbac
```

On-Water-Enter And On-Water-Exit Sound Effect



It is possible to specify an **Audio Clip** to play when a rigidbody falls into or gets out of the water.

The **Pool Size** property controls the number of audio sources to pool when the game starts playing.

The **Can Expand** property controls whether or not the number of pooled audio sources can increase at runtime if needed.

The **Volume** property controls the audio clip volume when played.

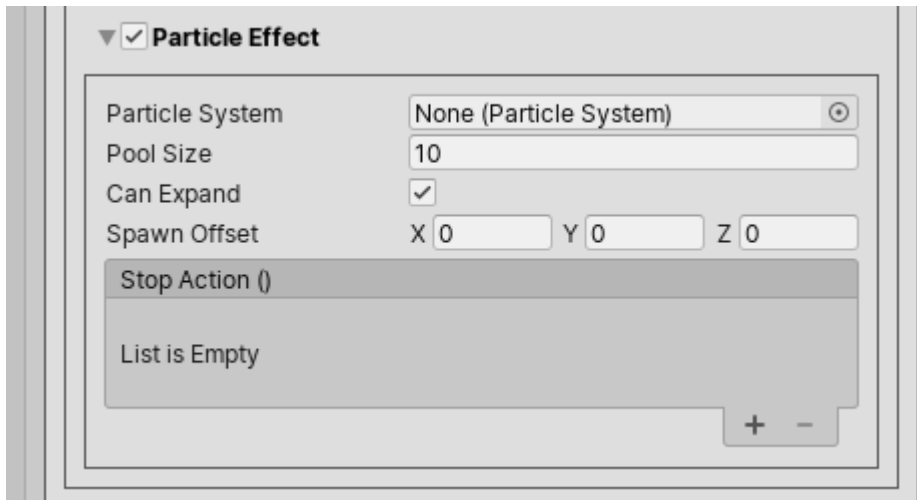
Concerning the audio clip pitch (playback speed), we can use a constant audio pitch by toggling the **Constant Pitch** property on and setting the constant audio clip pitch value.

But, if we would like to vary the audio clip pitch depending on the velocity of the rigidbody hitting the water, we keep the **Constant Pitch** property toggled off and provide a **Minimum Pitch** and a **Maximum Pitch** values instead. And in this case, the *greater* the rigidbody's velocity, the *lower* the pitch value.

```
// On-Water-Enter Sound Effect
waterObject.OnCollisionRipplesModule.OnWaterEnterRipplesSoundEffect.IsActive
= true;
waterObject.OnCollisionRipplesModule.OnWaterEnterRipplesSoundEffect.AudioClip
= audioClip;
waterObject.OnCollisionRipplesModule.OnWaterEnterRipplesSoundEffect.PoolSize
= 10;
waterObject.OnCollisionRipplesModule.OnWaterEnterRipplesSoundEffect.CanExpandPo
= true;
waterObject.OnCollisionRipplesModule.OnWaterEnterRipplesSoundEffect.Volume =
0.1f;
waterObject.OnCollisionRipplesModule.OnWaterEnterRipplesSoundEffect.IsUsingConst
= false;
//
waterObject.OnCollisionRipplesModule.OnWaterEnterRipplesSoundEffect.AudioPitch
= 1f;
waterObject.OnCollisionRipplesModule.OnWaterEnterRipplesSoundEffect.MinimumAud
= 0.85f;
waterObject.OnCollisionRipplesModule.OnWaterEnterRipplesSoundEffect.MaximumAud
= 1.15f;

// On-Water-Exit Sound Effect
waterObject.OnCollisionRipplesModule.OnWaterExitRipplesSoundEffect.IsActive
= true;
waterObject.OnCollisionRipplesModule.OnWaterExitRipplesSoundEffect.AudioClip
= audioClip;
waterObject.OnCollisionRipplesModule.OnWaterExitRipplesSoundEffect.PoolSize
= 10;
waterObject.OnCollisionRipplesModule.OnWaterExitRipplesSoundEffect.CanExpandPoc
= true;
waterObject.OnCollisionRipplesModule.OnWaterExitRipplesSoundEffect.Volume =
0.1f;
waterObject.OnCollisionRipplesModule.OnWaterExitRipplesSoundEffect.IsUsingConst
= false;
//
waterObject.OnCollisionRipplesModule.OnWaterExitRipplesSoundEffect.AudioPitch
= 1f;
waterObject.OnCollisionRipplesModule.OnWaterExitRipplesSoundEffect.MinimumAudio
= 0.85f;
waterObject.OnCollisionRipplesModule.OnWaterExitRipplesSoundEffect.MaximumAudio
= 1.15f;
```

On-Water-Enter And On-Water-Exit Particle Effect



It is possible to specify a **Particle System** to play when a rigidbody falls into or gets out of the water.

The **Pool Size** property controls the number of particle systems to pool when the game starts playing.

The **Can Expand** property controls whether or not the number of pooled particle systems can increase at runtime if needed.

The **Spawn Offset** property controls how much to shift the particle system spawn position.

Lastly, the **StopAction** event is triggered whenever a particle system ends playing.

```
// On-Water-Enter Particle Effect
waterObject.OnCollisionRipplesModule.OnWaterEnterRipplesParticleEffect.IsActive
= true;
waterObject.OnCollisionRipplesModule.OnWaterEnterRipplesParticleEffect.Particle
= particleSystem;
waterObject.OnCollisionRipplesModule.OnWaterEnterRipplesParticleEffect.PoolSize
= 10;
waterObject.OnCollisionRipplesModule.OnWaterEnterRipplesParticleEffect.CanExpan
= true;
waterObject.OnCollisionRipplesModule.OnWaterEnterRipplesParticleEffect.SpawnOff
= new Vector3(0f, -0.55f, 0.03f);
waterObject.OnCollisionRipplesModule.OnWaterEnterRipplesParticleEffect.StopActi

// On-Water-Exit Particle Effect
waterObject.OnCollisionRipplesModule.OnWaterExitRipplesParticleEffect.IsActive
= true;
waterObject.OnCollisionRipplesModule.OnWaterExitRipplesParticleEffect.Particle
= particleSystem;
waterObject.OnCollisionRipplesModule.OnWaterExitRipplesParticleEffect.PoolSize
= 10;
waterObject.OnCollisionRipplesModule.OnWaterExitRipplesParticleEffect.CanExpan
= true;
waterObject.OnCollisionRipplesModule.OnWaterExitRipplesParticleEffect.SpawnOff
= new Vector3(0f, -0.55f, 0.03f);
waterObject.OnCollisionRipplesModule.OnWaterExitRipplesParticleEffect.StopActi
```

Constant Ripples

Constant Ripples are created at regular time intervals.

☒ **Water**

Size X Y

Subdivisions Per Unit

☐ Use Edge Collider

▾ Simulation Properties

Simulation Preview

Buoyancy Effector Properties

Surface Level 0.02

Wave Properties

Stiffness
 Spread
 Damping 0.05
 Use Custom Boundaries ☐

▶ On-Collision Ripples Properties

▾ ☒ Constant Ripples Properties

Continue creating ripples when off-screen ☐

Time Interval Properties

Randomize Time Interval ☐
 Time Interval

Source Properties

Randomize Positions ☒
 Ripples Source Count

Disturbance Properties

Randomize Disturbance ☐
 Disturbance
 Smooth Ripples ☒
 Smoothing Factor 0.125

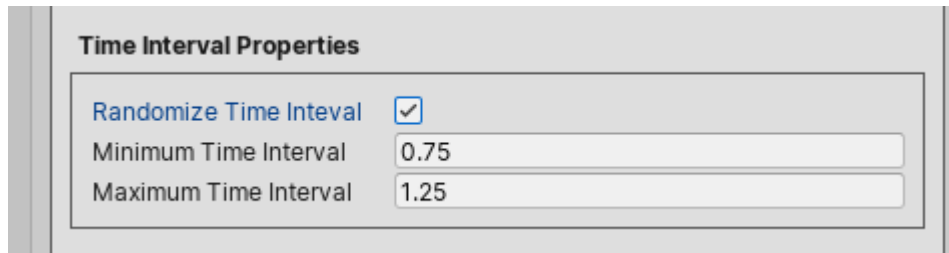
▶ ☐ Sound Effect
 ▶ ☐ Particle Effect

The ***Continue creating ripples when off-screen*** property controls whether or not the water script keeps creating ripples even when the water object is not visible to any camera.

Script Reference

```
waterObject.ConstantRipplesModule.IsActive = true;  
waterObject.ConstantRipplesModule.UpdateWhenOffscreen = false;
```

Constant Ripples Time Interval



The panel titled "Time Interval Properties" contains a checkbox labeled "Randomize Time Interval" which is checked. Below it are two input fields: "Minimum Time Interval" with the value "0.75" and "Maximum Time Interval" with the value "1.25".

We could specify a fixed time interval, or just provide the minimum and the maximum time intervals and let the water script pick a random time interval each time ripples are created.

Script Reference

```
waterObject.ConstantRipplesModule.RandomizeTimeInterval = true;  
//waterObject.ConstantRipplesModule.TimeInterval = 0.5f;  
waterObject.ConstantRipplesModule.MinimumTimeInterval = 0.5f;  
waterObject.ConstantRipplesModule.MaximumTimeInterval = 1f;
```

Source Properties

A constant ripple originates from the disturbance of a surface vertex, which we will refer to as the ripple source.

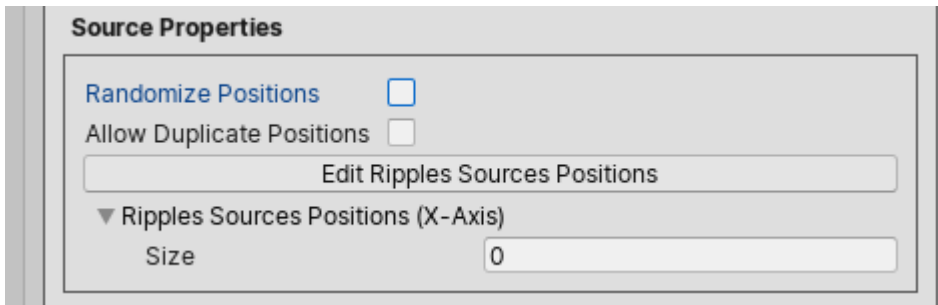


The panel titled "Source Properties" contains a checkbox labeled "Randomize Positions" which is checked. Below it is an input field labeled "Ripples Source Count" with the value "8".

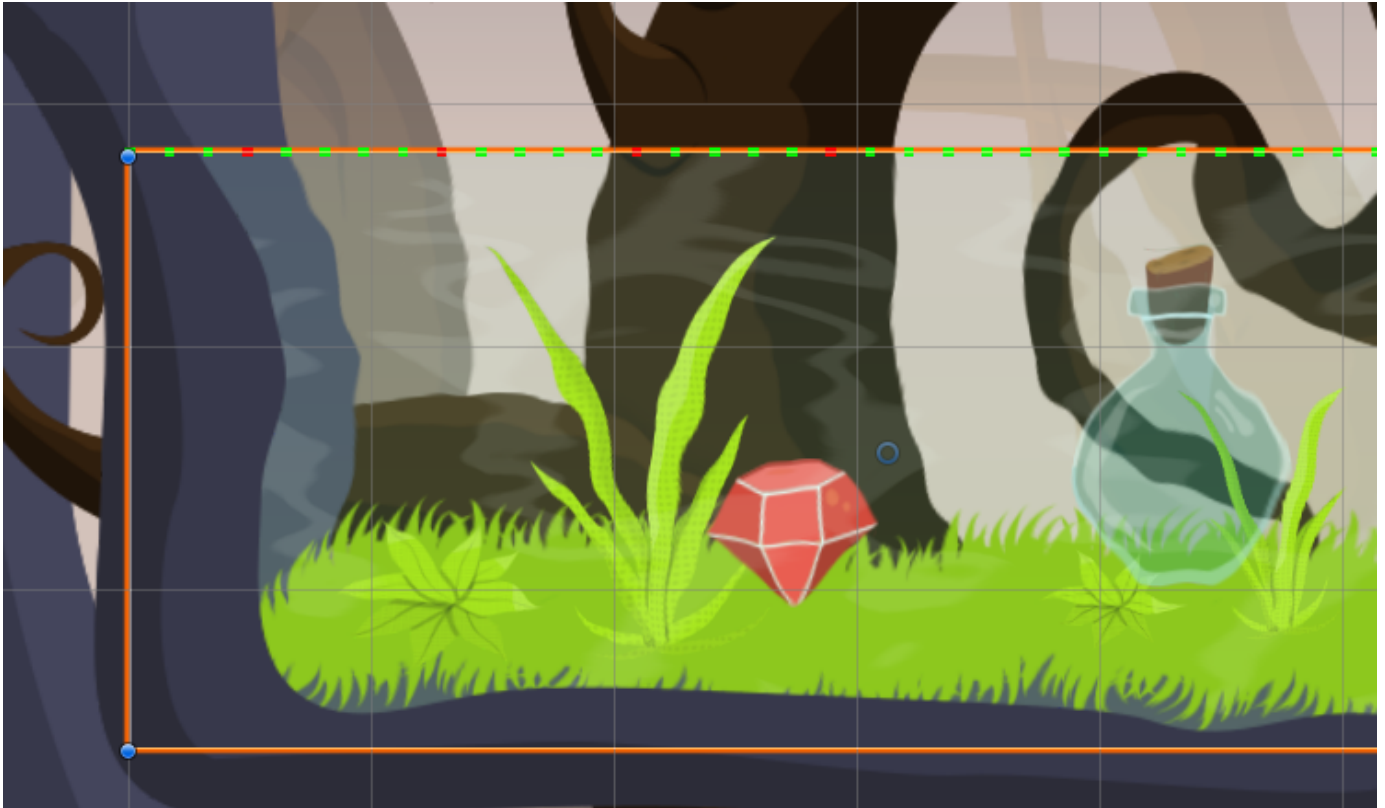
If the **Randomize Source** property is toggled on, the water script will randomly pick "**Ripples Source Count**" surface vertices, and disturb them.

Script Reference

```
waterObject.ConstantRipplesModule.RandomizeRipplesSourcePositions = true;  
waterObject.ConstantRipplesModule.RandomRipplesSourceCount = 8;
```



But if the **Randomize Source** property is left toggled off, we select the surface vertices ourselves in the scene-view.



Info

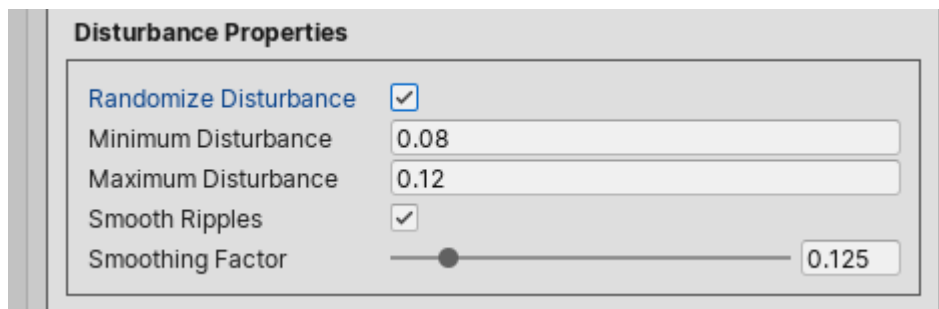
The green and the red dots in the scene-view represent the water mesh surface vertices. Clicking the green dot adds the surface vertex to the ripples source positions list, and clicking the red dot removes it from the list.

The **Allow Duplicate Positions** property controls whether or not to disturb a surface vertex, that is present in ripples source positions list multiple times, more than once.

Script Reference

```
waterObject.ConstantRipplesModule.RandomizeRipplesSourcePositions = false;  
waterObject.ConstantRipplesModule.SourcePositions.Add(new Vector3(-5.35f,  
0.87f, 0f));  
waterObject.ConstantRipplesModule.AllowDuplicateRipplesSourcePositions =  
false;
```

Constant Ripples Disturbance Properties



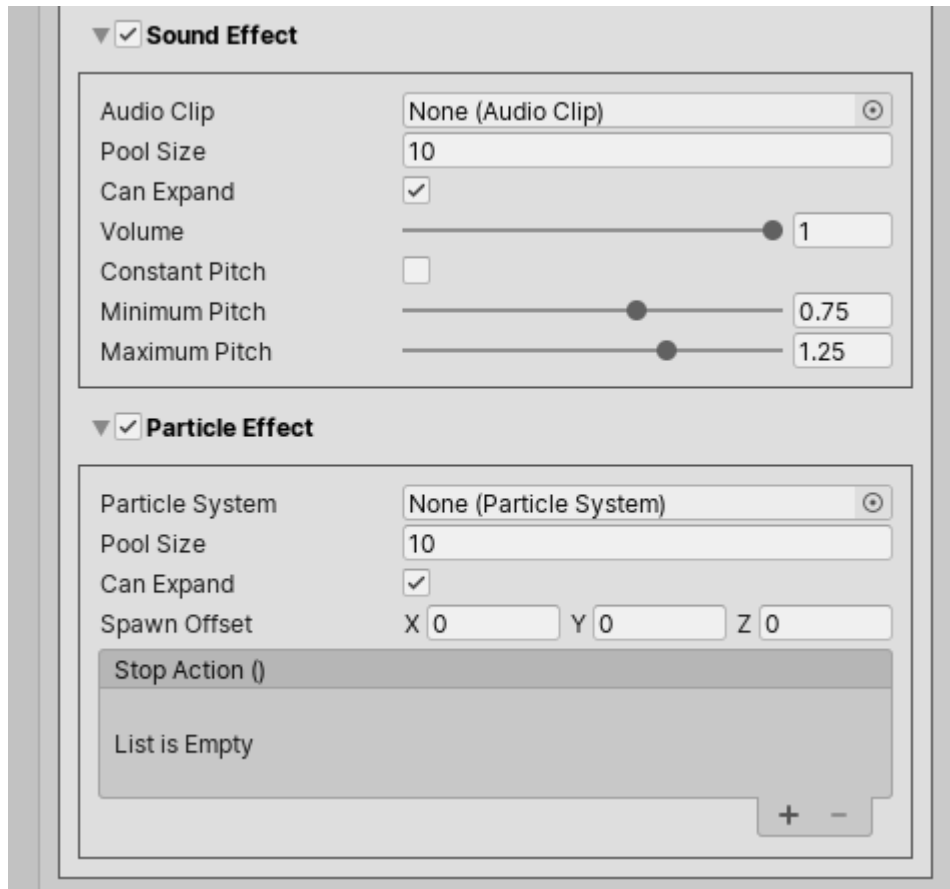
We could specify a fixed disturbance value, or just provide the minimum and the maximum disturbance values and let the water script pick a random disturbance value each time ripples are created.

If the **Smooth Ripples** is toggled on, the ripple source as well as its neighbor surface vertices are disturbed to form a *smoother* ripple. The **Smoothing Factor** property controls the amount of disturbance to apply to neighbor vertices, and then how smooth the created ripple looks.

Script Reference

```
waterObject.ConstantRipplesModule.RandomizeDisturbance = true;  
//waterObject.ConstantRipplesModule.Disturbance = 0.10f;  
waterObject.ConstantRipplesModule.MinimumDisturbance = 0.08f;  
waterObject.ConstantRipplesModule.MaximumDisturbance = 0.12f;  
waterObject.ConstantRipplesModule.SmoothRipples = true;  
waterObject.ConstantRipplesModule.SmoothingFactor = 0.125f;
```

Constant Ripples Sound And Particle Effects



As was the case with on-collision ripples, we could activate a sound and a particle effects to play whenever a constant ripple is created.

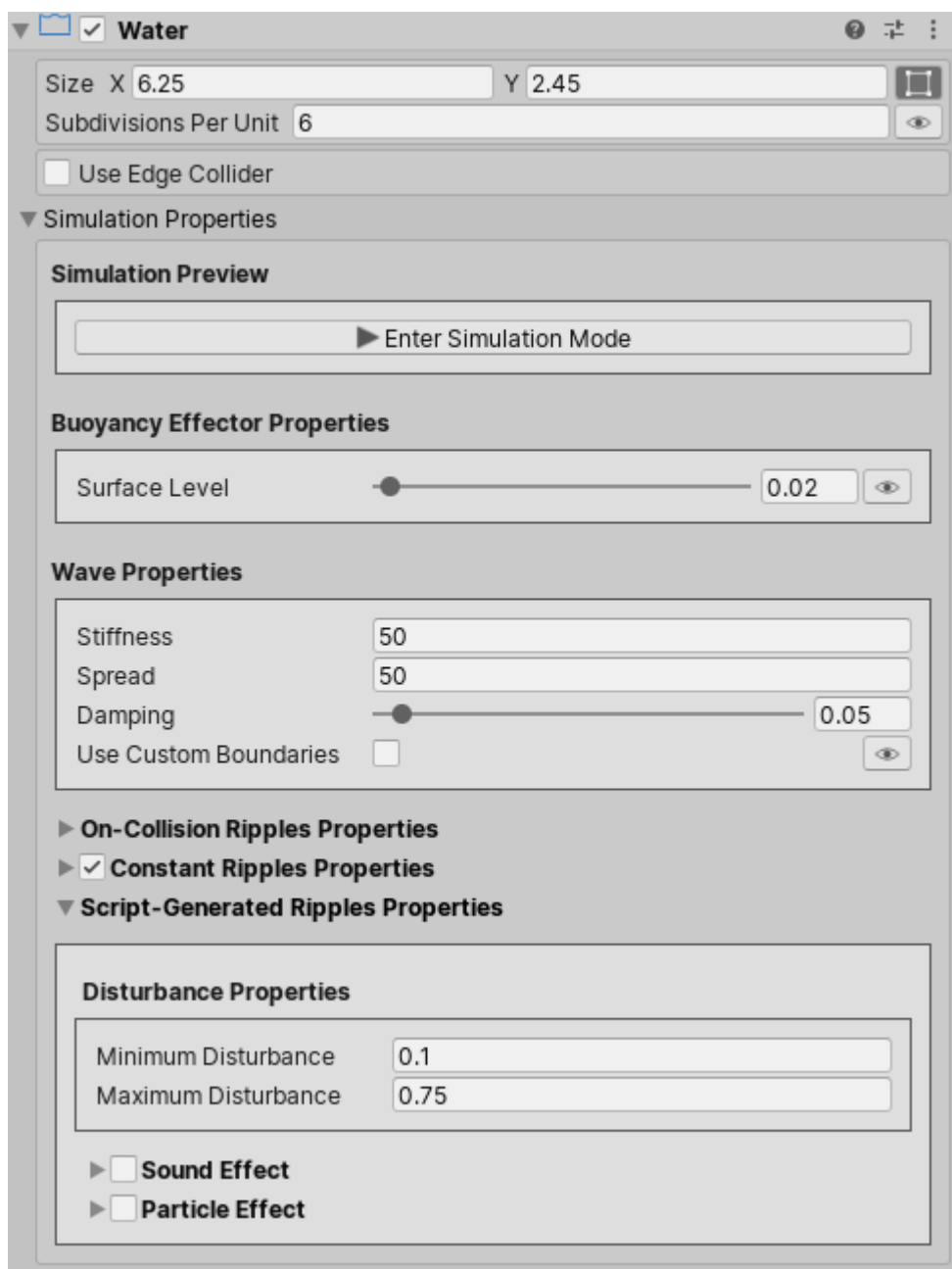
The description of the constant ripples sound and particle effects properties is exactly the same as the on-collision ripples, as discussed [here](#).

Script Reference - Sound Effect Properties

```
waterObject.ConstantRipplesModule.SoundEffect.IsActive = true;
waterObject.ConstantRipplesModule.SoundEffect.AudioClip = audioClip;
waterObject.ConstantRipplesModule.SoundEffect.PoolSize = 10;
waterObject.ConstantRipplesModule.SoundEffect.CanExpandPool = false;
waterObject.ConstantRipplesModule.SoundEffect.Volume = 1f;
waterObject.ConstantRipplesModule.SoundEffect.IsUsingConstantAudioPitch =
false;
//waterObject.ConstantRipplesModule.SoundEffect.AudioPitch = 1f;
waterObject.ConstantRipplesModule.SoundEffect.MinimumAudioPitch = 0.75f;
waterObject.ConstantRipplesModule.SoundEffect.MaximumAudioPitch = 1.25f;
```

```
waterObject.ConstantRipplesModule.ParticleEffect.IsActive = true;
waterObject.ConstantRipplesModule.ParticleEffect.ParticleSystem =
particleSystem;
waterObject.ConstantRipplesModule.ParticleEffect.PoolSize = 10;
waterObject.ConstantRipplesModule.ParticleEffect.CanExpandPool = false;
waterObject.ConstantRipplesModule.ParticleEffect.SpawnOffset = Vector3.zero;
waterObject.ConstantRipplesModule.ParticleEffect.StopAction.AddListener(OnPart:
```

Script-Generated Ripples



Script-Generated ripples are created in code by calling the `GenerateRipple()` method.

Example

```
namespace Game2DWaterKit.DemoTutorial
{
    using UnityEngine;

    public class RipplesGenerator : MonoBehaviour
    {
        private Camera _mainCamera;

        public Game2DWater waterObject;

        [Range(0f,1f)] public float disturbanceFactor = 0.5f;
        public bool pullWaterDown = true;
        public bool playSoundEffect = false;
        public bool playParticleEffect = false;
        public bool smoothRipple = true;
        [Range(0f,1f)] public float smoothingFactor = 0.5f;

        private void Awake()
        {
            _mainCamera = Camera.main;
        }

        private void Update()
        {
            if (Input.GetMouseButtonDown(0) && waterObject != null)
            {
                Vector2 position =
                    _mainCamera.ScreenToWorldPoint(Input.mousePosition);

                // Create ripple

                waterObject.ScriptGeneratedRipplesModule.GenerateRipple(position,
                    disturbanceFactor, pullWaterDown, playSoundEffect, playParticleEffect,
                    smoothRipple, smoothingFactor);
            }
        }
    }
}
```

The GenerateRipple() method takes 7 parameters:

- **Position:** [Vector2] controls the position where we would like to create the ripple. The nearest surface vertex to this position is disturbed.
- **Disturbance Factor:** [Float, Range: 0..1] controls the amount of disturbance to apply to the water surface. The actual applied disturbance is computed by interpolating the **Minimum Disturbance** and the **Maximum Disturbance** by this factor.

Info

The **Minimum Disturbance** and the **Maximum Disturbance** are set in the inspector. We'll see them shortly.

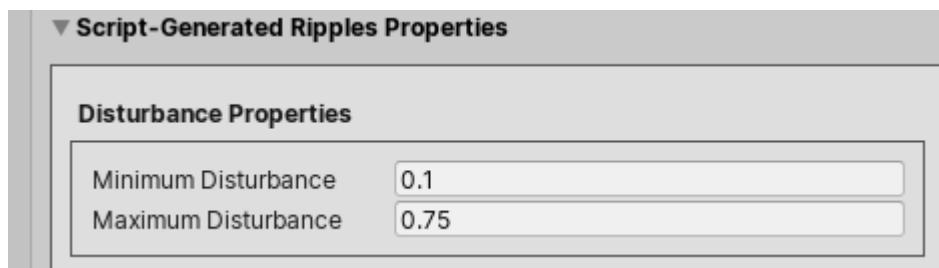
- **PullWaterDown:** [Bool] controls whether the water surface is pulled down or pushed up when creating the ripple. This mimics an object falling into water or getting out of water, respectively.
- **PlaySoundEffect:** [Bool] controls whether or not to play the sound effect.
- **PlayParticleEffect:** [Bool] controls whether or not to play the particle effect.

Info

The sound and the particle effects properties are set in the inspector. We'll see them shortly.

- **SmoothRipple:** [Bool] controls whether or not to disturb neighbor surface vertices in order to create a smoother ripple.
- **Smoothing Factor:** [Float, range: 0..1] controls the amount of disturbance to apply to neighbor vertices.

Script-Generated Ripples Disturbance Properties



We set the **Minimum Disturbance** and the **Maximum Disturbance** properties in the inspector.

Script Reference

```
waterObject.ScriptGeneratedRipplesModule.MinimumDisturbance = 0.1f;  
waterObject.ScriptGeneratedRipplesModule.MaximumDisturbance = 0.75f;
```

Script-Generated Ripples Sound And Particle Effects Properties

▼ Script-Generated Ripples Properties

Disturbance Properties

Minimum Disturbance

0.1

Maximum Disturbance

0.75

▼ ☒ Sound Effect

Audio Clip

None (Audio Clip)

Pool Size

10

Can Expand

☒

Volume

1

Constant Pitch

☐

Minimum Pitch

0.75

Maximum Pitch

1.25

▼ ☒ Particle Effect

Particle System

None (Particle System)

Pool Size

10

Can Expand

☒

Spawn Offset

X 0 Y 0 Z 0

Stop Action ()

List is Empty

+

-

The description of the script-generated ripples sound and particle effects properties is exactly the same as the on-collision ripples, as discussed [here](#).

Script Reference - Sound Effect Properties

```
waterObject.ScriptGeneratedRipplesModule.SoundEffect.IsActive = true;
waterObject.ScriptGeneratedRipplesModule.SoundEffect.AudioClip = audioClip;
waterObject.ScriptGeneratedRipplesModule.SoundEffect.PoolSize = 10;
waterObject.ScriptGeneratedRipplesModule.SoundEffect.CanExpandPool = false;
waterObject.ScriptGeneratedRipplesModule.SoundEffect.Volume = 1f;
waterObject.ScriptGeneratedRipplesModule.SoundEffect.IsUsingConstantAudioPitch
= false;
//waterObject.ScriptGeneratedRipplesModule.SoundEffect.AudioPitch = 1f;
waterObject.ScriptGeneratedRipplesModule.SoundEffect.MinimumAudioPitch =
0.75f;
waterObject.ScriptGeneratedRipplesModule.SoundEffect.MaximumAudioPitch =
1.25f;
```

Script Reference - Particle Effect Properties

```
waterObject.ScriptGeneratedRipplesModule.ParticleEffect.IsActive = true;
waterObject.ScriptGeneratedRipplesModule.ParticleEffect.ParticleSystem =
particleSystem;
waterObject.ScriptGeneratedRipplesModule.ParticleEffect.PoolSize = 10;
waterObject.ScriptGeneratedRipplesModule.ParticleEffect.CanExpandPool =
false;
waterObject.ScriptGeneratedRipplesModule.ParticleEffect.SpawnOffset =
Vector3.zero;
waterObject.ScriptGeneratedRipplesModule.ParticleEffect.StopAction.AddListener()
```

Water Size Animation

It's possible to animate the water size in code using the method `AnimateWaterSize(targetSize, duration, constraint, wrapMode)`.

- **targetSize:** [Vector2] sets the water target size.
- **duration:** [float] sets the animation duration in seconds.
- **constraint:** [enum WaterAnimationConstraint] constraints the position of one/multiple water edge(s)

WaterAnimationConstraint enum

None, Top, Bottom, Left, Right, TopLeft, TopRight, BottomLeft, BottomRight

- **wrapMode:** [enum WaterAnimationWrapMode]

WaterAnimationWrapMode enum

- **Once:** Stops playing the animation once the target size is reached.
- **Loop:** The water size is reset to the initial size and the animation is restarted once the target size is reached.
- **PingPong:** When the target size is reached, the initial size is set to the new target size and the animation restarts. So the water size will *ping-pong* back and forth between the initial size and the target size.

Example

```
using Game2DWaterKit.Animation;

....

var targetSize = waterObject.MainModule.WaterSize + new Vector2(0f, 3f); //
increase the water height by 3 units
var duration = 2f; // 2 seconds
var constraint = WaterAnimationConstraint.Bottom; // constraints the
position of the bottom edges, so only the top edge "moves"
var wrapMode = WaterAnimationWrapMode.Once; // play the animation once

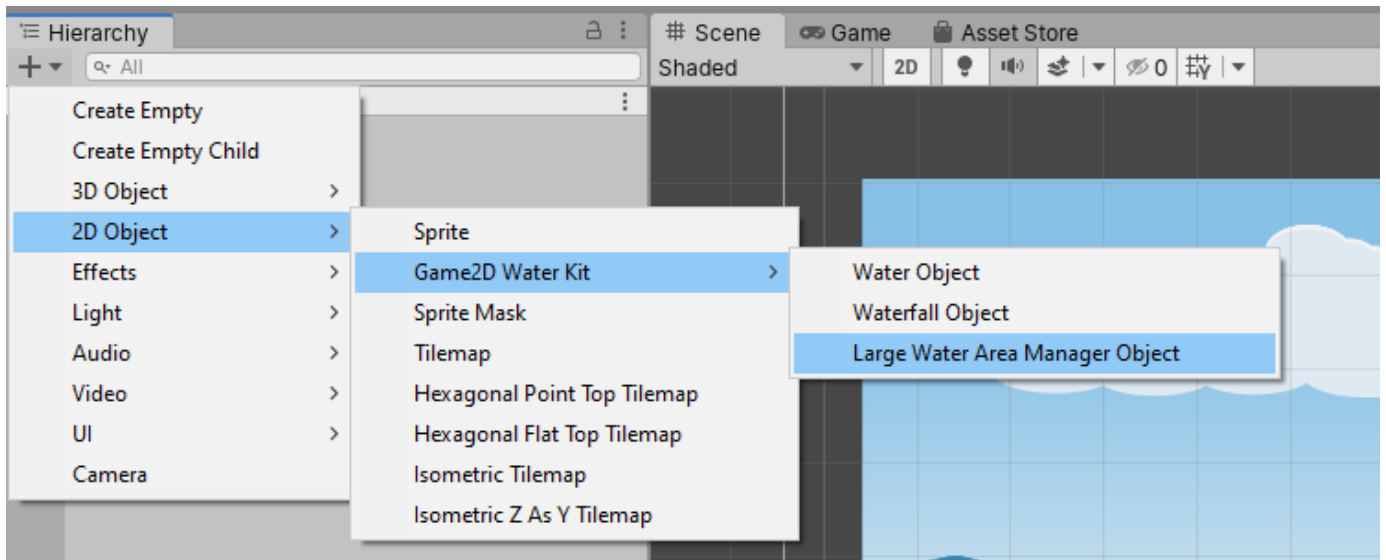
waterObject.AnimationModule.AnimateWaterSize(targetSize, duration,
constraint, wrapMode);
```

Large Water Area Manager

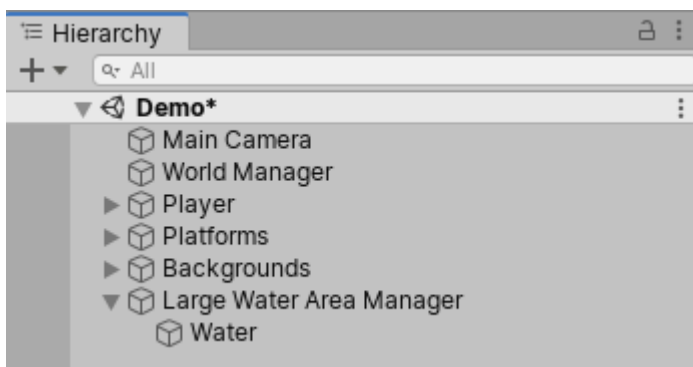
Creating Large Water Area Manager

We create a large water area manager object from the Hierarchy's Create menu:

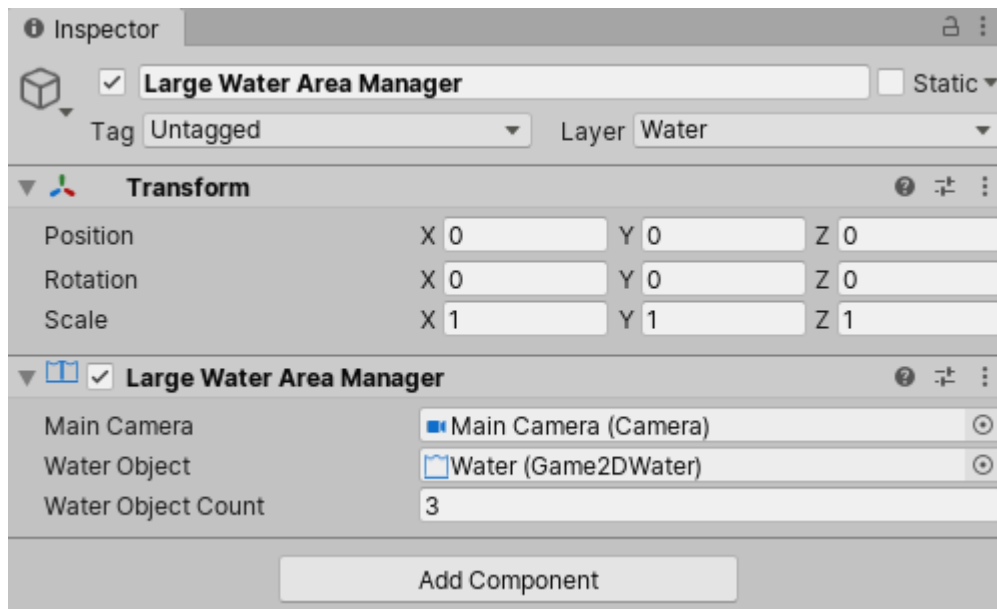
2D Object → Game2D Water Kit → Large Water Area Manager Object



A new large water area manager object is added in the Hierarchy, with a child water object.



Large Water Area Manager Component Properties



Main Camera

The **Main Camera** property sets the scene main camera that will be used to determine the visibility of each spawned water object. So when a water object is no longer visible to this camera, it gets properly repositioned.

Water Object

The **Water Object** property sets the base object from which to instantiate the other water objects. It is assigned by default to the child water object.

Important

The water object should have a width that is at least half of the **Main Camera** view frustum width.

Note

The water object doesn't necessarily have to be a child of the large water area manager object.

Water Object Count

The **Water Object Count** property sets the number of water objects to spawn when the game starts.

Script Reference

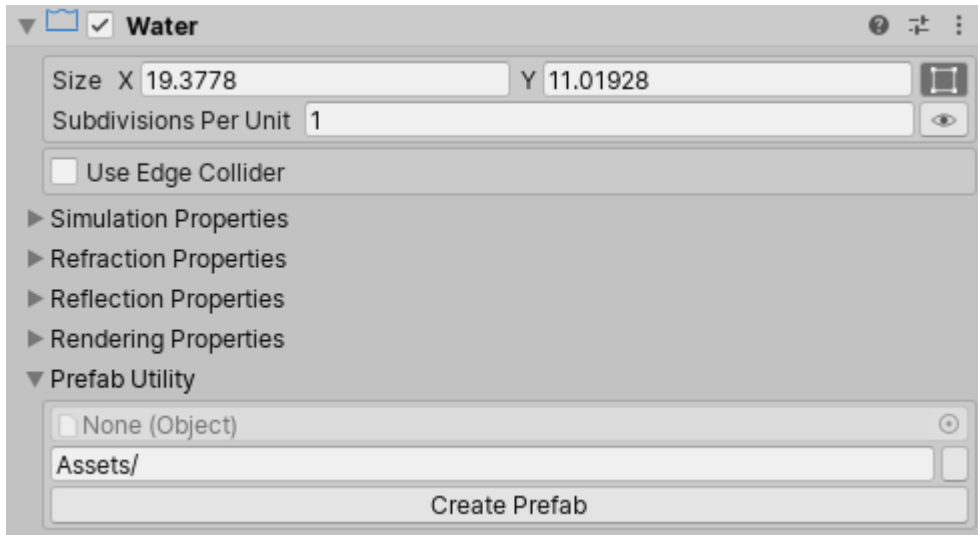
```
largeWaterAreaManager.MainCamera = mainCamera;  
largeWaterAreaManager.WaterObject = waterObject;  
largeWaterAreaManager.WaterObjectCount = 3;
```

Tip

We can get the water object that is located at a specific position using the function `GetWaterObjectLocatedAt(float xWorldSpacePosition)`. This function returns *null* if there's currently no water object in that position

```
Game2DWater waterObject = largeWaterAreaManager.GetWaterObjectLocatedAt(5f);
```

Prefab Utility



The prefab utility serves to save the water object as a prefab properly, along with its material and its generated noise texture.

We only need to choose where we would like to save the prefab, and then hit the **Create Prefab** button!