

Water2D Tool v1.x

Introduction

Water2D Tool is a tool for Unity3D that allows you to quickly create 2D and 2.5D water with dynamic properties for your 2D or 2.5D games.

Water Creation Process

To create a new water object you can access the menu (GameObject->2D Water) where you can choose between a water with a 2D or a 3D collider, Figure 1, or use the shortcuts (CTRL+W, CTRL+SHIFT+W).

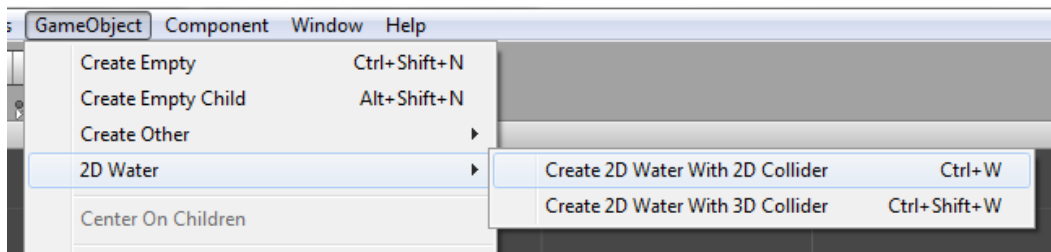


Figure 1 - Water2D menu.

After the water is created you can use the 4 handles to change its size and the position of its 4 edges, Figure 2.

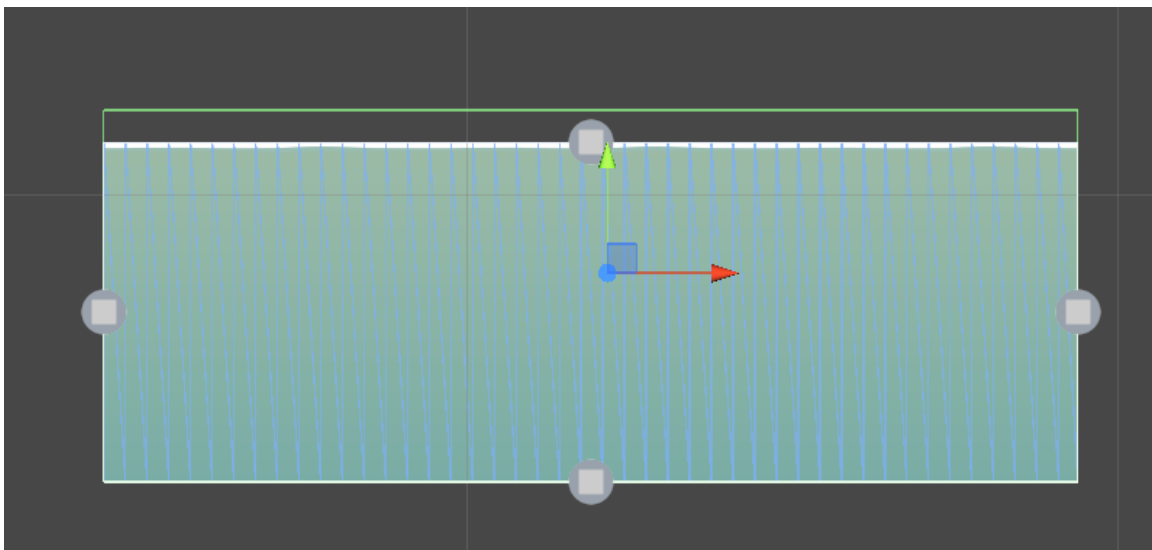


Figure 2 – Water size manipulation.

Springs and water mesh

Water2D Tool creates the waves, by modeling the surface of the water as a series of vertical springs, **Figure 3**. Each surface vertex has a spring attached to it that controls its position.

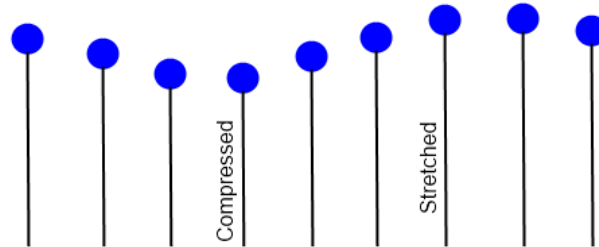


Figure 3 - Springs

Figure 4 shows how the mesh vertex indices are arranged. For simplicity the first half of the water mesh vertices form the bottom of the water mesh while the second half form the top of the water mesh.

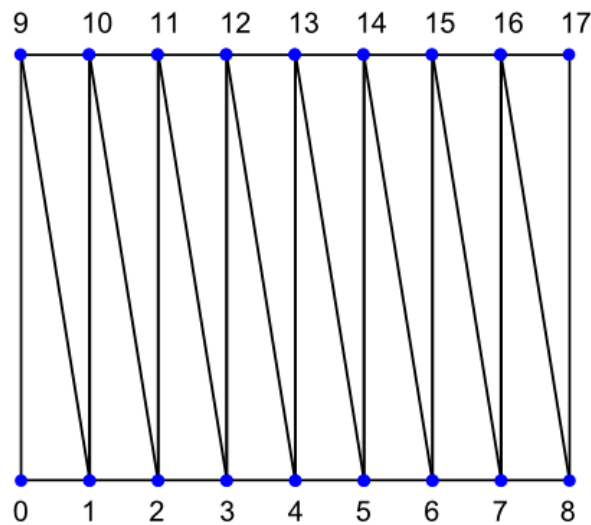


Figure 4 – The water mesh

Vertical Tiling

When enabled, if the water height is greater than the max water height that can be created with the current *Pixels To Units* value, the texture will be tiled vertically, **Figure 5**. Here the green and yellow rectangles are a single 512x512px texture.

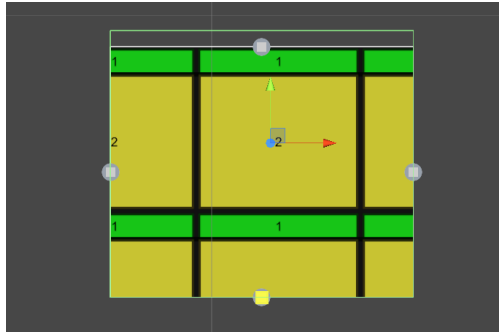


Figure 5 – Vertical Tiling Enabled.

When disabled the texture will be stretched if the water height is greater than the max water height that can be created with the current *Pixels To Units* value, **Figure 6.**

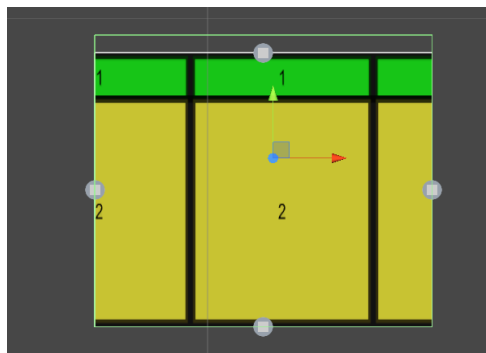


Figure 6 – Vertical Tiling Disabled.

Segments to Units

The density of the mesh vertices on the X axis can be controlled using the value of the field *Segments to Units* from the *Visual Properties* group. The value of 3 means that in 1 m of Unity space on the X axis, 3 columns (or horizontal segments) will be placed. The triangles marked in green form a column, **Figure 7.**

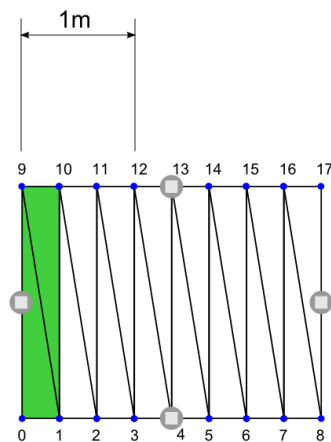


Figure 7 – Segments to Units

2.5D Water

When this toggle is enabled, a horizontal mesh will be added to the vertical mesh, **Figure 8**.

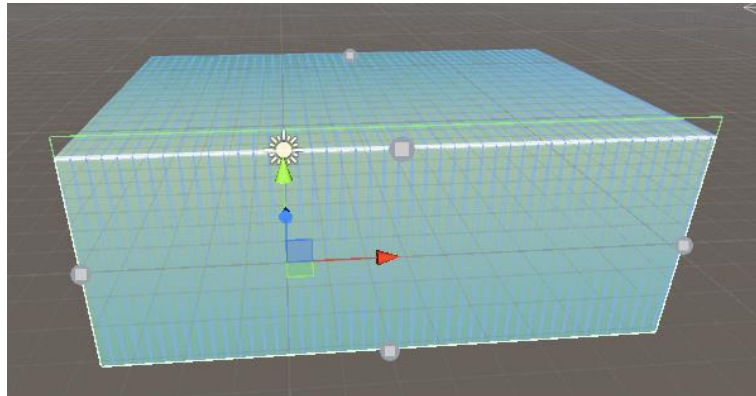


Figure 8 – 2.5D Water

If you look at the water Mesh Renderer, you can see that the water has 2 materials now. The first material is for the vertical mesh, while the second for the horizontal mesh, **Figure 9-11**.

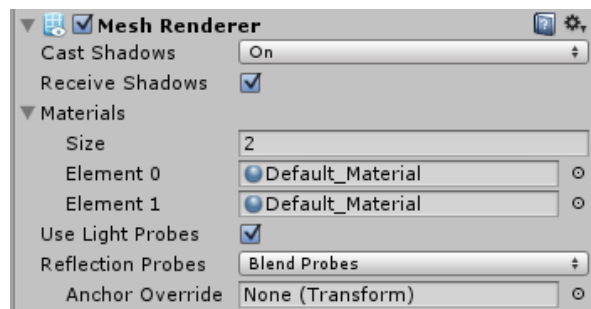


Figure 9 – 2.5D water materials

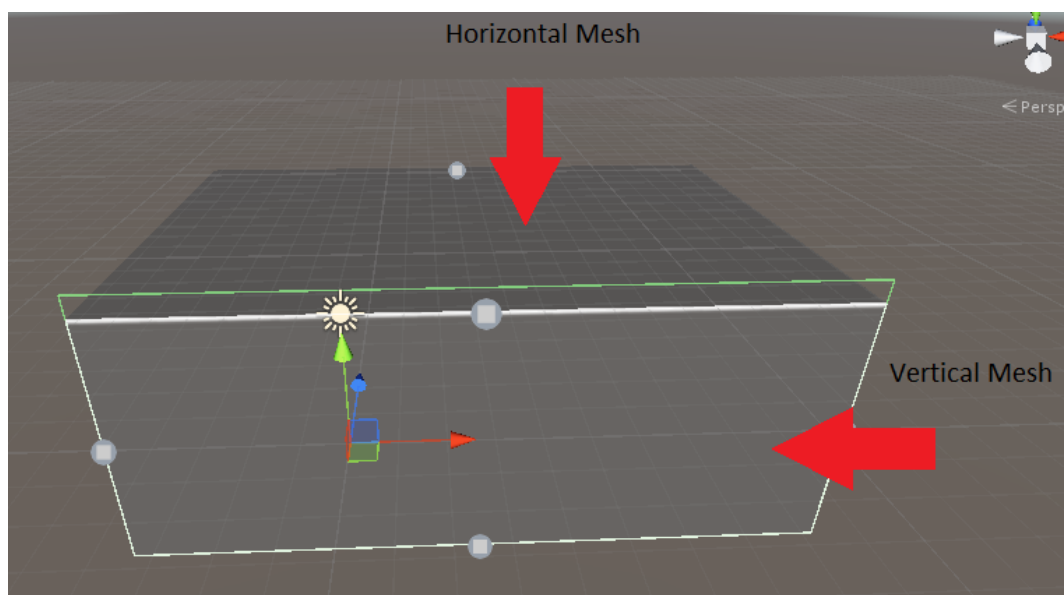


Figure 10 – 2.5D water meshes



Figure 11 – 2.5D water materials

Buoyancy

Polygon Clipping

This dropdown menu contains 2 options that determine which method will be used to calculate the intersection between the object collider and water collider, **Figure 12**.

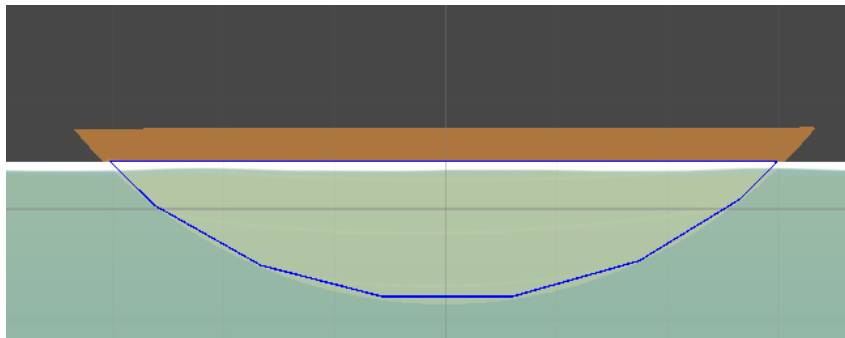


Figure 12 – The intersection polygon, simple clipping

The intersection polygon is represented by the blue lines.

The process of calculating the intersection between 2 polygons is usually called polygon clipping, where one polygon is the subject polygon and the other is the clipping polygon. In our case the collider of the object is the subject polygon.

Simple option – The *simple* clipping option uses the Sutherland Hodgman polygon clipping algorithm and the clipping polygon is always a horizontal line made of 2 points. The position on the X axis of this 2 points corresponds to the global positions of the water edges on the X axis, while the position on the Y axis is equal the global position of the vertex that is closest to the center of the object.

The *simple clipping* is the cheapest option in terms of performance, but it has a disadvantage. Because the clipping polygon is always a horizontal line, the objects behavior on big waves is not very real, **Figure 13**.

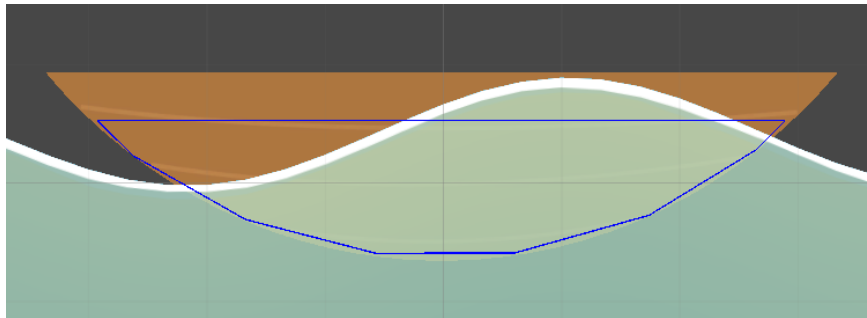


Figure 13 - The intersection polygon, simple clipping

This problem is not present in the complex option.

Complex option – The complex clipping option uses the Clipper library. Here the clipping polygon has at list 4 edges. The clipping polygon can be seen in **Figure 14**, represented by the green lines.

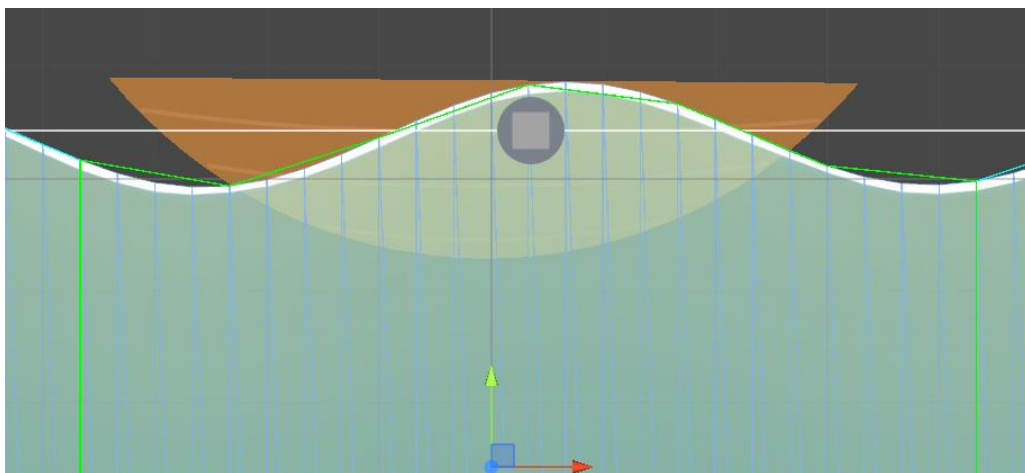


Figure 14 - Complex polygon clipping

Water Line Segments

To get a clipping polygon that has a shape that is close to that of the water wave, but still get a good performance, only the positions of certain vertices are used.

Which vertices are used is determined by the value of the field ***Water Line Segments*** from the ***Buoyancy*** group in the inspector.

As an example let's say the water line left most vertex has the index 9 and the value of the field **Water Line Segments** is 4, **Figure 15**. In this case the positions of the vertices with the indices 9, 13 and 17 will be used to build the clipping polygon. As you can see the value of the field **Water Line Segments** is equal to the number of horizontal segments between the vertices that are used.

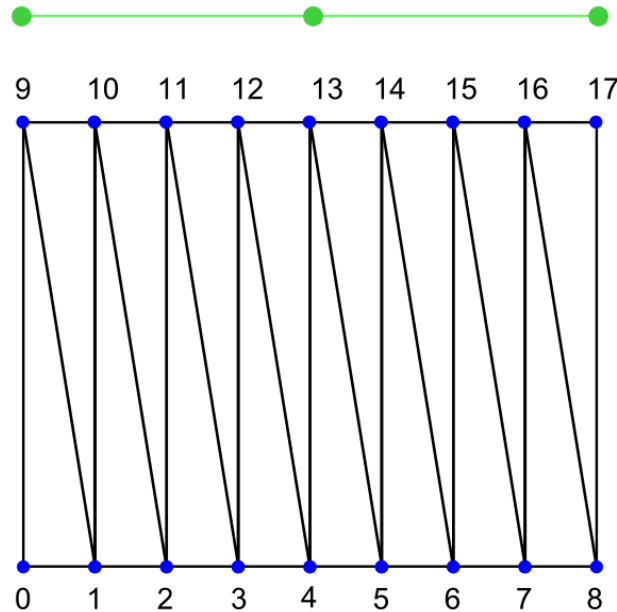


Figure 15 – The water mesh

Polygon Vertices

For objects that have a CircleCollider2D or a SphereCollider, an imaginary regular polygon is created based on the radius of the CircleCollider2D or SphereCollider. This is done so that a drag and a lift can be applied to this object, **Figure 16**. The value of this field determines the number of corners this polygon will have.

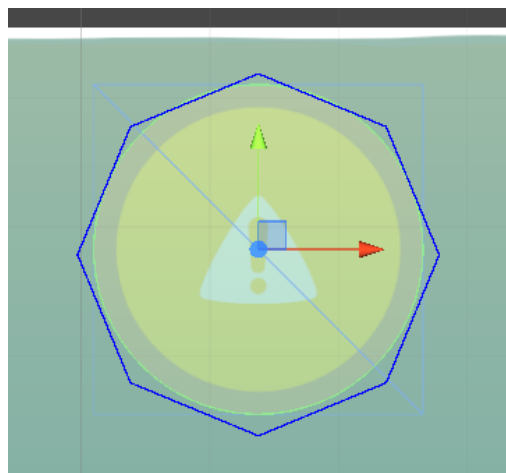


Figure 16 – An imaginary regular polygon for an object with a CircleCollider2D

Float Height

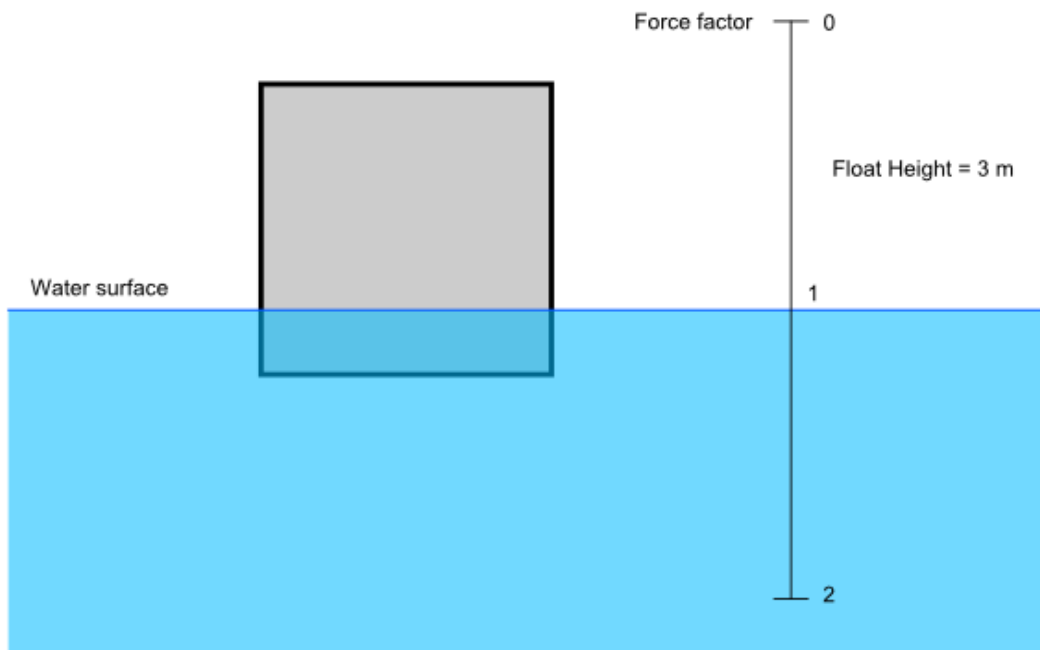


Figure 17 – Linear buoyancy, Float Height

Float Height determines how much force should be applied to an object submerged in the water. A value of 3 means that 3 m under the water the force applied to an object will be 2 times greater than the force applied at the surface of the water, **Figure 17**.

Surface Waves

This are the waves that are not created by objects.

Random Splashes

When this option is selected, the velocity of random surface vertices will be changed based on a random velocity value generated between the values of **Max Velocity** and **Min Velocity**. The velocity of its left and right neighbors are changed too.

This option can be used to simulate water rain.

Sine Waves

When **Sine Waves** option is selected in the **Surface Waves** drop down menu, the velocity of the surface vertices will be changed using a sine function.

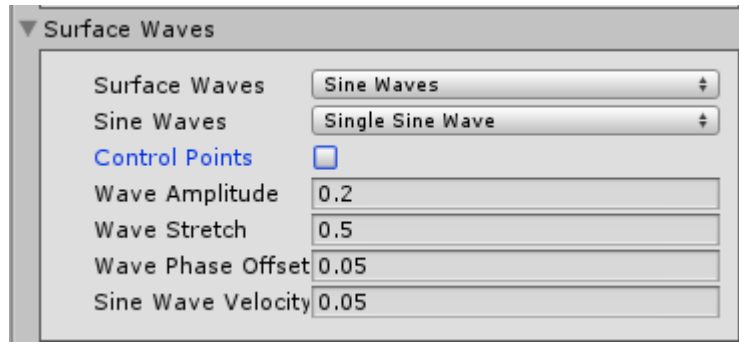


Figure 18 – Surface Waves group

You can use a single sine wave to change the velocity of the surface vertices by selecting *Single Sine Wave* from the *Sine Waves* drop down menu, **Figure 148** or multiple waves by selecting the option *Multiple Sine Waves*, **Figure 19**.

When Single Sine Wave option is selected you can change dynamically the *Wave Amplitude*, *Wave Stretch* and *Wave Phase Offset* of the sine wave by changing the values of the public variables: `waveAmplitude`, `waveStretch` and `wavePhaseOffset`, using the animator or a scrip.

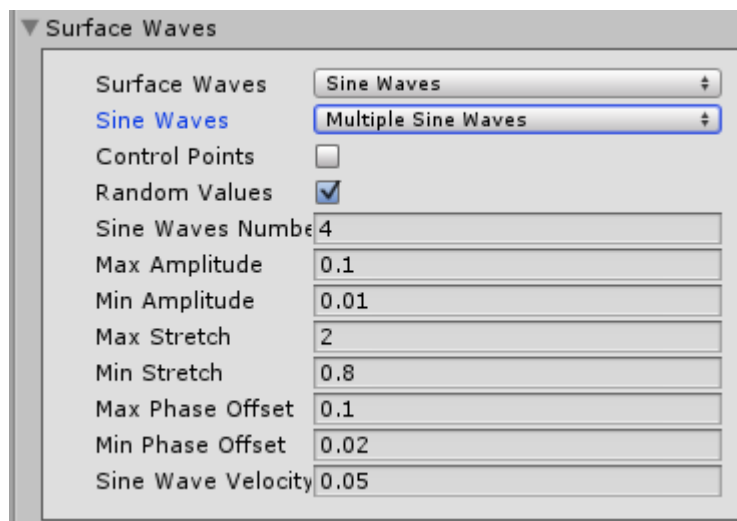


Figure 19 - Surface Waves group

Control Points Toggle

When this toggle is enabled the velocity of only a single vertex will be changed by the Sine function. The velocity of the first water line vertex or the last is changed depending on the option selected in the *Control Point Position* drop down menu, **Figure 20**.

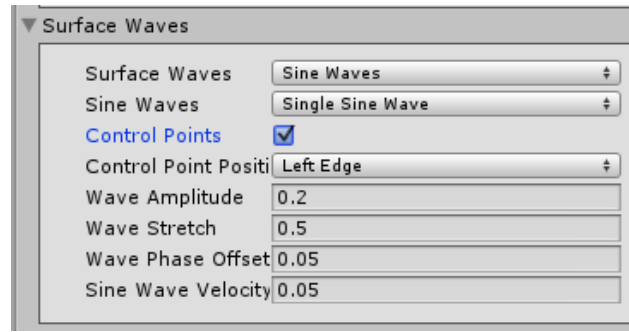


Figure 20 - Surface Waves group

The effect of using the Control Points toggle can be seen in **Figure 21**.

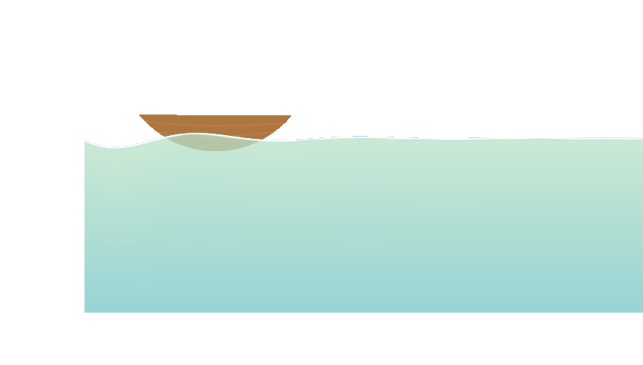


Figure 21 – Only the velocity of the first water line vertex is changed

Random Values

When this toggle is enabled, the amplitude, stretch and phase offset will be random values generated in the start method, for each sine wave.

You can disable this option and use your own values, **Figure 22**.

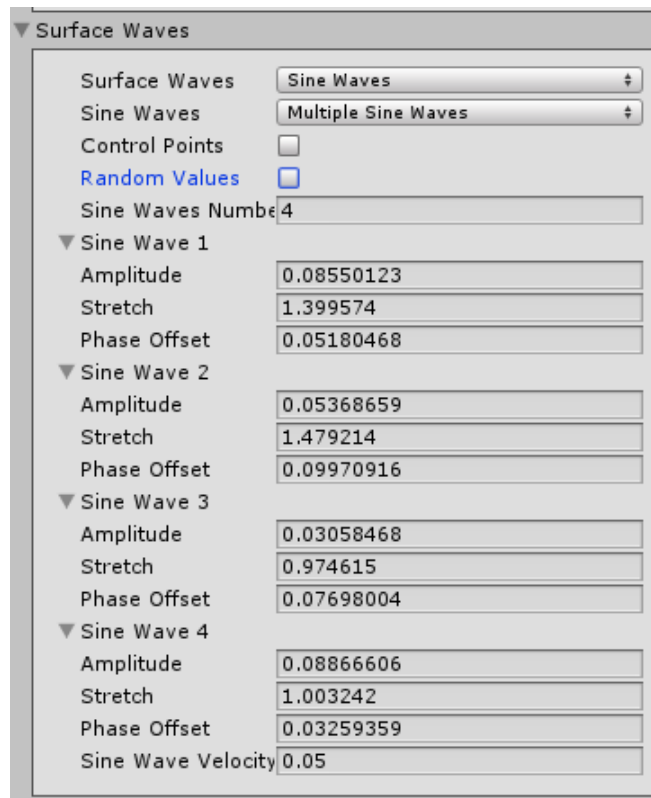


Figure 22 - Surface Waves group

Surface Waves, Control Point option

When this option is selected, the velocity of the first or last water line vertex will be changed using the public variable `controlPointVelocity`, the value of which you can change using the animator or a script.

Miscellaneous

Interaction Region

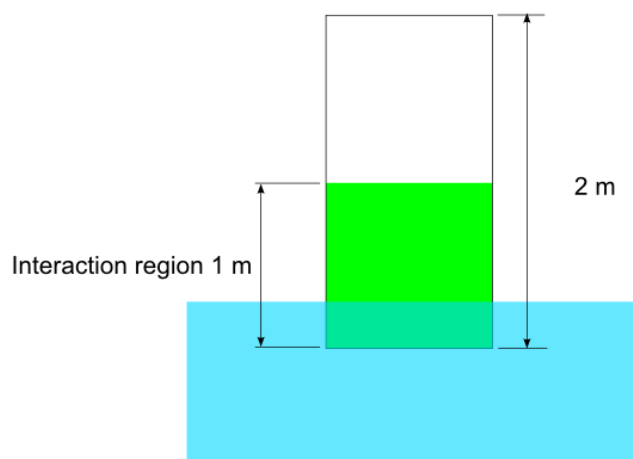


Figure 23 – Interaction Region

This is the bottom region of a colliders bounding box that can affect the velocity of a water vertex. This value is used to limit the ability of the objects with big bounding boxes to affect the velocity of the surface vertices. Only the velocity of the vertices that are inside the interaction region will be affected by an object, **Figure 23**.