WATER & BUOYANCY SYSTEM

Version 1.0

Intro

The Water & Buoyancy system is mainly comprised of three main components:

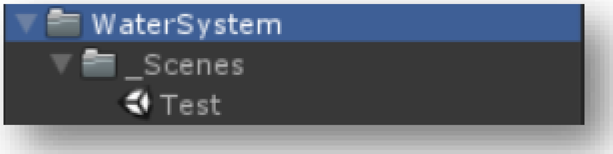
- The Water Shader
- The Water Component
- The Buoyancy Component

The Water Shader is just the Shader you will use to create your water materials, and is responsible for the way the surface of the water looks. It is **not** responsible for the waves and general surface deformation - that all happens on the CPU side in the Water Component.

The Water Component is a script that is applied to the Water object in your scene. This controls the deformation of the water, and this is where you will tune and preview all the waves and noise on your water. It also controls the water's side of the buoyancy calculations, and manages the creation and removal of Buoyant Force objects at runtime.

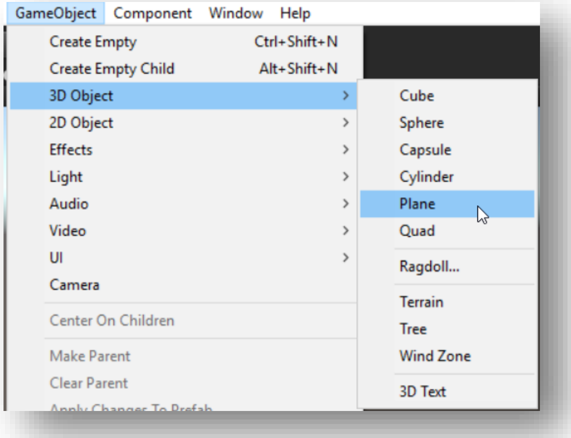
Lastly the Buoyancy Component is applied to any objects that you want to float. While the Buoyant object needs to have a collider of some sort for initial collision detection, it must also have a specified Buoyancy Hull mesh as well. Much like a mesh collider, this is expected to contour the shape of the mesh, but cost-savings go a long way towards making the simulation run smoothly, so I strongly advise making the Buoyancy hull as low-poly as possible.

If you're ever confused how any of these components should be set up, feel free to refer to my demo scene, and it should give you a good example as to what a reasonable setup might look like.



Without further ado, let's jump into setting up each of these components individually.

Water Component



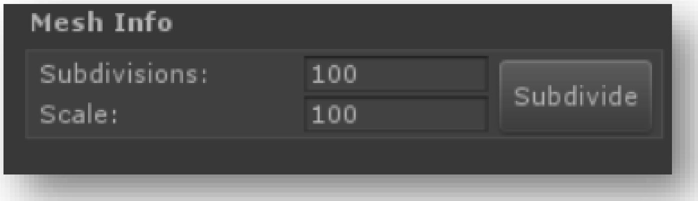
Creating a Water Plane

To create a Water Plane, you can start by just using the GameObject menu to create a 3D plane. By default, the plane should be oriented facing upwards (this is good!). A few things to note here:

- You will always want the water object's Y position to be 0. The shader will act strangely if it's any higher or lower than this.
- It's probably also best if X and Z are 0 as well.
- Don't scale the water plane in any axis. Use the Mesh resolution settings noted below to adjust the size of the water.

Once you have your plane situated at (0, 0, 0), go ahead and assign the Water component to it. You'll notice that it created a Water component as well as a Mirror Reflection component. The Mirror Reflection is necessary to have realtime reflections in-game, so the Water component creates it automatically.

Setting Water Mesh Resolution



Subdivisions: 100
Scale: 100

The Water Component inspector features the ability to resize and re-tessellate your mesh via the inspector. Just set the Scale and Subdivisions fields on the Editor, and hit the Subdivide button. After you do this, it will delete your existing mesh and create a new one with the specified settings.

NOTE: If you had any vertex colors or any other mesh information on your plane, it will not persist if you use the subdivide option.

Adjusting Water-side Physics Settings

The Physics box on the Water component script contains a few settings regarding how it handles physics for buoyant objects.

Physics Depth

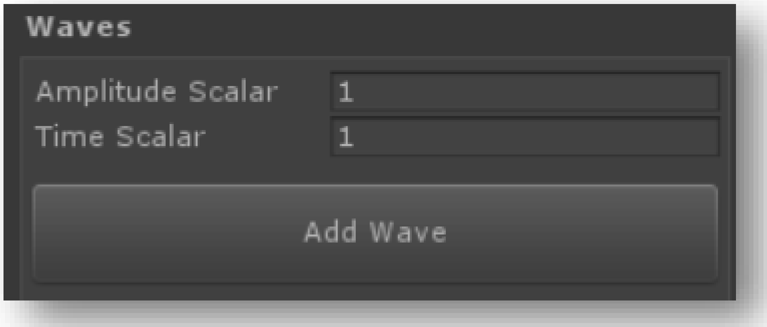
At runtime, a box collider will be generated for the water. This will span in the X and Z directions based on the dimensions of the mesh. The Y length of the box collider is determined by the Physics Depth field. When something leaves the water's collider, its buoyancy will stop being simulated, so if something sinks beyond the Physics Depth, then it will simply fall out of the simulation and continue downwards, and we won't waste any more time calculating it.

Simulation Quality

This is a global value that applies to all buoyant objects that float in this water. The higher the Simulation Quality, the more accurate the buoyant forces applied to that object will be, but the more expensive those simulations will be as well. Check out the section on Buoyant Force Planes for more details on how this works.

Creating Waves

The Wave Editor



Amplitude Scalar 1
Time Scalar 1

The Wave Editor is the area on the Water Component Editor where you can add, remove, and edit your waves. If you have no waves, you will simply see two fields: Amplitude Scalar and Time Scalar. These are values that affect every Wave on your Water, so they can be used to easily adjust the look of the system after all the setup is done.

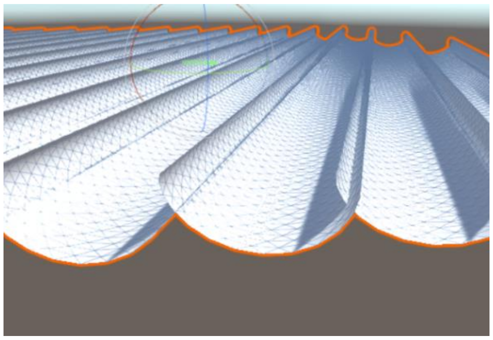
Adding a Wave

To add a Wave, all you have to do is hit the large “Add Wave” button. Once you do, the new Wave and all its settings will appear beneath the button.

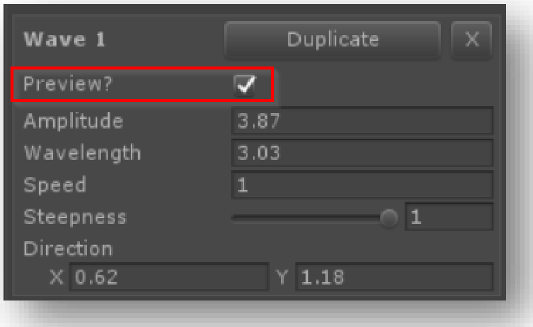
Editing a Wave

The wave is just a set of mathematical values used to determine its shape.

- **Amplitude** - The height of the wave (in meters)
- **Wavelength** - The distance from crest to crest (in meters)
- **Speed** - How quickly the wave cycles
- **Steepness** - How sharp the wave is. 1 is pointy, 0 is smooth.
- **Direction** - The direction the wave travels in. Overdriving these values past a normalized vector will give you artifacting if your steepness is set to 1. (Pictured below)



Previewing a Wave



Preview? ☑
Amplitude 3.87
Wavelength 3.03
Speed 1
Steepness 1
Direction
X 0.62 Y 1.18

To preview your waves, just go to the top of the Water Component and hit the big Preview button. This will cause your waves (and noise) to play, even in the Editor. You can also adjust the values live to see how they change the look. You can also mark individual Waves and Noises to be previewable (or not) in their Editor. When you are done previewing, just hit the Preview button again (now it says ‘Stop’) to stop.

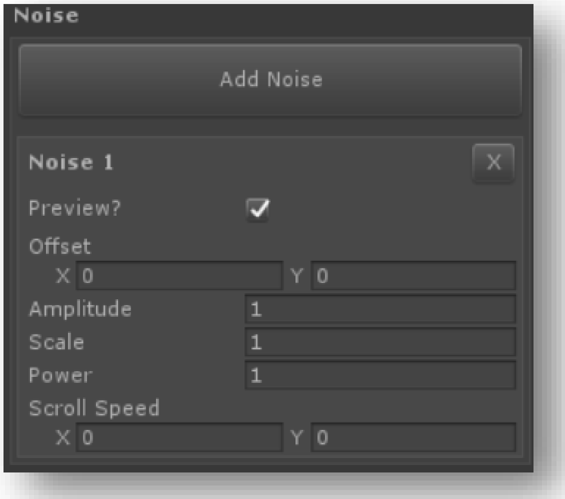
Removing a Wave

To remove a Wave, all you have to do is click the “X” button in the top-right corner of its editor. This may throw an error in the debug window - disregard it. Everything is fine.

Duplicating a Wave

To duplicate a Wave, just hit the “Duplicate” button at the top of the editor for the Wave you want to duplicate. This will create a new Wave at the bottom of the list identical to the one you duplicated.

Noise



Noise 1

Preview? ☑

Offset X 0 Y 0

Amplitude 1

Scale 1

Power 1

Scroll Speed X 0 Y 0

Noise can be created, edited, removed, and previewed just like waves, so no need to be redundant there. Noise is just simple Perlin, with some basic mathematical parameters.

- **Offset** - The initial offset for the noise. If you are using more than one, this can be used to make sure they don't start in the same place.
- **Amplitude** - The height of the noise (in meters).
- **Scale** - This is basically the "wavelength" of the noise - how broad it is on a 2D plane.
- **Power** - Similar to the "steepness" variable of the wave - how sharp is the falloff?
- **Scroll Speed** - The X and Y speed at which the noise moves.

NOTE: Because of the associated computation times, normals are not calculated for noise like they are for waves. For the best possible look, I recommend keeping noise to a minimum - only use it to break the monotony in your waves.

Mirror Reflection

First a quick note: I didn't write this script, so I can't guarantee I can debug it.

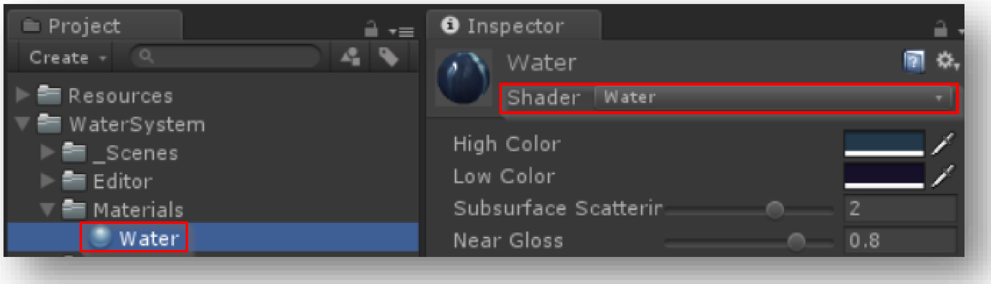
This script generates realtime reflections to be passed into the Water Shader. It does have a few values that can be tweaked.

- **Disable Pixel Lights** - Whether or not lights should be disabled in the reflection. Since the reflection here is basically just tones and silhouettes, I recommend leaving this checked.
- **Texture Size** - The size of the generated reflection texture. Again, we're just looking for basic tones and silhouettes, so I like to leave it at 256.
- **Clip Plane Offset** - The Clip Plane offset for the rendering camera. Unlikely to be relevant, so the default value here is fine.
- **Reflect Layers** - Which layers you would like to show up in the reflection. If there are any objects that you would like to exclude from the water's reflection (maybe for performance reasons), this would be the place to do it.

Water Shader

Creating & Applying the Water Shader

When creating a new Water Shader, I highly recommend duplicating my existing one, located at “Assets / WaterSystem / Materials / Water”. Most of the tedious setup is done for you, and it’s much easier than starting from scratch. If you do want to make a new one, however, just create a new material and assign the “Water” shader in the dropdown.



Shader Water

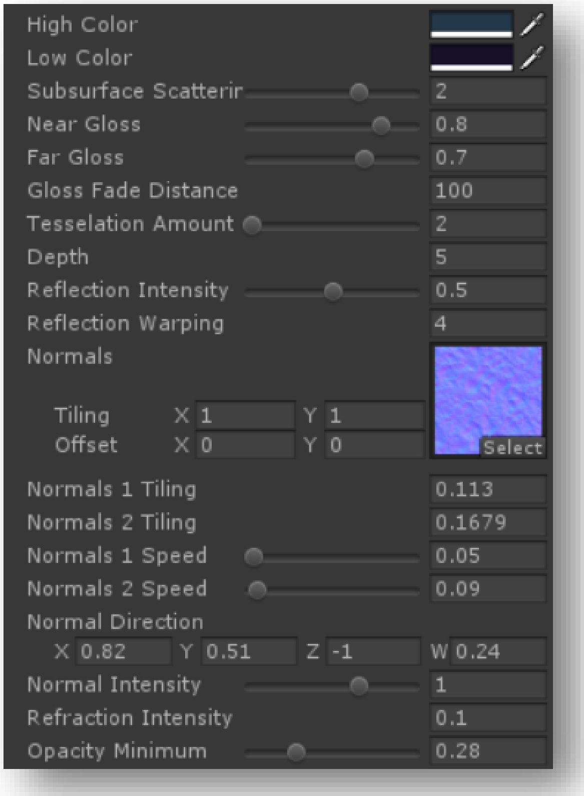
Subsurface Scatterir 2

Near Gloss 0.8

To apply the Water Shader, just drag it on top of your water plane in the scene view, or assign it in the Mesh Renderer component.

The Water Shader Values

The Water Shader has a lot of things going on, so I’ll take a moment here to explain what they all mean.



High Color	
Low Color	
Subsurface Scatterir	2
Near Gloss	0.8
Far Gloss	0.7
Gloss Fade Distance	100
Tesselation Amount	2
Depth	5
Reflection Intensity	0.5
Reflection Warping	4
Normals	
Tiling	X 1 Y 1
Offset	X 0 Y 0
Normals 1 Tiling	0.113
Normals 2 Tiling	0.1679
Normals 1 Speed	0.05
Normals 2 Speed	0.09
Normal Direction	X 0.82 Y 0.51 Z -1 W 0.24
Normal Intensity	1
Refraction Intensity	0.1
Opacity Minimum	0.28

High Color & Low Color

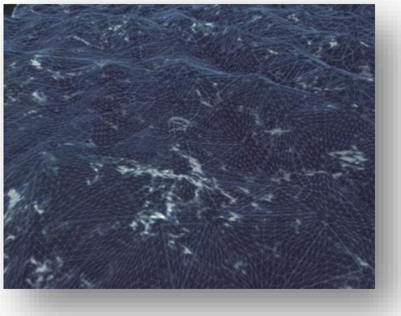
The Water shader uses the wave height to change the diffuse color. The High Color is the color that the waves will be at their peak - the Low Color is what the diffuse color will be at the trough. If you want these to be the same, just assign the same color to each.

Subsurface Scattering

As the name implies, how much the water scatters light. Higher values make the lighting on the water softer, but don't go higher than 2.

Near Gloss, Far Gloss, Fade Distance

The Water shader blends gloss values depending on how far the camera is from the pixel being rendered. Near Gloss is the glossiness of a pixel directly in front of the camera, Far Gloss is the glossiness of a pixel that is (Gloss Fade Distance) meters away from the camera and beyond.



Tessellation Amount

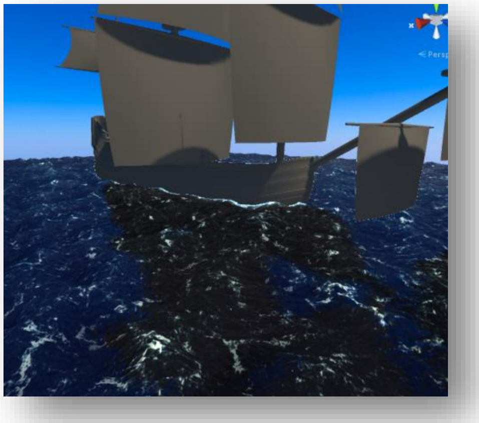
This value controls the phong tessellation applied to the mesh via shader. Somewhat unintuitively, the lower the amount, the more tessellated the object is.

Depth

How far beneath the water you can see another object before the surface of the water becomes opaque.

Reflection Intensity

This controls the intensity of realtime reflections on the surface of the water.

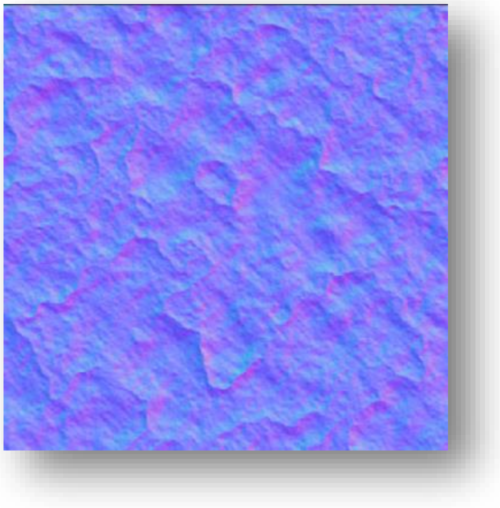


Reflection Warping

The extent to which the reflection is warped by the surface normals of the water.

Normals

The Water Shader uses two instances of the same Normal map scrolling against each other. The Normals texture option is looking for a normal map to act as the scrolling surface normals for your water.



Normals 1 & 2 Tiling

The Tiling values for the two different samples of the normal map. A couple tips - make sure the numbers aren't multiples of one another, and add as many erroneous decimal places as possible to ensure that an overlap is incredibly unlikely.

Normals 1 & 2 Speed

The speed at which the normal maps scroll.

Normal Direction

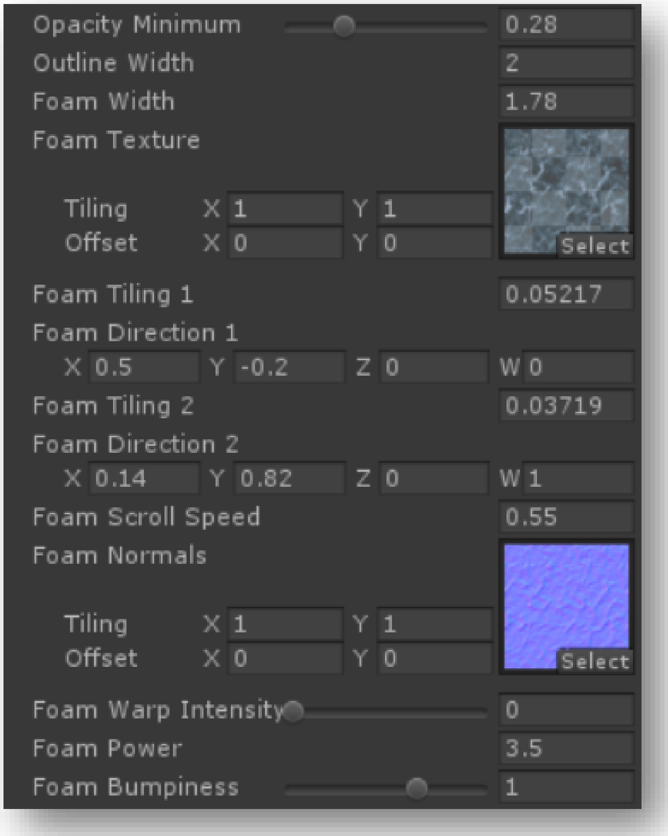
The direction information for the normal map scrolling has been trimmed into one Vector 4, so the X & Y values refer to the U & V values of the first sample, and the Z & W values refer to the U & V values of the second sample.

Normal Intensity

This scale the surface normal intensity of your normal maps, and ranges from 0 to 1.5. A value of 0 will only show the mesh normals as defined by the waves, and a value of 1.5 will exaggerate the surface normals from the scrolling.

Refraction Intensity

Objects beneath the water will be refracted based on the surface normals of the water - this field controls just how much warping is present.



Opacity Minimum 0.28

Outline Width 2

Foam Width 1.78

Foam Texture

Tiling X 1 Y 1

Offset X 0 Y 0

Foam Tiling 1 0.05217

Foam Direction 1

X 0.5 Y -0.2 Z 0 W 0

Foam Tiling 2 0.03719

Foam Direction 2

X 0.14 Y 0.82 Z 0 W 1

Foam Scroll Speed 0.55

Foam Normals

Tiling X 1 Y 1

Offset X 0 Y 0

Foam Warp Intensity 0

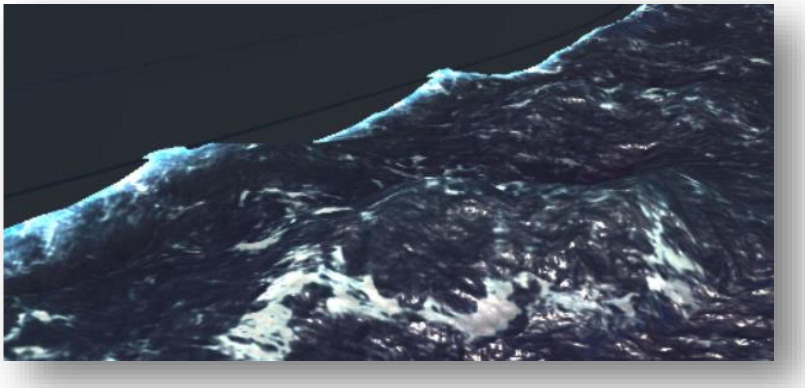
Foam Power 3.5

Foam Bumpiness 1

Opacity Minimum

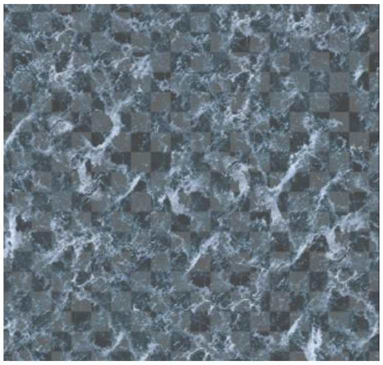
This is the least opaque the water will ever get. Even if there is a visible object immediately below its surface, the water shader will still draw over top of it with at least that much opacity.

Outline Width



The Water Shader uses an outline to highlight intersection with other objects. This controls the width of that foam intersection.

Foam Texture



This refers to a texture to be used as the foam on top of the waves. It should have an RGB diffuse texture as well as an Alpha channel that determines its opacity overlaying the water.

Foam Tiling 1 & 2

Just like with the normals, the Foam is sampled twice and scrolled against each other to make it look more dynamic. These values, just like with the normals, control the tiling values for the individual samples.

Foam Direction 1 & 2

Unlike with the normals, this time you have two Vectors to orient the direction of the scrolling. Disregard the Z and W values - they do nothing.

Foam Scroll Speed

Unlike with the normals, the speed value applies to the scrolling-speed of **both** samples of the Foam texture.

Foam Normals

This is a normal map generated for the Foam. It will be factored into the surface normal calculation to ensure that the foam looks like it has volume and doesn't adhere to the choppy water underneath it.

Foam Warp Intensity

This value applies a warp to the foam based on whether or not it's sloping down off a wave. Depending on the intensity of your simulation, this can be a cool effect.

Foam Power

This controls the sharpness of the falloff for the foam. The foam will only show up on the tops of waves by default, but you can tweak exactly how low or high the foam goes here.

Foam Bumpiness

Here you can control the intensity of the Foam's normal map, just like with the water surface normals.

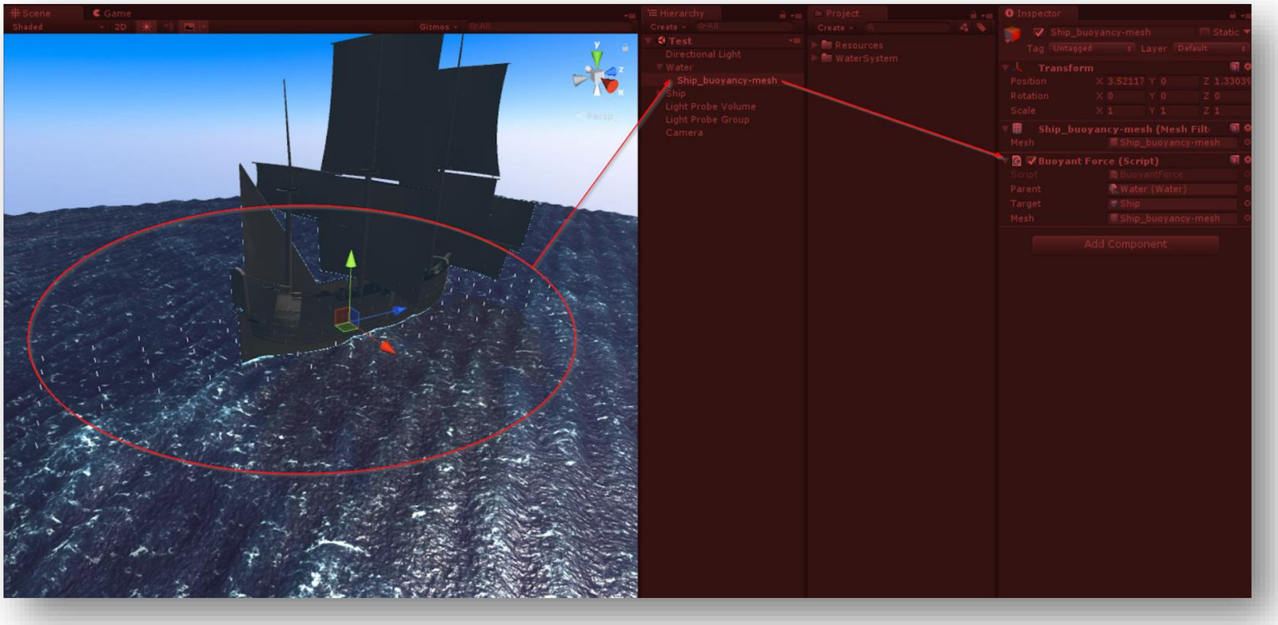
Buoyancy Component

What is it?

The Buoyancy Component is what you would apply to objects that you want to float in the water. Any buoyant object must have a collider and a rigidbody, otherwise it will not be simulated.

How does it work?

When an object with a Buoyancy component comes into contact with the water, the Water object creates a "Buoyant Force Plane" that follows the floating object and applies buoyant forces to it as it floats. In the picture below you can see the gizmos drawn for the buoyant force plane (it doesn't have a mesh renderer so you can't actually see it), as well as the hierarchy and inspector for it.



Test
Directional Light
Water
Ship_buoyancy-mesh
Ship
Light Probe Volume
Light Probe Group
Camera

Resources
WaterSystem

Ship_buoyancy-mesh Static
Tag Untagged Layer Default

Transform

Position	X 3.52117	Y 0	Z 1.33039
Rotation	X 0	Y 0	Z 0
Scale	X 1	Y 1	Z 1

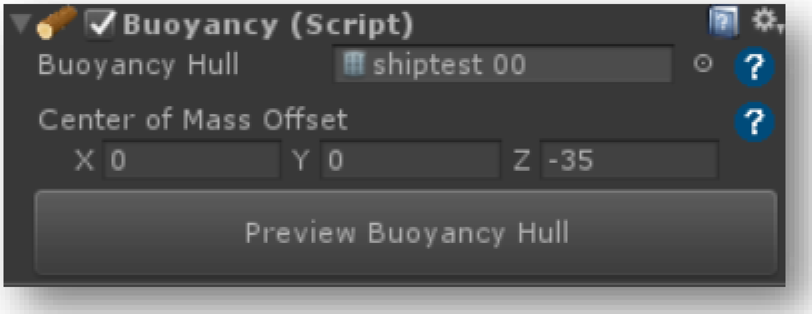
Ship_buoyancy-mesh (Mesh Filt

Mesh	Ship_buoyancy-mesh

Buoyant Force (Script)

Script	BuoyantForce
Parent	Water (Water)
Target	Ship
Mesh	Ship_buoyancy-mesh

Buoyancy Hull


Buoyancy (Script)

Buoyancy Hull: shiptest 00

Center of Mass Offset: X 0 Y 0 Z -35

The Buoyancy Hull is the mesh referenced by the water simulation to determine how much of the mesh is submerged and how forces should be applied. This should be similar to a mesh collider, **but it should be as cheap as possible**. Buoyancy simulations are expensive - reducing the number of triangles that need to be accounted for goes a long way. To see your Buoyancy Hull in the editor, just hit the “Preview Buoyancy Hull” button.

Center of Mass Offset

This offsets the center of mass for the rigidbody on Start. This is being used in the ship to move the center of mass way lower to make it much more stable and upright.