SIMPLE BUOYANCY PHYSICS

Version 1.0

Introduction

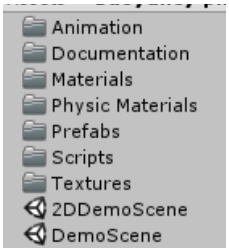
First of all, thank you for purchasing this asset. As you may know, you can simulate basic buoyancy physics in your games with a few clicks using this code.

This asset is easy to use and configurate. However might not be suitable for games that are supposed to recreate realistic water simulation.

You can get it working on any platform. Has been tested on PC, Web Player, Android and iOS so far with excelent frame rates.

1. Creating your first water and objects - 3D

If you haven't the package imported, you must import it througt Unity Asset Store. Once is loaded, you will find a new folder into your Proyect tab called “Buoyancy physics”. If you click on this root folder you will find sub-folders containing Scripts, Prefabs and a couple of example scenes.

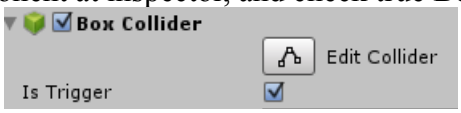


But let's start a new scene to learn how to implement buoyancy to your game.

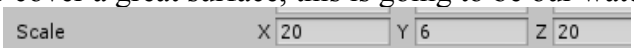
First, we will start with a 3D based scene.

First of all, you will need **water:**

- Let's create a cube (GameObject > 3D Object > Cube).
- Once created, rename it to “Water”.
- Check the object's component at inspector, and check true Box Collider to Is Trigger



- Scale the object to cover a great surface, this is going to be our water after all.

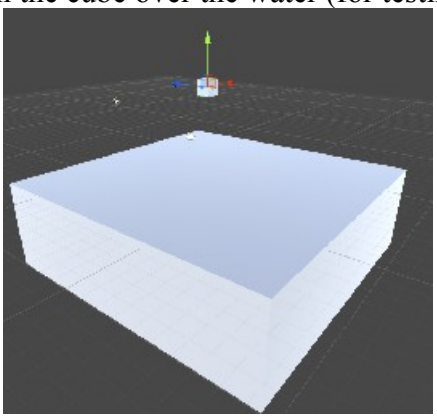


You can scale the water at any size you want.

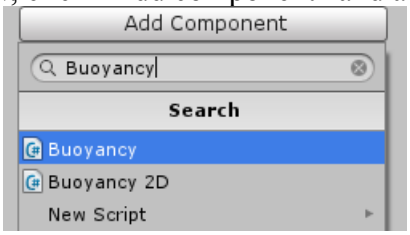
That's it! We have our water ready.

Now, lets create our **first object**. This time a cube will work for us.

- Like before, create a cube (GameObject > 3D Object > Cube).
- In the scene view, position the cube over the water (for testing purposes).

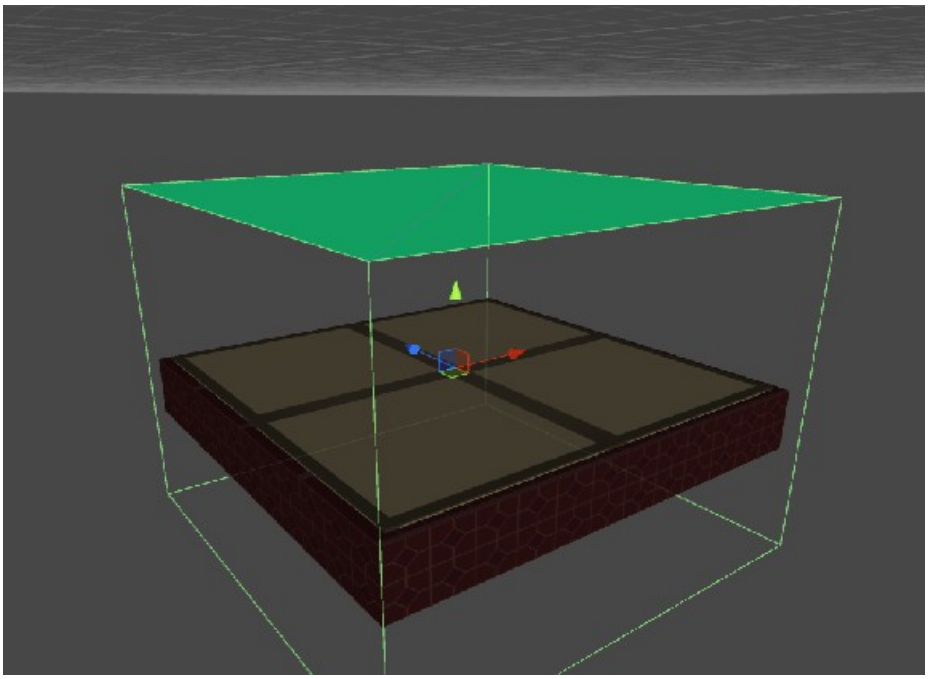


- Go to cube's inspector view, click “Add component” and add “Buoyancy”.



- Well done! Press play to test it.

Awesome, this is it. Now you may think that this water is ugly. Well, it is, cause is not water. You can set the object “Mesh Renderer” to false or delete it and put a custom mesh (a plane or a quad, for example) on the top of the collider, just like in the Demo Scene.



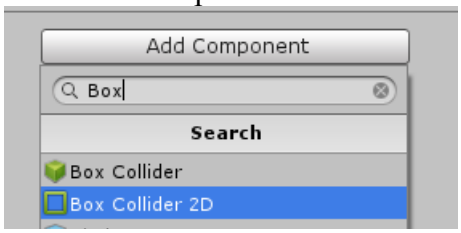
This water object can be animated or moved, at any height, or as many as you want in your scene, no need to extra configuration.

2. Creating your first water and objects – 2D

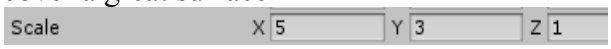
Easy as 3D. Create a new scene and set the view to 2D for a better control.

Now, lets create water!

- Create an empty sprite (GameObject > 2D Object > Sprite) and name it “Water”
- In the inspector of this new sprite, set an sprite for the “Sprite Renderer”
- Staying in the inspector, add a new component called “Box Collider 2D”

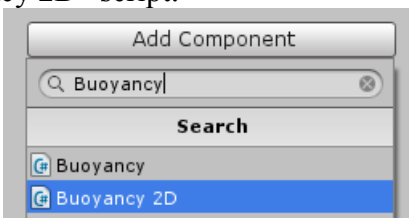


-
- Scale the object to cover a great surface

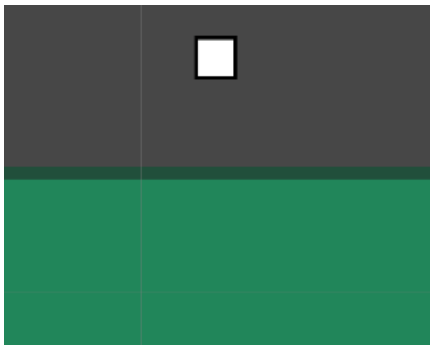


Done! Now let's create an object too:

- Again, create an empty sprite like before.
- Attach it a 2D Collider and set its “Is Trigger” to true.
- Set a sprite for this object, scale it as you want, and place it over the water.
- Now attach it the “Buoyancy 2D” script.

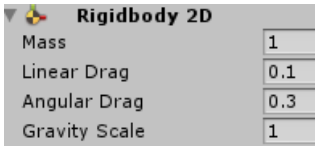


Again, you are done! Press play to test it.



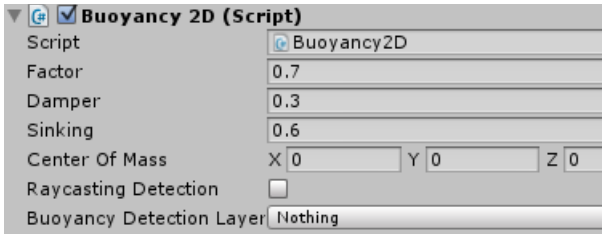
3. Further configuration
Are you wondering how to simulate different kind of buoyancy? Wonder no more. Lets explain the configurable parameters of an object. Configuration is almost the same either for 3D and 2D, so lets check 2D because it's the same as 3D, but with a few more things.

First, as you may know, every object that simulate physics needs a rigidbody (at least in this case). **Rigidbody,** either 2D or 3D, will affect directly to the buoyancy behaviour.



We have 3 basic values that might affect buoyancy. For example, increasing **Mass** will make the object sink faster and deeper, and push other object with less mass to the bottom or viceversa. **Linear Drag** (or just **Drag** in 3D) makes the object buoyancy softer, as it does out of water too. And **Angular Drag** will stop the object faster at higher values from turning around itself.

Now lets check **Buoyancy (2D) parameters**.



Factor: increases the force that is applied to the object from the surface.
Damper: smooths the object forces.
Sinking: determines how much the object will sink.
Center Of Mass: where's the center of the buoyancy? Factor force will be applied from there.

(Only 2D)
Raycasting Detection: allows to use Meshes or Edge Colliders if you have an animated mesh (such waves or deforming mesh)
Buoyancy Detection Layer: if Raycasting Detection is enabled, you will need to set this layer as the same layer of the water.

Note: in this package, the 2D Example is using this buoyancy method. Note that the water is using and Edge Collider and every 2D Prefab has this two parameters configurated.

Usually by changing the rigidbody mass you will get an object sinking deeper, but you can tweak the Buoyancy parameters to get it floating over the surface (for example, to make floating bridges).

For example, if you check the 2D prefab object "HeavyBox" you will see that is sinking to the bottom. It's 5(kg) and it's Buoyancy "Factor" is 0.3. If I set it to 5, it will be floating over the surface.

I recommend playing with this parameters to get the desired behaviour.

Thanks for reading this little Starting guide and I hope you find this asset useful.

Please, if you have any question, write me at:
motbest@gmail.com

I will reply back as soon as I can.