Name: Kuan-Ting Chin

Student ID: 33430072

Maths Assignment: Wave simulation with buoyancy test.

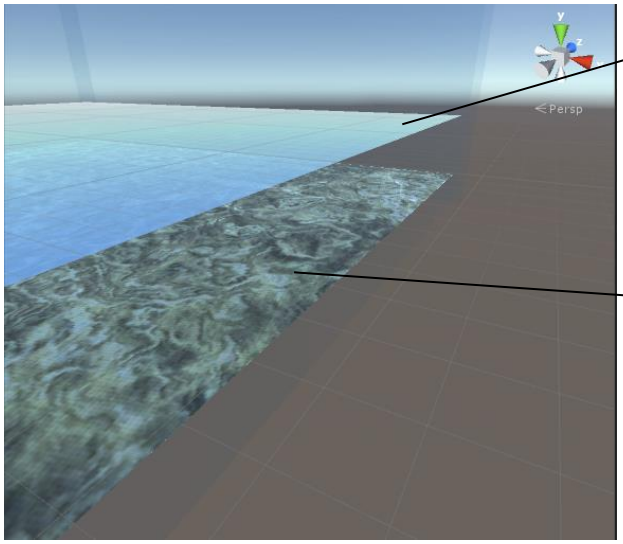
Aim: To create a simulation project to render in real time.

Objective:

The main objective of this project was to create a simulation of ocean waves. For my project, I also decided to include buoyancy test allowing the user to see different forces affecting the object via waves. The project was created in Unity with C# scripting as the main focus.

Project Outline:

The project was created using 2 Gameobject planes, one which represented the ocean itself, and another which represented the sea bed.



Sea Plane Gameobject

Sea bed Plane Gameobject.

The seaplane has a blue default material colour, with Fade rendering mode to simulate translucent water. On top of the planes, the project also included the use of projectors to create Caustics Images, thanks to the Caustics Generator. This tool provides animation through refracted lighting in water, done through automatically generated images, and Unity's Projector function.

Preview of images. Ability to change the colour if you so wish.

Rendered images, akin to sprite sheets.

The first step was to simulate the waves. This lead to the creation of a game manager to handle the wave controls. The wave control composed of a scale slider:

```
scaleSliderValue = GUI.HorizontalSlider(new Rect(100, 94, 100, 30), scaleSliderValue, 1.0f, 3.5f);
```

This alters the scale of the ocean waves, and all limited by the inbuilt OnGUI HorizontalSlider function. This allows a look at simulation of potential waves as well as different wave forms to help simulate the buoyancy tests. This is the same for the speed as well as the frequency, with the frequency slider having set interval values instead of true float values.

The wave control script has the GetWaveYpos method with 2 parameters (x co-ords and z co-ords) which allows the WaveGenerator script to generate sine waves from the main formula:

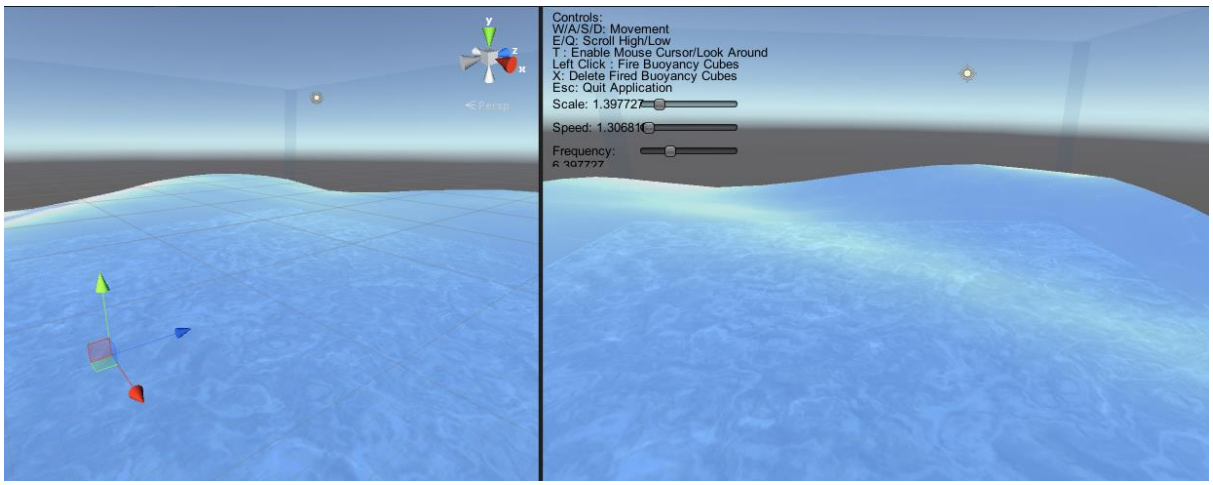
```
y_coord += Mathf.Sin((Time.time * waveSpeed + x_coord + z_coord)* waveFreq / waveDistance) * waveScale;
```

The next line of code within the method is the perlin noise generation

```
y_coord += Mathf.PerlinNoise(x_coord + noiseWalk, z_coord + Mathf.Sin(Time.time * 0.1f)) * noiseStrength;
```

This allows the waves to have a modifier that allows the waves to seem more natural as opposed to a more perfect flow.

The method is fed in the vertices x and z co-ordinates in the WaveGenerator Script, which calculates and loops through all vertices of the plane (the sea) and modifies the y co-ordinate, which would be the height of the waves. This is to loop through the vertices of the plane and calculate the original vertices, followed by the new vertex of the plane once a wave is passed through, and store the new vertex information to calculate the height of the wave for the buoyancy test.



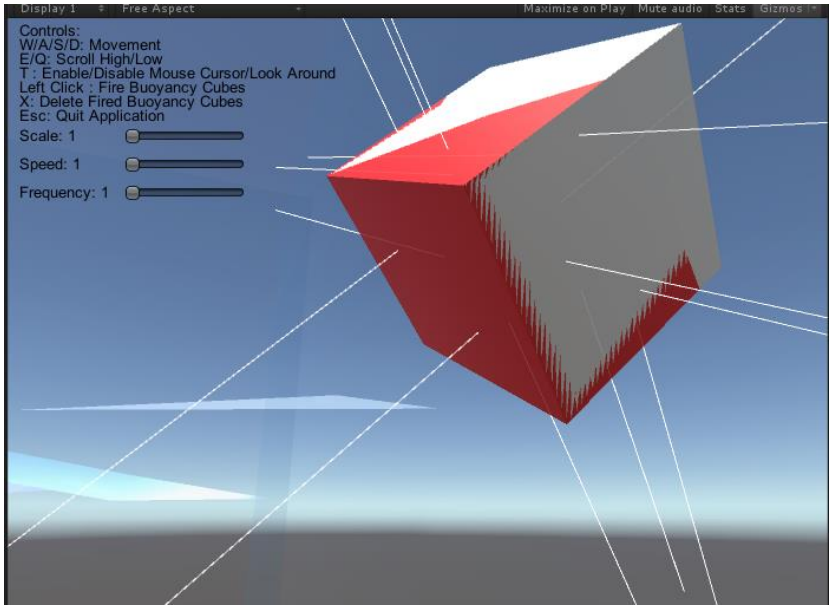
Example of waves working with manual configuration thanks to slider controls.

After the waves were created, the next was to focus on buoyancy of objects. The object themselves would be standard cube prefabs in Unity's library, but would require scripting the objects so the vertex count would be clockwise. This is to ensure that the objects' vertices in the object array form a clockwise loop on the corners to ensure the triangle images are not inside out.

There was a problem with the cube oscillating and having unrealistic movements, but according to the website Habrador, whom I have based the buoyancy script on, altering the object you wish to test buoyancy on's maxAngularVelocity of the rigidbody to 0.5 solved the issue.

The next step was to create a method which detected if the object was underwater or under the effect of water, and if so by how much. This is done via the ObjectBuoyancy script, with multiple test cases that determine if an x amount of corners are above or below water, with lists of underwater vertices and triangles. We would be able to calculate the center of the triangle with the help of the vertices, which allows us to calculate the distance to the surface from the center. This is done via a abstract class Distance. Once that is done, we calculate the normal of the triangle, utilizing cross product calculations and normalizing the value.

To add buoyancy force, we need to utilize the Hydrostatic force formula:
dW * g * z * dS * n, where dW is the density of water, g is gravity, z is distance to surface dS is surface area and n is the normal to the surface. This is the key formula to allow the object to float, alongside Archimedes principle, where "The upward buoyant force that is exerted on a body immersed in a fluid, whether fully or partially submerged, is equal to the weight of the fluid that the body displaces and acts in the upward direction at the centre of mass…".



The object transitions from white to red depending on if the object is submerged in water or not. The lines appearing on the object indicate force acting on the object, indicating that a force is being applied to enact a buoyant force simulation.

Conclusion

Overall, I felt this project allowed me to really see how much is required to simulate real life events like ocean waves and buoyancy, and the importance of understanding how each element in the mathematical world is actually applicable everyday natural occurrences and events. Whilst I did base this project heavily on the tutorial provided by Habrador, I felt I have gained an extensive understanding of wave simulation through sine waves, new technology to simulate real life events and Hydrostatic forces and its relevance to Archimedes Principle.

Reference:

CAUSTICS GENERATOR - SEAMLESS WATER TEXTURE RENDERING

In-text: ("Caustics Generator - Seamless Water Texture Rendering")

Your Bibliography: "Caustics Generator - Seamless Water Texture Rendering". *Dualheights.se*. N.p., 2016. Web. 10 Mar. 2016.

TUTORIAL: HOW TO MAKE A REALISTIC BOAT IN UNITY - ADD WAVES | HABRADOR.COM

In-text: ("Tutorial: How To Make A Realistic Boat In Unity - Add Waves | Habrador.Com")

Your Bibliography: "Tutorial: How To Make A Realistic Boat In Unity - Add Waves | Habrador.Com".*Habrador.com*. N.p., 2016. Web. 25 Mar. 2016.