CSC3231 Coursework

Alex Barnett 200960975

Standard Features Implemented:

- A section of the planet surface.
- Multiple textures on the planet surface
- The environment should be lit.
- Some elements moving and rotating.
- A scene hierarchy where some elements move/rotate in relation to parent objects
- A navigable camera, moving around the 3D environment.
- Display the framerate being achieved.
- Display the total memory being used.

Advanced Features Implemented:

- A body of realistic looking water
- Strange lights in the sky
- Weather effects (rain, fog, clear).
- A day/night cycle with appropriate lighting changes
- Lens-flare (Custom Render Feature)
- Any other advanced graphical effects. (Custom Fog Render Feature)

Water:

For the water I used a shader graph, scrolling normal textures over the surface to create wave effects which line up according to their real-world position in the scene. This is then applied to a material which is added to a quad, my scene includes many of these quads in a large grid, together creating a large body of water. To crate the graph I watched some tutorials on water similar to what I was looking for, then built of them to create one which worked best for me.

Strange lights in the sky:

I created a light object prefab using a URP lit material with a coloured emission, and a point light. These prefabs are then instantiated in LightSpawner and move upwards in a spiral around the spawner as controlled by SpawnedLightController

Weather effects:

The main weather effect implemented is the rain. This rain was created using a VFX graph, this runs on the GPU and so is much more efficient than a regular particle emitter. My Visual Effect Graph is somewhat based on this tutorial: https://danielilett.com/2020-05-19-tut5-7-urp-rain/, mainly in the “Rain Particles” section. I used this as a baseline to build my work off and created a rain effect which worked for my use case. Weather can be changed in-game using the buttons controlled by WeatherController script. This uses my WeatherCondition class and a coroutine to change the attributes of my sun, rain emitter, fog and lens flare materials over time to create transitions between each weather condition.

A day/night cycle:
The time of day can be changed using the buttons in game to set times, as controlled by TimeController script, this translates selected times to angle rotations for my sun object. It also alters my post-processing volume effects when it changes from day to night, or back to night. These changes include lift, gain and split toning, which change over time using a coroutine, at the same time as my sun's rotation to create a smooth transition between different time periods.

Lens-flare:
The lens flare I made is a custom render feature, this is created in my CustomLensFlareFeature and uses a ScriptableRenderPass to draw the lens flare effect on top of my scene. This uses my custom shader for the lens flare which takes the scene from camera view as an input and returns the scene with the lens flare effect drawn on top of it.

Custom Fog Render Feature:
I also have another custom render feature I have made for the fog; this is more complex than the lens-flare as it needs to also draw fog according to the real-world position not just the screen-space positions. This uses the FogEffectRenderFeature ScriptableRenderPass to pass in the screen space texture and run the fog effect. The fog is then drawn on top of the scene according to the distance to the camera to add atmosphere. It also uses the real-world height to draw fog up past the water to hide the edges of the scene.

References:
Terrain Painter for adding textures to the terrain:
https://assetstore.unity.com/packages/tools/terrain/procedural-terrain-painter-188357

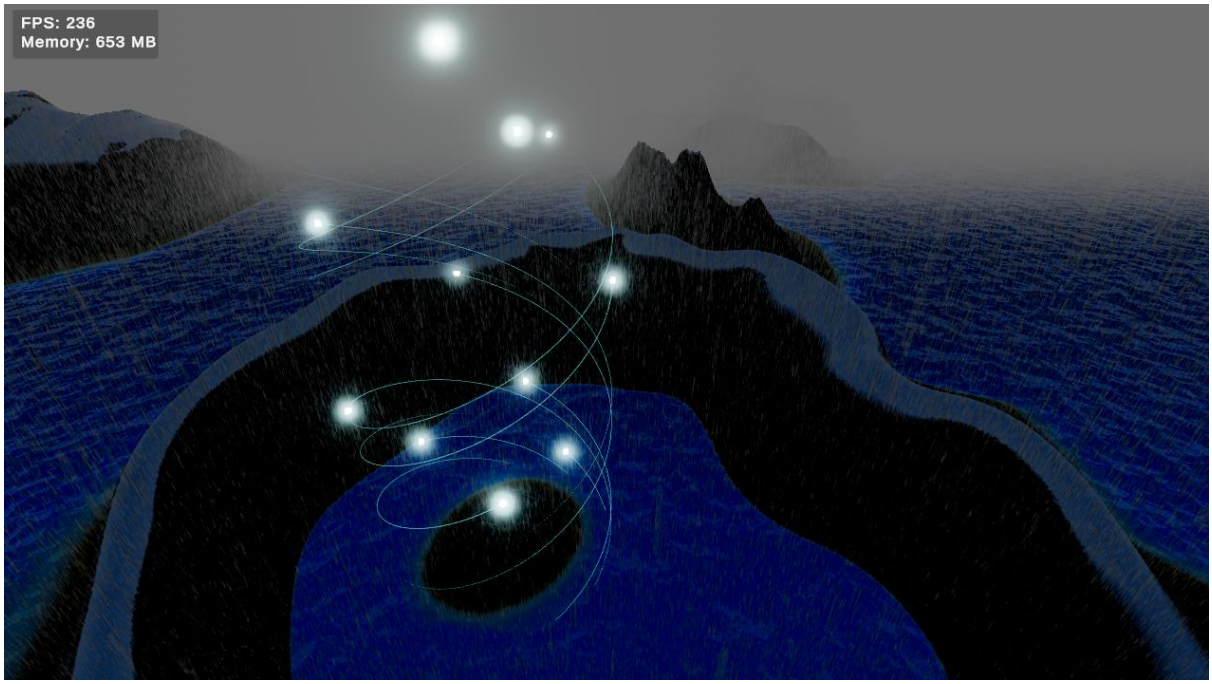
Rain VFX Graph Tutorial:
https://danielilett.com/2020-05-19-tut5-7-urp-rain/

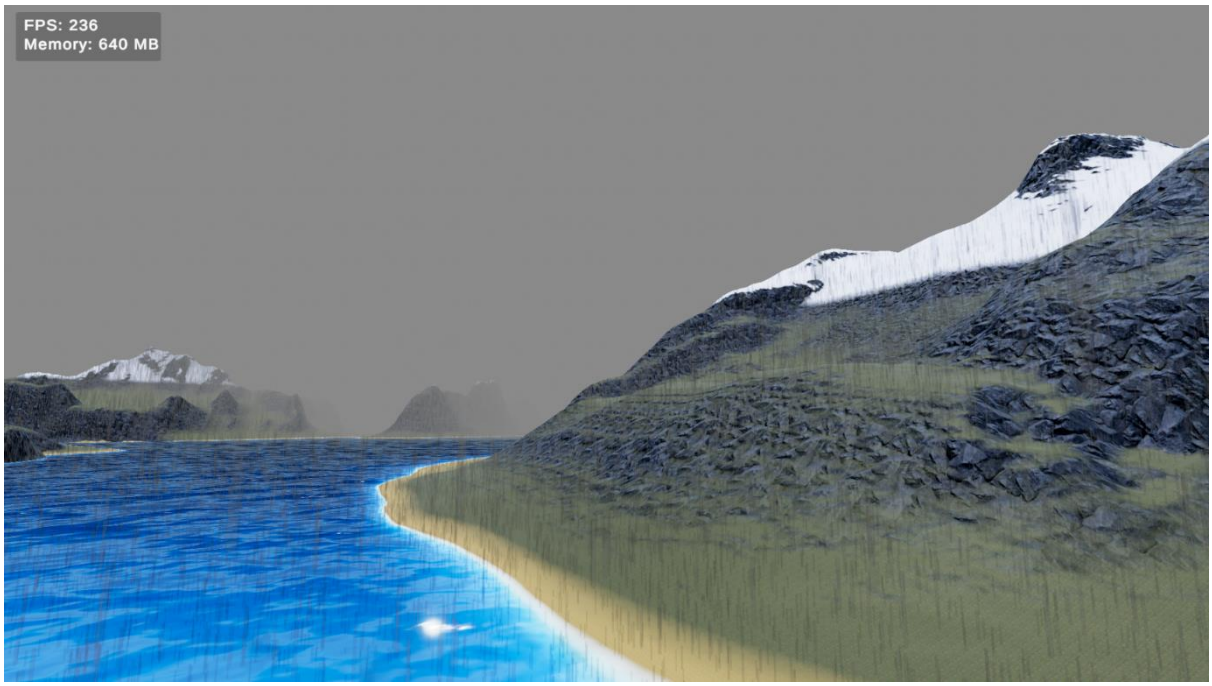
Water Shader Graph:
https://www.youtube.com/watch?v=1yevpCAA_rU

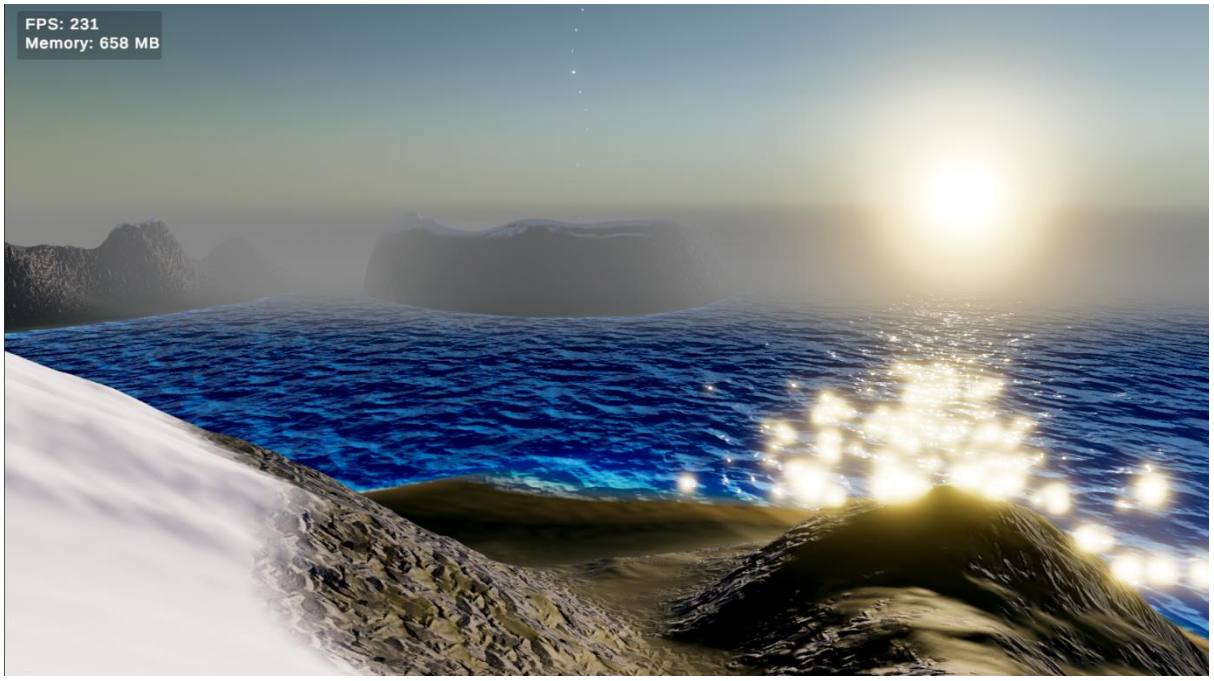
Some additional code comments added by ChatGPT:
https://chat.openai.com/

Link To YouTube Video:
https://youtu.be/6t0sLV5X16w

Screenshots:






FPS: 231
Memory: 658 MB


FPS: 261
Memory: 634 MB


FPS: 259
Memory: 664 MB