EZG2022 Group 69: Ocean Story

Miran Saman (01631794)

Chew Wen Jie Jeremy (12229288)

- ***Brief description of the implementation. Which technology are you using? Has anything changed from what you have planned during Submission 1 - Proposal?***

The camera pans around the landmass, while three dolphins jump over the water at specific intervals. We then dip underwater. A manta ray glides by, and we see three fishes crossing our paths. Continuing underwater, we pass by a few more fishes, before coming to a rest near some seagrass. Looking to our left, we see a whale leisurely swimming by. Turning back, two clownfish zoom past us, seemingly escaping the shark which comes by a few seconds later. We then back out of the ocean, finishing with a view of the sun, its reflection in the ocean, as well as the landmass.

We are using C++ and the OpenGL API.

There have not been changes from the original proposal.

- ***What additional libraries (e.g., for collision, object-loading, sound, …) were used, including references (URL)?***

Libraries used

- Assimp (model loading)
- GLFW + GLAD
- GLM (vector, matrix maths)
- DearImGui (Temporary debug GUI)
- STB (image loading)
- irrKlang (sound)

Resources

- Fish model with textures from Sketchfab
- Animated fish models by **quaternius** from OpenGameArt
- Seagrass models by **quaternius** from OpenGameArt
- Ocean resources
    - Dudv map + normal map from this tutorial by **ThinMatrix**
    - Caustics texture by **leeor_net** from OpenGameArt
- Cloud texture by **Luos** from itch.io
- The landmass noise texture was generated using Python and the noise library.

- Background music (RPG - Costal Town Background Music) by **Hitctrl** from OpenGameArt
- Wave sounds (Beach Ocean Waves) by **jasinkski** from freesound

- ***The graphics card on which you tested on (NVIDIA, AMD, Intel, exact model(s)).***

NVIDIA GeForce GTX 1660 SUPER

NVIDIA GeForce RTX 2060 SUPER

- ***Where in your scene can which effects be observed?***

Non-complex effects

- We render to texture quite often, and hence we created a framebuffer class that is able to render to RGB and depth textures. Furthermore, if required, it can also be initialised in such a manner to allow for post-processing effects.
- The ocean is able to reflect and refract its surroundings (achieved using clipping planes and projective texture mapping). These reflections/refractions are distorted using a dudv map.
- The ocean surface also uses a normal map for added detail. Blinn-Phong lighting is implemented as well.
- The ocean's ebb and flow is modelled using Gertsner waves.
- The "foam" around the shallow regions of the landmass is achieved by sampling the depth texture and linearly blending from white to the original ocean colour based on the depth.
- The landmass's height was obtained using a pre-computed noise texture.
- The landmass' normals were mapped using triplanar mapping.
- The sun's brightness distortion was achieved using the Henyey-Greenstein phase function.
- The blue environment is a result of a distance-based fog implemented as a post-processing effect. A box blur is also applied at far distances.
- Underwater caustics was achieved using a pre-generated caustics texture.
    - To determine whether a given fragment was underwater (and hence whether to draw the caustics), we copy the same Gertsner wave function in the ocean's vertex shader to the land's vertex shader in order to compute the height of the ocean at a given XZ coordinate. Because the Gertsner waves also move the XZ coordinates, we can only estimate the height at a given point. First, we compute the XZ displacement of the point we are interested in. Then, from the original position, we move in the opposite direction and compute the XZ displacement of that point again. Repeating this twice is enough to give a fairly accurate estimate of the height of the ocean at a given position.
- The paths of the fishes are quadratic Bezier curves. In order to avoid unnecessary draw calls, the fish paths each have a predetermined start and end time in which the fishes are drawn and updated.

- Dynamic shadows were also implemented underwater. The shadow map was generated by treating the sun as a directional light that follows the camera. As the viewing distance under water is limited, we do not need it to extend beyond that. The shadow map only includes the seaweed and fish, as other objects (Ocean, Land,...) do not cast shadows here.

Complex effects

- The underwater scenes are lit using volumetric lighting.
    - The shadow map we use is the same as for the underwater shadows.
    - Volumetric lighting was then implemented using the ray-marching approach that was shown in the lecture. However, the light contribution calculation was modified to fit our use case better.
    - The colour of the volumetrics depends on the y-depth relative to the ocean surface throughout the ray-marching. This was done to mimic the way water seems to turn more blue the deeper one goes.
- The fishes were animated using vertex skinning.
    - Assimp was used to read the initial model and bone information, which was then converted into custom structures that rely on hashmaps to avoid storing the full Assimp scene pointer, as well as avoid recursion when looking for parents and specific bones.

- **Describe possible controls of your demo (e.g. how to control the camera, how to enable/disable effects)!**

Once the animation is complete, you can control the camera as follows

WASD for moving, Space to ascend, LShift to descend

C to capture the mouse to look around, V to release it

You can also interact with the debug menu with the mouse.