Project Plan Undersea

November 2021

1 General

Our project will implement real-time rendering with OpenGL. The whole scene should be located in a fixed amount of space and camera is in the center.

Within this scene, all objects or lights are dynamic. We will define a fixed starting point and pattern of motion for them. The camera's position, direction and field of view can be interacted with mouse.

The technical features listed in our proposal will be switchable. For each feature, we can enable/disable and alter it with buttons or other Qt elements.

2 Detail plans

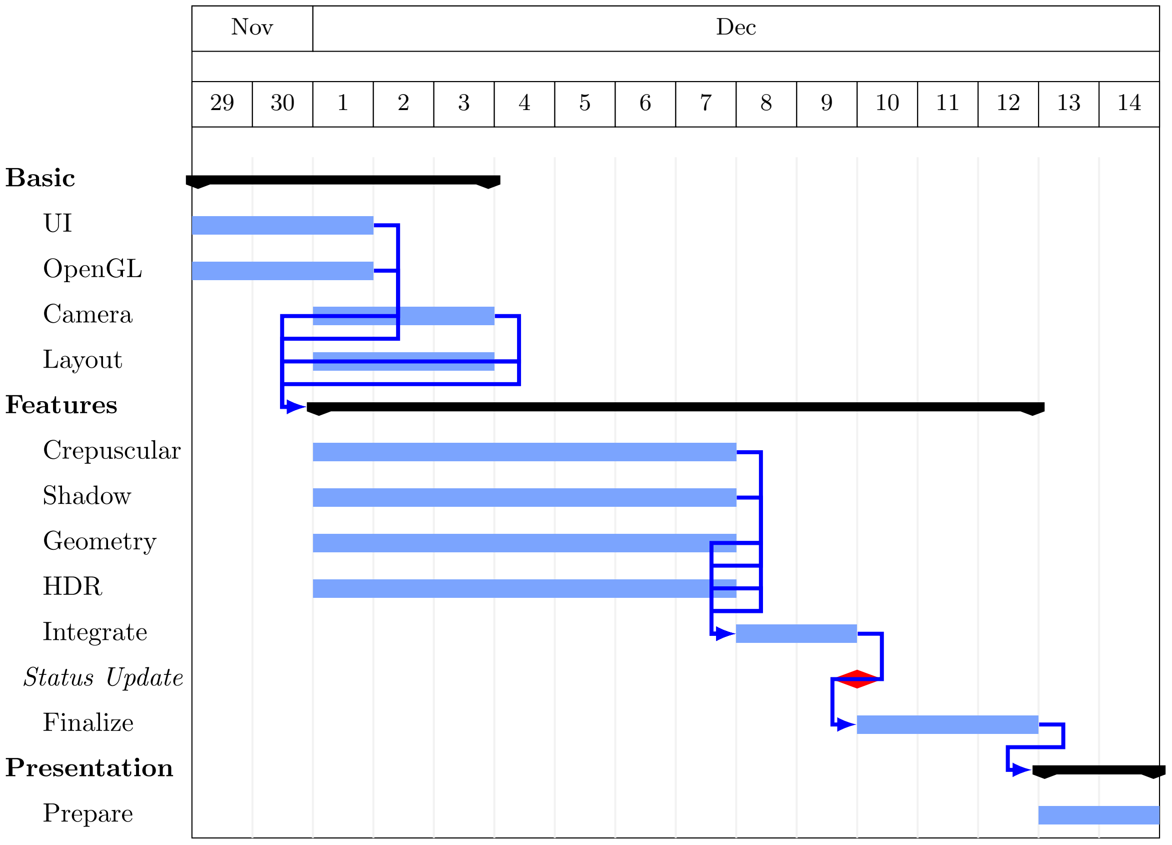

Figure 1: Gantt Chart

feature	start date	end date
UI design	Nov/29	Dec/1
OpenGL frame	Nov/29	Dec/1
Scene Layout	Dec/1	Dec/3
Camera Interaction	Dec/1	Dec/3
Crepuscular ray	Dec/1	Dec/7
Shadow mapping	Dec/1	Dec/7
Geometry shaders	Dec/1	Dec/7
HDR	Dec/1	Dec/7
Integration	Dec/7	Dec/9
Improve and finalize	Dec/9	Dec/12
Demo Preparation	Dec/12	Dec/14

Table 1: Accurate timeline for each technical feature

3 Task Assignment

name	basic task	feature task
Ziang Liu	OpenGL	Crepuscular ray
Yihao Zhou	UI	HDR
Yijia Zhang	Camera	Geometry shaders
Sijie Ding	Layout	Shadow mapping

Table 2: Task assignments of two stages.

4 Implementation Design

4.1 Crepuscular ray

Crepuscular rays are mostly caused by the scattering of light in the media it goes through. The daylight scattering model is:

$$L(s,\theta) = L_0 e^{-\beta_{ex} S} + \frac{1}{\beta_{ex}} E_{sun} \beta_{sc}(\theta)(1 - e^{-\beta_{ex} S})$$

where β_{ex} is the extinction constant, E_{sun} is the source illumination and β_{sc} is the angular scattering term. **To make things simpler, the two constants should be tested out through experiment while implementing. The scattering term will just use Rayleigh scattering at the beginning,** $\beta_{sc}(\theta) = \frac{3}{16\pi}(1 + \cos^2\theta)$

To account for the possible occlusion in the scene, the result from previous model will be transformed to:

$$L(s,\theta,\phi) = (1 - D(\phi))L(s,\theta)$$

where $D(\phi)$ is the combined attenuated sun-occluding objects' opacity for the view location. **It is not clear now if more complicated objects like clouds or bubbles will be added in the scene.(They depend on the time table) Right now only the influence of the sea surface itself is considered, making a difference between rays above and below the sea surface.**

Ignoring super sampling for now, only attenuation due to the view point position will be considered. So the final model will be:

$$L(s,\theta,\phi) = exposure \times decay \times L(s,\theta,\phi)$$

where *exposure* controls the overall light intensity and *decay* reflects the attenuation.

Reference: https://en.wikipedia.org/wiki/Sunbeam#Crepuscular_rays

4.2 HDR

One common method to implement HDR is tone mapping. We can allow for a much larger range of color values to render to, collecting a large range of dark and bright details of a scene, and at the end we transform all the HDR values back to the low dynamic range (LDR) of [0.0, 1.0].
Resource: https://learnopengl.com/Advanced-Lighting/HDR

4.3 Geometry shaders

We planned to use Blender or Maya to build a new 3D model to simulate the environment of the undersea situation, or use an existed model from open online resources. The model will contain creatures and plants undersea. The creatures may include seaweeds and something else. Also, we will simulate the wave over the surface of the sea. For texturing, a special one is that we will use translucent texture for the sea water. We also want to simulate the brightness and darkness change through the water. And we will see if that will require a change of texturing of water and other objects. A obj. or other possibly required file will be outputted and will be able to be used in following process. For the implement of Geometry Shader, we will use OpenGL as tool. According to documents of OPENGL Geometry Shader.
Reference: https://www.khronos.org/opengl/wiki/Geometry_Shader

4.4 Shadow mapping

First, render the scene from the point of view of light with only depth of each fragment is computed. Then, we render the scene as usual, but do an extra test to see if the current fragment is in the shadow. The test is very simple. If the current sample is further from the light than the shadow map at the same point, this means that the scene involves an object that is closer to the light. In other words, the current fragment is in the shadow.
Resource: http://www.opengl-tutorial.org/intermediate-tutorials/tutorial-16-shadow-mapping/