Project Proposal: Real Time Raytracer

Team: Patrick Cheng (pc2720)
Charles Pan (cp3723)
Nick Li (ql2015)

For this project, we seek to improve and expand on the raytracer program by extending additional functionality to the program.

Task 1: Adding Camera System (4 %)

The first task will consist of refining the camera movement functionality. The camera will have a 'free move' mode where it will be able to move forward, backward, left, and right with the 'WASD' keys. It will also be able to rotate up, down, left, and right with the arrow keys. There will also be a function that will allow the user to select an object in the scene as a target. This will move and focus the camera on the object.

Task 2: Object Class & Interaction (12 %)

We plan to implement classes and functions to generate several object forms in order to organize and optimize object and scene generation. There will be key enabled functions which will allow the user to transform the objects in real time like **scaling, translating, and rotating.**

For class interaction, we plan to implement the functionality to change object's coordinates, shape, texture in real time based on the environment.

Objects(vertex, UV, indice, texture coordinate): torus, cylinder, cone, capsule)

Task 3: Lighting(16 %)

We plan to implement light classes for different types of light sources such as directional lights, spot lights, etc. The light class will be customizable with adjustable parameters such as **position, angle, size, light color**, etc.

lights(area light, spot light)

Task 4 : Ray Tracing (16%)

Ray tracing implementation with the use of ray shader, implementing effects like shadow map, reflection, refraction, etc

Effects(shadow, reflection, refraction)

Task 5 : Spatial Data Structure(20 %)

We plan to optimize the ray tracer's computation process with spatial data structure by eliminating unnecessary intersection checks. We may attempt to try several different spatial data structures to choose a best fit for this project.

Task 6: Screen-space Methods (12 %)

This part extends the deferred shading pipeline that enables more post-processing effects to be applied on the quad texture such as **ambient occlusion, exposure, vignette, and etc**, with customizable parameters.

Task 7: Particle System (20 %)

Implementation of Particle System that is similar to Unity Engine. Basic 2d quad with particle texture. **Particle emitter and generator**