Project 5: Global Illumination

Name: Minsuk Kim(m.kim)
Instructor: Dr. Gary Herron
Class: CS562
Semester: Spring 2022
Date: 4/22/2022

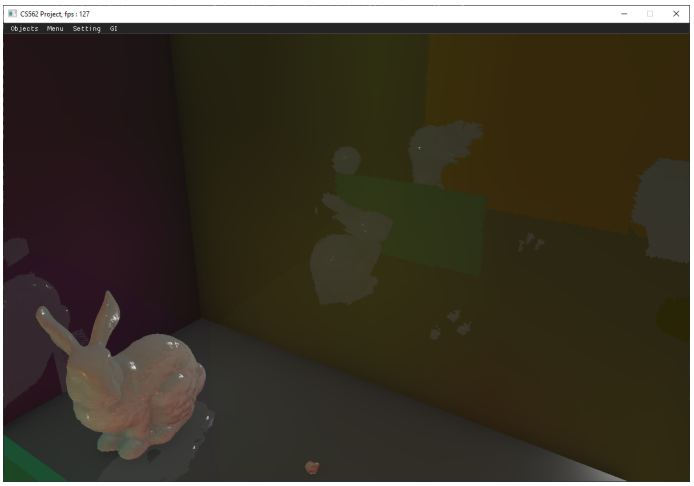


Table of Contents

Introduction **2**
Overview 2
Implementation 3

Result Images **4**
Images with diffuse reflection 5
Images with glossy reflection 6

Notes **7**
Octahedral mapping 7
Ray tracing 8

Introduction

Overview

The main purpose of the project is to implement real-time global illumination with pre-computed light probes for the static environment. Global illumination is lighting techniques for indirect lighting like reflection, refraction, and shadows. This report will cover just the reflection and refraction, diffuse lighting from the global illumination.

In this project, I will cover the pre-computed light probes for global illumination. The program will have three passes that involve this project: computing light probes on cube map texture pass, octahedral mapping pass, pre-filter irradiance map pass, and lighting calculation pass. The computing light probes on the cube map texture pass store all objects that light probes can see in the cube map texture. The octahedral mapping pass converts the cube map texture onto a two-dimensional texture using the octahedral projection. The pre-filter irradiance map filters distance, and radiance to generate the irradiance map and shadow variance map. The lighting calculation pass uses the pre-computed light probes data to calculate the diffuse and glossy reflection.

I started the project with Vulkan API and GLFW. I also used an external library Assimp that read the object files.

Implementation

The project follows the part of a paper called ‘Real-Time Global Illumination using Precomputed Light Field Probes’ from NVIDIA. The rendering sequence for this project will be the following: generating light probes cube map textures, using octahedral mapping to map the cube map textures to 2D textures, pre-filter the radiance and distance texture to get the irradiance and variance distance map. apply diffuse and glossy reflection in the final lighting result.

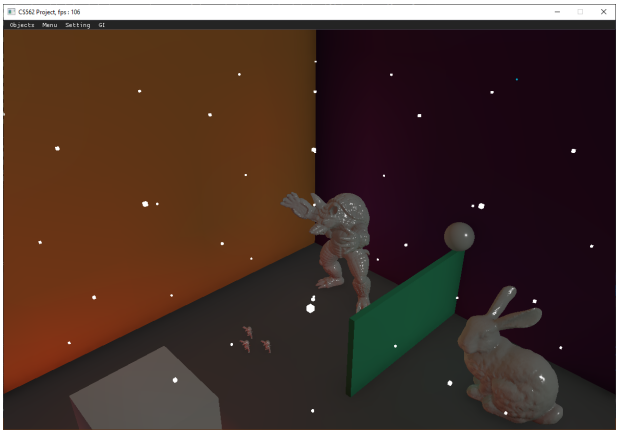
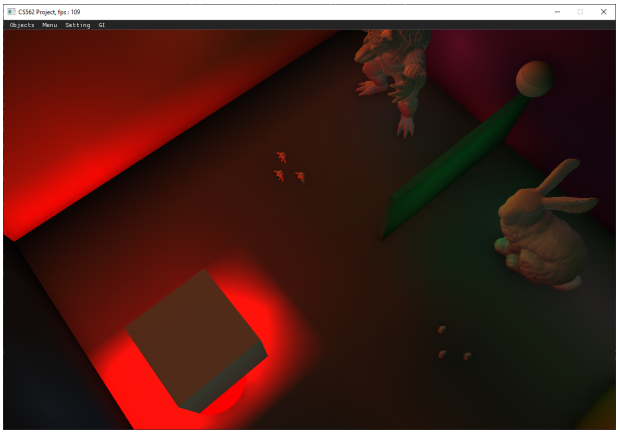
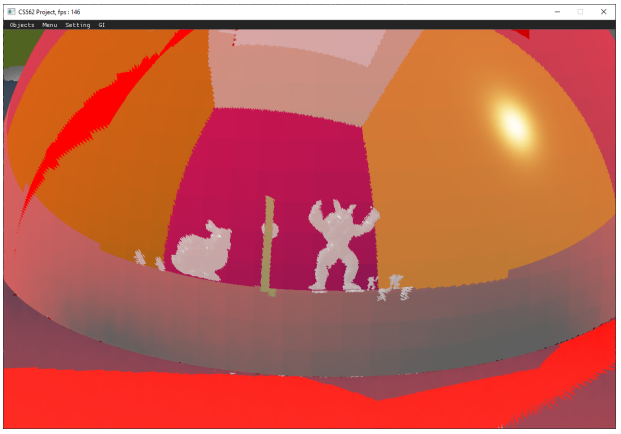
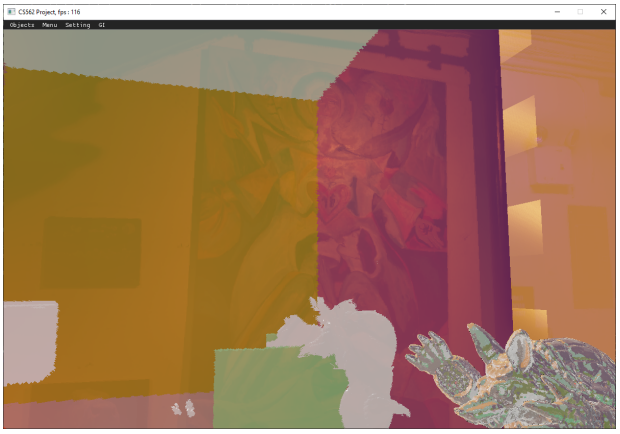
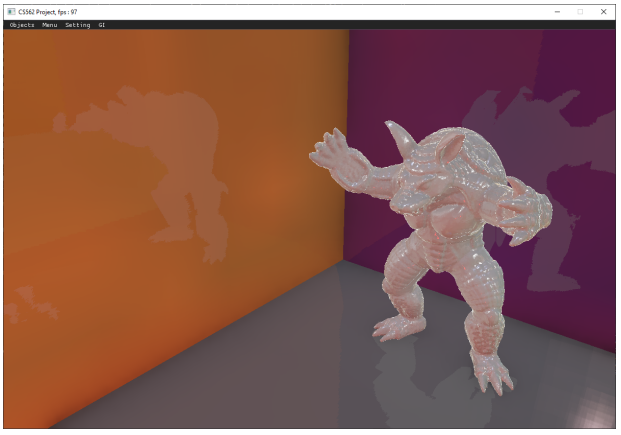
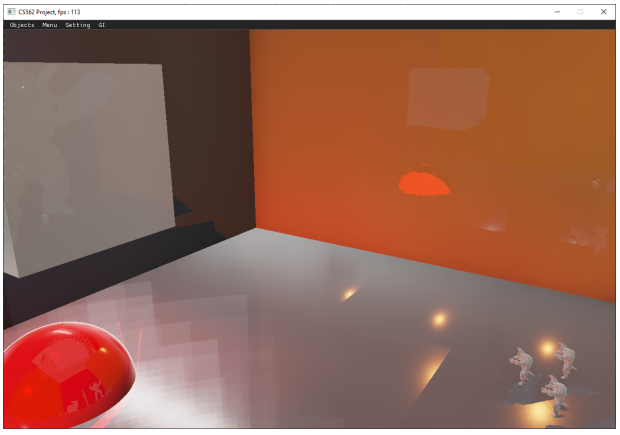
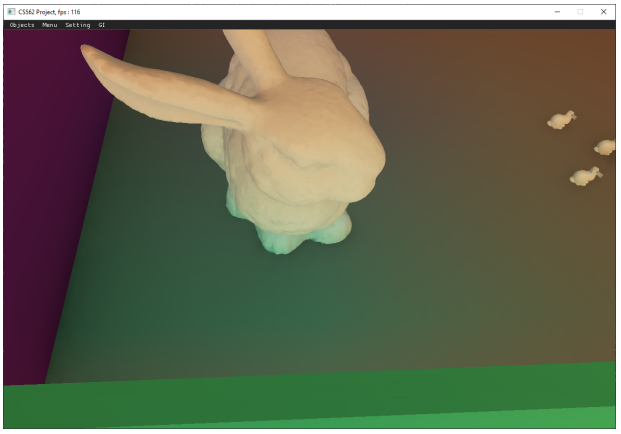
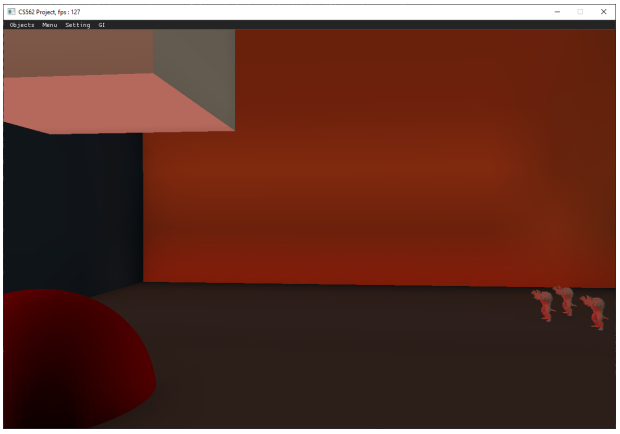
First, the program will generate a cube map texture for each light probe. The program stores the radiance, distance, and normal onto the cube map texture in the perspective of light probes. The program will iterate through all objects and project all geometry with the light probe perspective with the geometry shader.

After storing all surrounding information into each light probe's cube map texture, each light probe converts the cube map texture to a two-dimensional texture using the octahedral mapping. The program filters the radiance and distance map. The program uses compute shaders for calculating blur-filtered irradiance and distance maps.

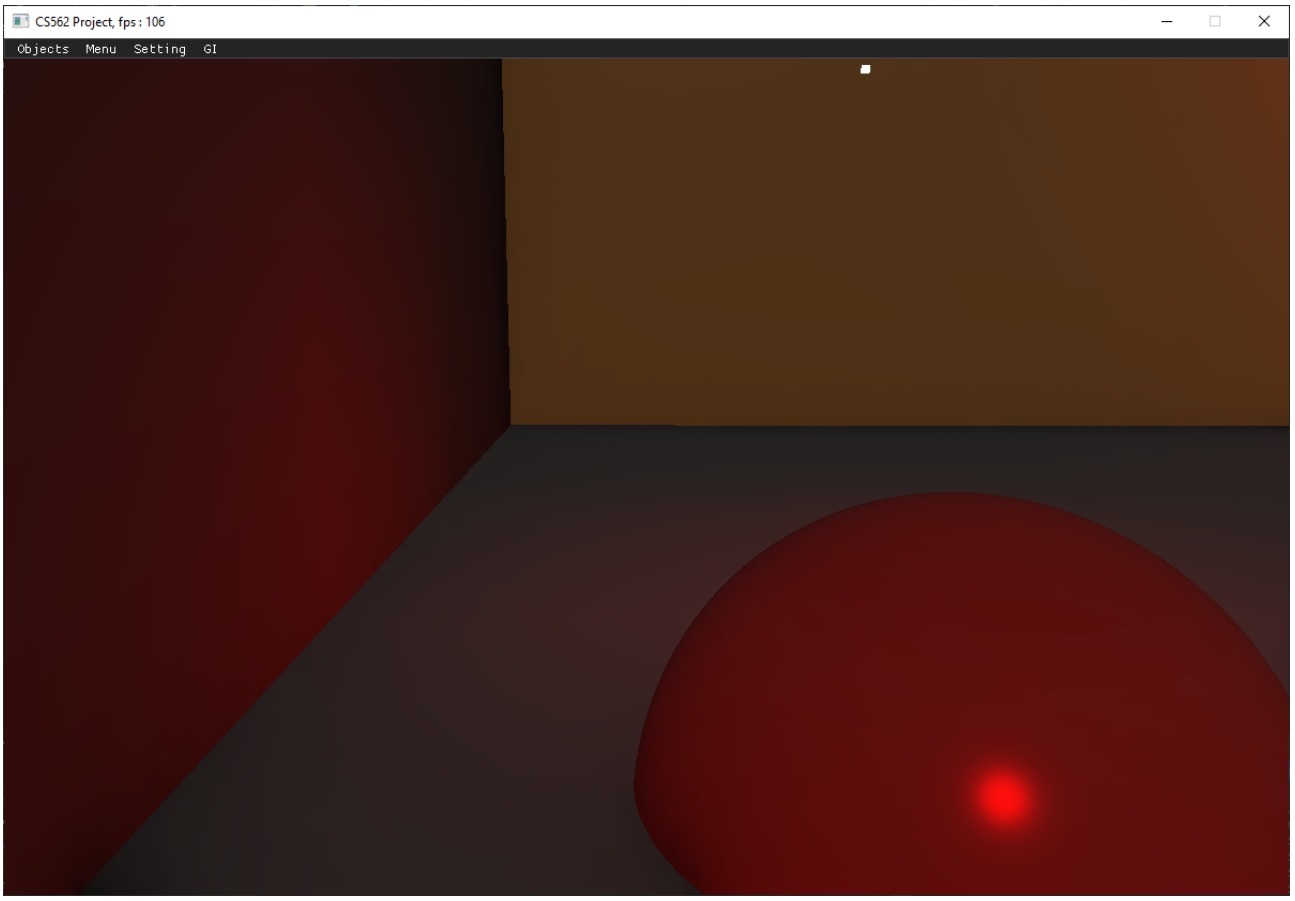
The final step is applying the diffuse and glossy reflection to the final scene. The final shader calculates the glossy reflection with the raymarching and the diffuse reflection with the irradiance map.

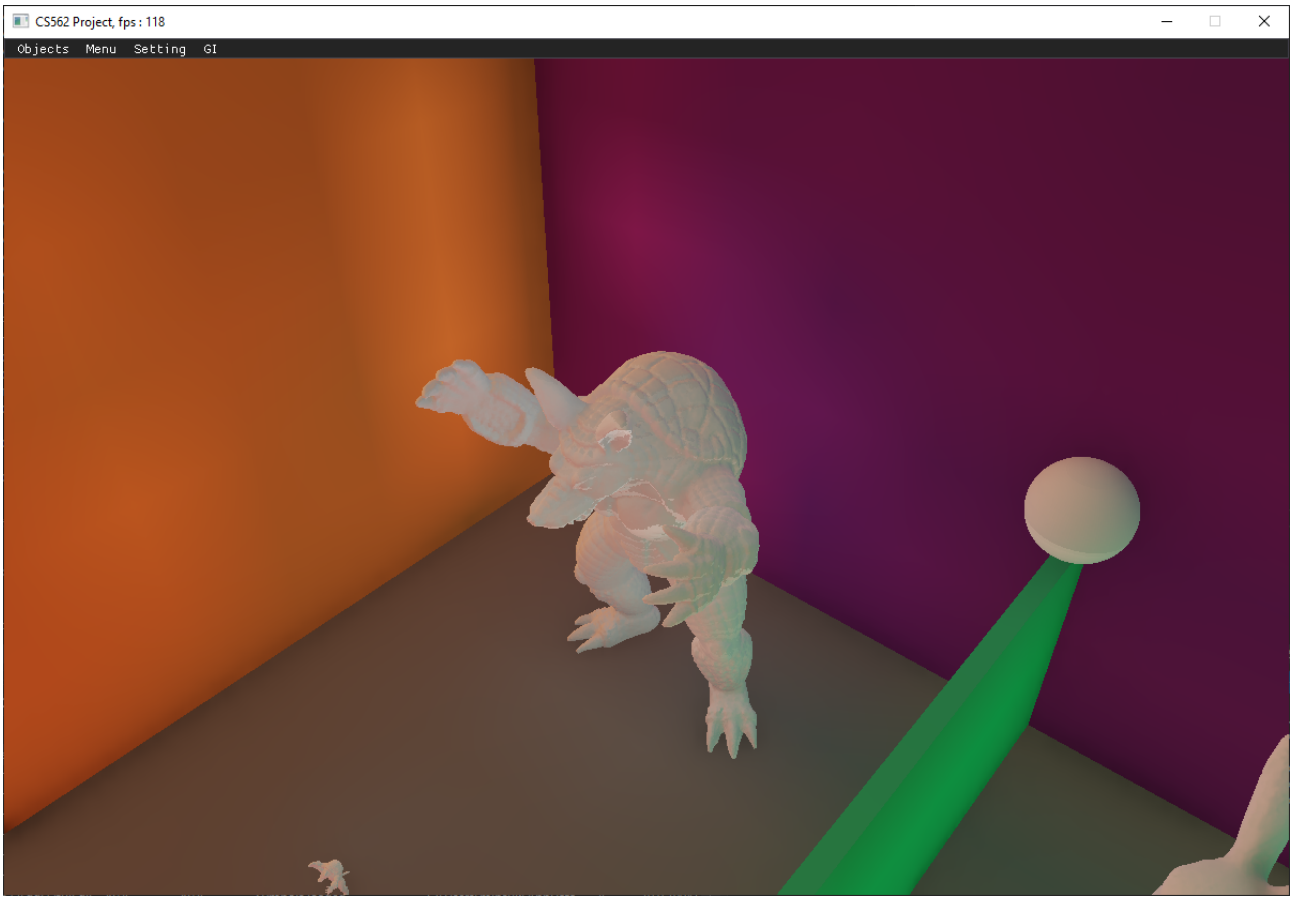
Result Images

some sample images

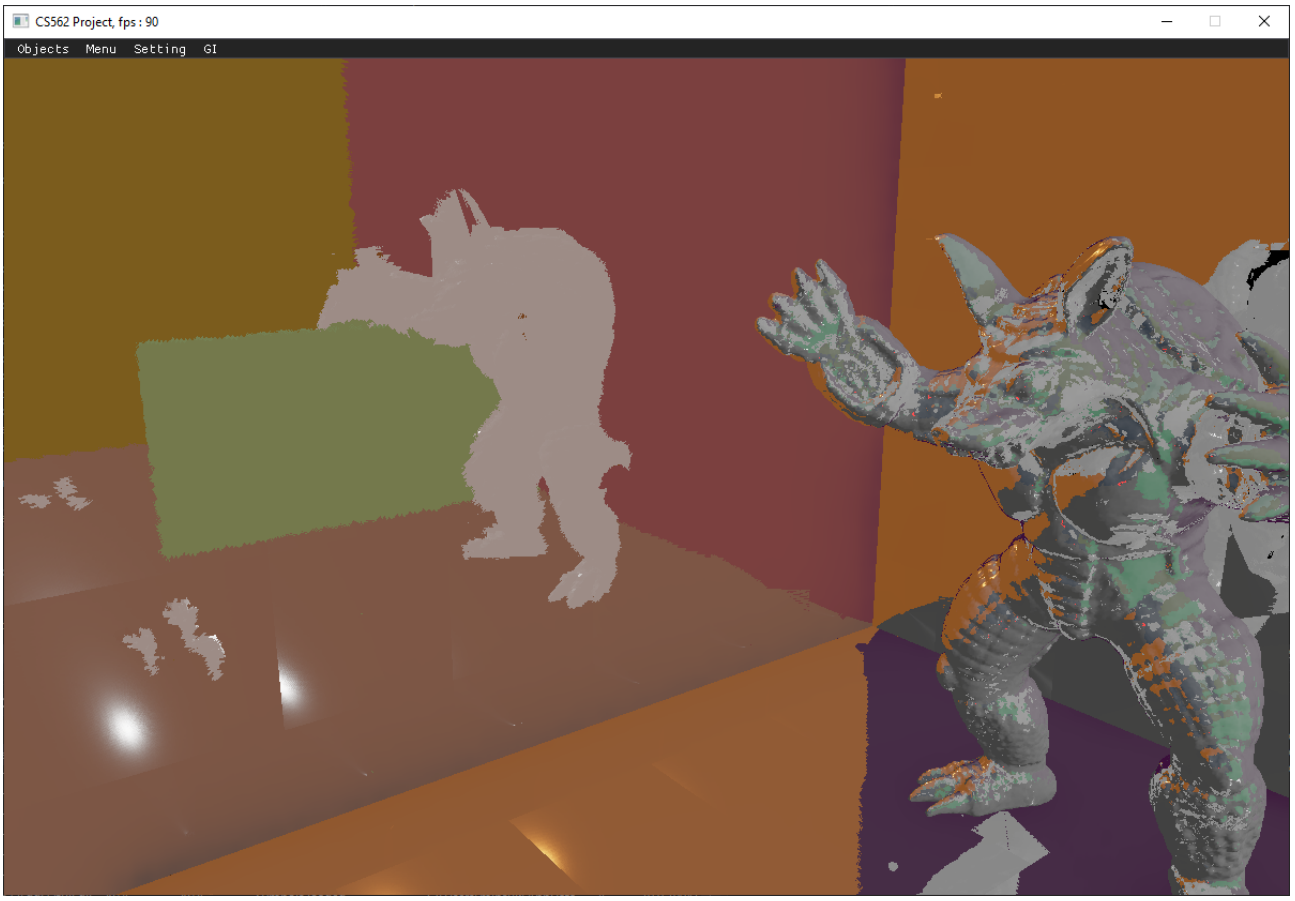









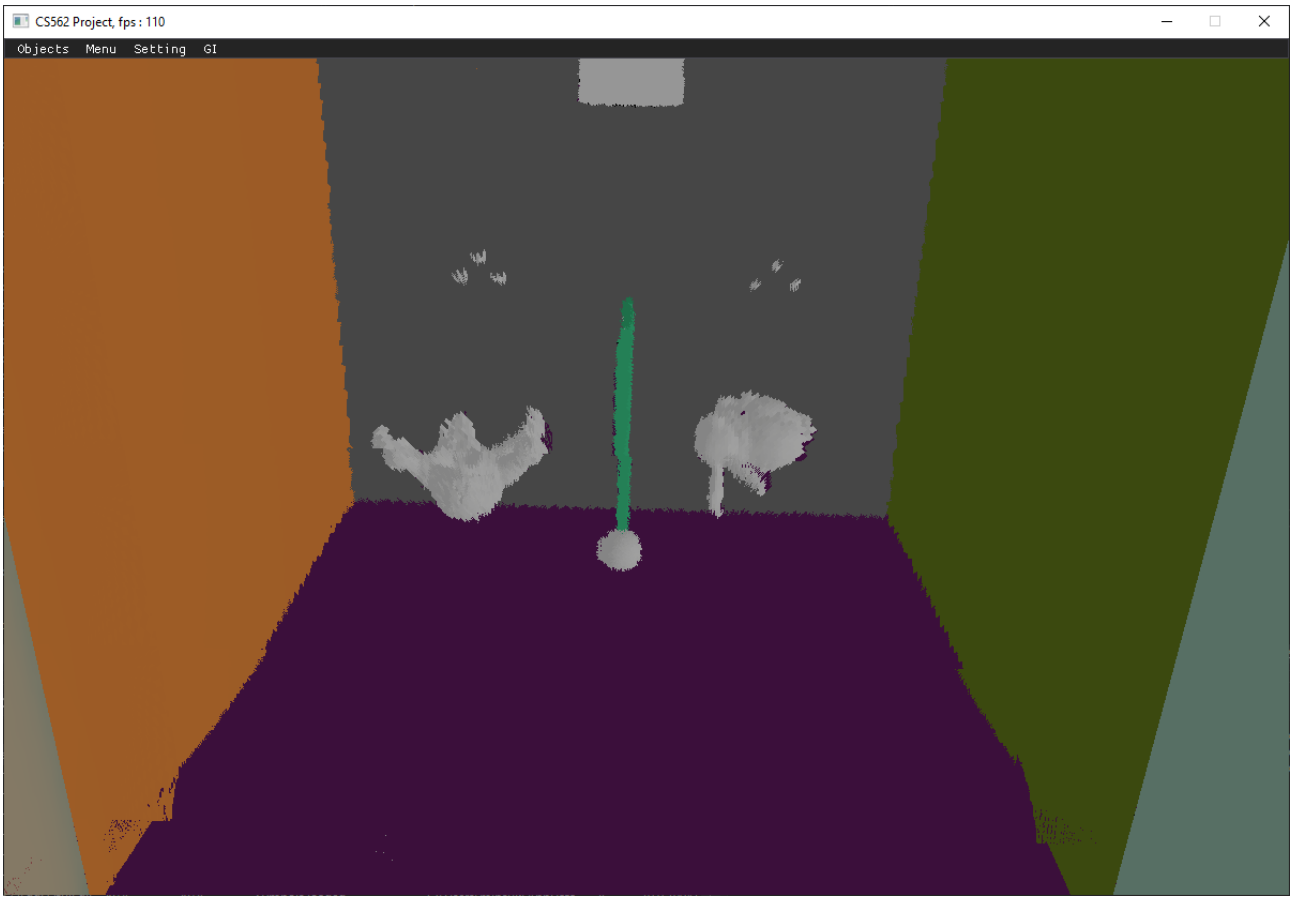
Images with diffuse reflection





Images with glossy reflection





Notes

Octahedral mapping

In this project, the program used octahedral mapping to store the cube map texture data to two-dimensional data for memory efficiency. The octahedral mapping is done by following:

$$UV = \frac{V.xy}{|V.x|+|V.y|+|V.z|}$$

$$if\ V.z < 0\ then,\ UV = (1.0 - |UV.yx|) * sign(UV.xy)$$

$$UV = 0.5 + UV * 0.5$$

, where UV is octahedral space texture coordinate and V is a three-dimensional vector.

We can unmapped the octahedral space two-dimensional vector three-dimensional coordinates with :

$$V.xy = UV * 2 - 1$$

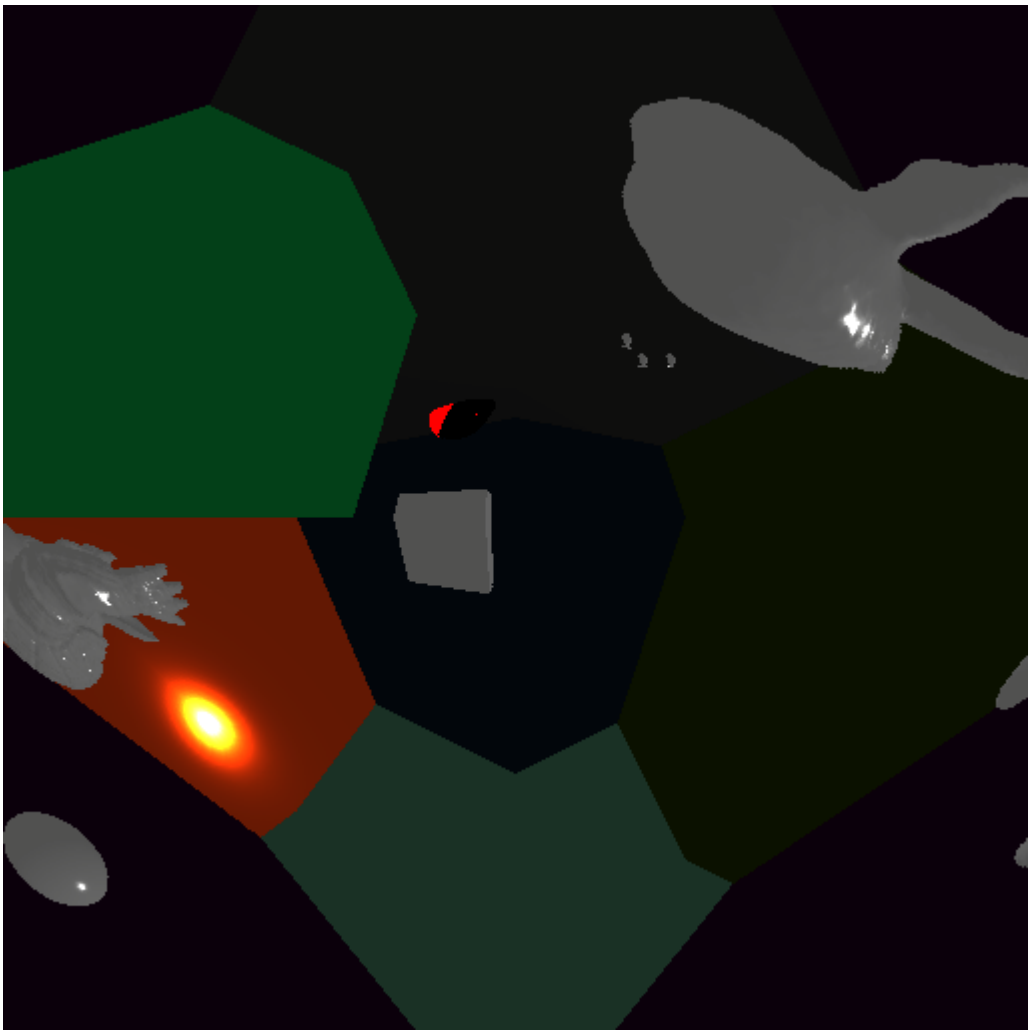
$$V.z = 1 - |UV.x| - |UV.y|$$

$$if\ V.z < 0\ then,\ V.xy = sign(V.xy) * (1 - |UV.yx|)$$

, where UV is octahedral space texture coordinate and V is a three-dimensional vector.

The program also uses the octahedral mapping in ray casting by projecting the ray line segments onto octahedral mapping.

the picture below shows the two-dimensional texture that mapped onto octahedral space



Ray tracing

To achieve the glossy reflection, I used the ray marching from each light probe to detect the indirect lighting coming from the surrounding object. For this project, I just do the perfect mirror for the glossy reflection. After computing the reflective vector of the surface, I choose the 8 cubic coordinate light probes that can wrap the position with a cubic cage. With the 8 light probes, I iterate through all light probes to check the ray trace results. When one of the probes detects the ray collision, I compute the color of the hit location with the result of ray tracing.

For each light probe ray tracing, I project the ray line segments (with the tiny time difference) onto octahedral space and see the distance data of light probes and see if the distance between the line segments is different or not. If the line segments between the two line segments, the program will assume the ray detects a collision on it and return the result.