CSE 472 Final Project

The aim of this project was to demonstrate the capabilities of the Ray Marching algorithm. Ray Marching is a ray intersection algorithm that defines objects in terms of their distance from any given point. For example, a sphere would have a distance from a point, p, of `length(p-sphere.center)-sphere.radius`. You can easily compose object distances by computing each of them individually and taking the minimum. The algorithm works by taking a ray and marching it in a direction by a specified distance at each step. As long as the distance is less than the distance of the closest object, the ray will not overshoot. By using the scene distance, we can tell how far to march the ray at each step. Once the distance from the scene is below a certain threshold, we consider the ray to have "collided". This is generally slower than exact ray intersection calculations, but it is more flexible because it is easier to define distances from objects than intersection with rays.

In addition, it is relatively simple to apply effects to objects by operating on their distances. For example, you can "melt" objects into one another. This is demonstrated with the "472" text and the blue cube and sphere. One simple operation is to subtract a constant from the object's distance, and it will appear rounded. This is demonstrated with the red box. Lastly, you can introduce "fake" soft shadows by tracking the minimum distance from the scene when casting a shadow feeler. The smaller the distance, the darker the area should appear.

This project also allows for modeling the scene outside the shader. The scene definition can be seen in "src/scene.rs". Objects and lights containing all their properties can be added to the shader via uniforms every frame. Any number of objects and lights can be added as long as the buffer size in the shader is large enough.

Extra Credit

As stated in my project proposal, I believe I meet both extra credit features. The shader implements Blinn-Phong lighting and allows for modeling the scene outside the shader so objects can be added dynamically and manipulated in real time.

Project Credits

- Ray marching explanation and motivation for this project: https://www.youtube.com/watch?v=svLzmFuSBhk
- Shader tutorial of Ray Marching algorithm: https://youtu.be/PGtv-dBi2wE
- Distance functions for primitives and operations: https://iquilezles.org/articles/distfunctions/
- Past assignments in this class referenced for doing Blinn-Phong correctly and working with shaders.