2016400348

UTAH TEAPOT

First of all, I implemented some headers files that are vec3.h, sphere.h, ray.h, color.h and constant.h.

- **vec3.h**: It is a class that has 3 points x,y,z that represent the coordinate system in 3D. Also, it has some helper methods for vector operations such as addition, multiplication or dot product.
- **triangle.h**: It is a class that has a center as vec3, radius as double and color as vec3. Also, it has a method called findIntersection that finds whether ray intersects itself within any t value.
- **ray.h**: It is a class that has an origin point as vec3 and direction point as vec3.Also, it has some helper methods for calculating intersection points.
- **color.h**: It has some helper methods for color calculation and output process.
- **constants.h**: It has some constant values and header files.

In the raytracer.cpp file, after handling the input parse processes, I set the **height** and **width** as 1000 (resolution is 1000X1000 pixels).Then, I set the simple camera view. I couldn't set the perspective using previous project implementations, so I used the camera options from https://www.scratchapixel.com by creating some brute force matrix definitions. (https://www.scratchapixel.com/code.php?id=35&origin=/lessons/advanced-rendering/bezier-curve-rendering-utah-teapot)

First of all, I need to create points for creating any small parts from patches and divide them into triangle faces. To create points, I defined the control points with given data. Then, I implemented some bezier curves methods to create points by giving control points as parameters. Also, I calculated the vertex normals of the triangles. I split the patches into 64 small parts(8*8) and this gives me 81 points with control points. Also, each part is composed of two triangles and that means there are 128 triangles. For each patch, I created 128 triangles. Using the intersection method of triangle class, I found the closest t values intersect with ray. Also, I used the normal vector of the vertices for shadowing.

How to Run

- You can run the following command -> **make**

Manual

- Run the following commands
- c++ raytracer.cpp -o raytracer.out -std=c++17
- ./raytracer.out >> scene.ppm
- open scene.ppm