COL781: Computer Graphics
Assignment 1

Suyash Agrawal (2015CS10262)
Aman Agrawal (2015CS10210)

March 13, 2019

1 Introduction

In this assignment we implement a ray-tracing module. We have implemented a recursive ray tracing algorithm principles as defined in the literature.

2 Models

We have worked with the following models:

1. Sphere
2. Quadric
3. Plane
4. Polygon
5. Triangle
6. Collection
7. Box

All of these models, can be defined in their own local co-ordinate systems and need only the parameters required to define them completely.

3 Lights

We have a functionality of specifying lights in the world co-ordinate system, and the color of the light emitted by them. Presently, all lights are point source.

4 Camera

We presently have a support for specifying a single camera, and a transformation matrix, that specifies the lookat direction and position of the camera.

5 Texture and Materials

We have a support of specifying texture for Sphere and Triangle. We find out the value of u and v for these models and map them to the texture image, to get the final color at the location in a texture map.

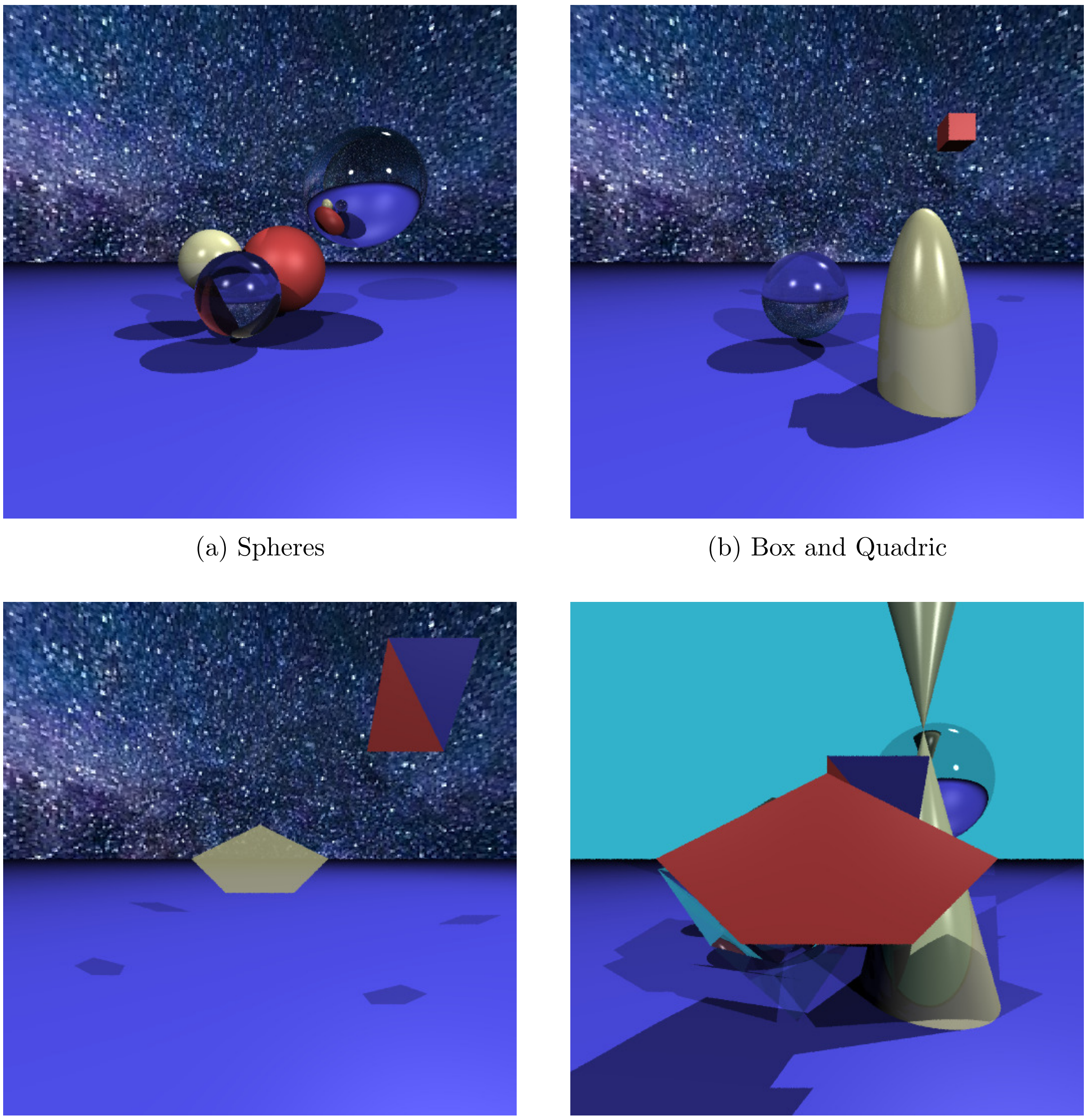
We also, have a support of specifying a class of materials each with their own parameters, and each model object can be assigned a texture or a material.

6 Ray Tracing

Finally, we perform a recursive ray tracing that renders the scene. Some of the samples of the rendered scenes are in fig. 1

7 OpenGL Simulation

For OpenGL simulation we had to create new classes for our models. We created classes for Box, Sphere, Lights, Lines, Mesh and Camera. For the simulation we first re-drew our scene using the input json file and parsing them in our OpenGL model classes and then we calculated all the intersection lines using our ray traces code and drew these lines in OpenGL window. This gave us the required simulation of ray tracer for a given point in image. Some of the things like refraction and shadows were missing from our rendered scene since OpenGL uses rasterization instead of ray tracing and it is really tough to give shadows and refraction in rasterization. We also made the camera system movable, thus giving us freedom to roam around the scene and explore the whole environment. See fig. 2 for an example.


(a) Spheres (b) Box and Quadric

(c) Polygon and Collection (d) Complete Models

Figure 1: Results of Ray-Tracing with various models, and textures

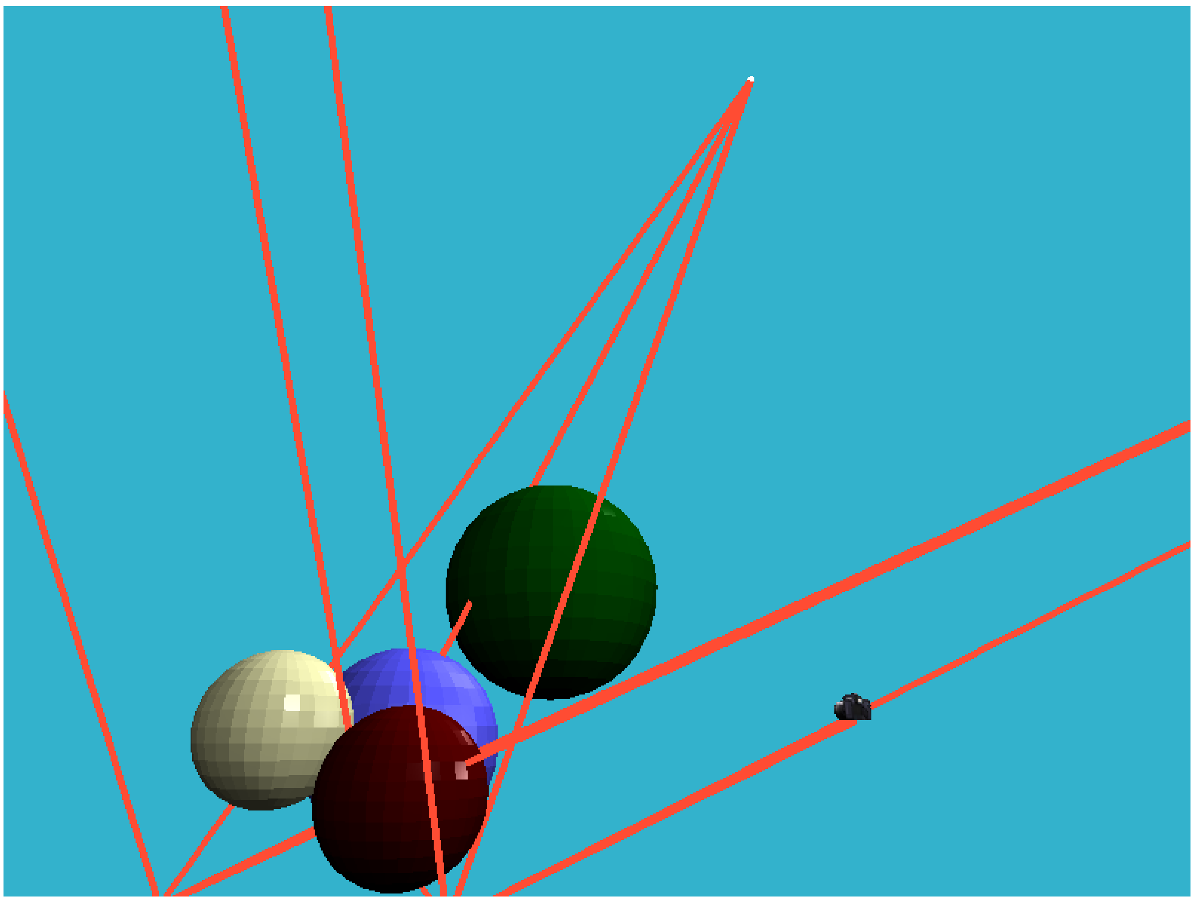


Figure 2: Results of OpenGL simulation