Matthew Martin
998354224
mdmartin@ucdavis.edu

ECS 175, Winter 2017
March 20, 2017

Project 5: A Simple Ray Tracer

Compilation

A makefile has been included. To compile, simply run “make”.

The executable produced will be named “project5”. This executable can, optionally, be run with a command line argument which specifies the OpenGL window’s size. For example, “project5 800” will run the program with a window width and height of 800. Default window size is 300.

Implementation Locations

CVM:

viewport.cpp • lines 582 - 603

Ray Tracing:

viewport.cpp • lines 474 - 580

Usage

General:

The application runs using a command line interface, and the window opened by OpenGL. As commands are entered into the command line, changes will be reflected in the viewing windows. 15 commands are supported, each of which can be accessed by typing the command name followed by arguments. Each command is detailed below, in the “commands” section. A full list of command names that map to specific commands can be found in lines 56 - 138 of commandHandler.h.

I/O File Format:

Saved scene files are stored in the following way. Firstly, the data about the scene-scope data is stored. The background color, from point, at point, up vector, viewing angle, and ambient light intensity appear. Next, the number of lights in the scene, followed by information about each light. For each light: color, intensity, position.

Next the number of shapes in the scene. For each shape, the color, reflection fraction, refraction fraction, refractive index, and phong exponent are stored. Shapes can be stored in 2 different formats: implicit or surface-defined.
For implicit shapes, the coefficient values (c_200 to c_000) are stored next.
For surface-defined shapes, the shape point, and surface indices are stored next.

An example file is shown in the figure below (sample scenes have been included, in the format *.data).

```
0 0 0
126 42 36
0 0 0
0 0 1
30
0.2

2
1 1 1
500
-200 200 200
1 1 1
600
800 0 0

2

SURFACE_SHAPE
0 1 1
0.6
0.1
1.5
4
8
-950 -350 -350
-950 150 -350
-450 -350 -350
-450 150 -350
-950 -350 150
-950 150 150
-450 -350 150
-450 150 150
12
1 3 2
3 4 2
5 6 8
8 7 5
1 5 3
5 7 3
4 6 2
4 8 6
2 6 1
5 1 6
4 3 8
3 7 8
```

Commands

The application supports 15 commands, each of which take specific arguments. Arguments can be entered separated by whitespace, commas, parenthesis, or other common separators. Some of these commands are overloaded (can take a variable number of arguments). All commands are outlined below.

Example:
Command Name <>; <arg1_1>; <arg2_1, arg2_2>: "command_to_enter"
A brief description of what the command does for each overloaded version.

Example usages:
>command_to_enter
>command_to_enter arg1_1
>command_to_enter arg2_1, arg2_2

Curve Loading/ Saving

Load <filename>: "ld"
Loads the shape(s) and light(s) from the passed file into the scene.

Save <>; <filename>: "sv"
Saves the shape(s) and light(s) from the scene to the passed file, overwriting the contents of the file. Overloaded, with no arguments, to save to the last loaded/saved file.

Creation/ Deletion

Add Cube <x, y, z, edge_length, r, g, b, refl, refr, refr_index, phong_exp>: "cube"
Adds a new cube to the scene.

Add Sphere <x, y, z, radius, r, g, b, refl, refr, refr_index, phong_exp>: "sphere"
Adds a new cube to the scene.

Add Implicit <c200, c020, c002, c110, c101, c011, c100, c010, c001, c000, r, g, b, refl, refr, refr_index, phong_exp>: "implicit"
Adds a new implicit object to the scene.

Add Light <r, g, b, intensity, x, y, z>: "light"
Adds a light of the specified color and intensity at the specified position.

Delete <index>: “del”

Deletes the specified curve. This shifts the index of all later shapes down by 1.

Delete Light <index>: “dlight”

Deletes the specified light. Shifts the index of all later lights down by 1.

CVM Movement

Some CVM movement commands take a “direction” argument. This is a string specifying the direction {up, down, left, right, forward, backward}.

Move Camera <direction, magnitude>: “move”

Moves the camera in the specified direction. Does not turn the camera.

Move From Point <direction, magnitude>: “mf”

Moves the “from point” in the specified direction. Results in the camera direction turning.

Move At Point <direction, magnitude>: “ma”

Moves the “at point” in the specified direction. Results in the camera direction turning.

Set Viewing Angle <alpha>: “va”

Sets the viewing angle to the specified value (in degrees). Default viewing angle is 30 degrees.

Set From Point <>; <x, y, z>: “ff”

Sets the from point to the specified position. Overloaded to print the current from point with to arguments.

Set At Point <>; <x, y, z>: “aa”

Sets the at point to the specified position. Overloaded to print the current at point with to arguments.

Misc.

Quit <>: “q”

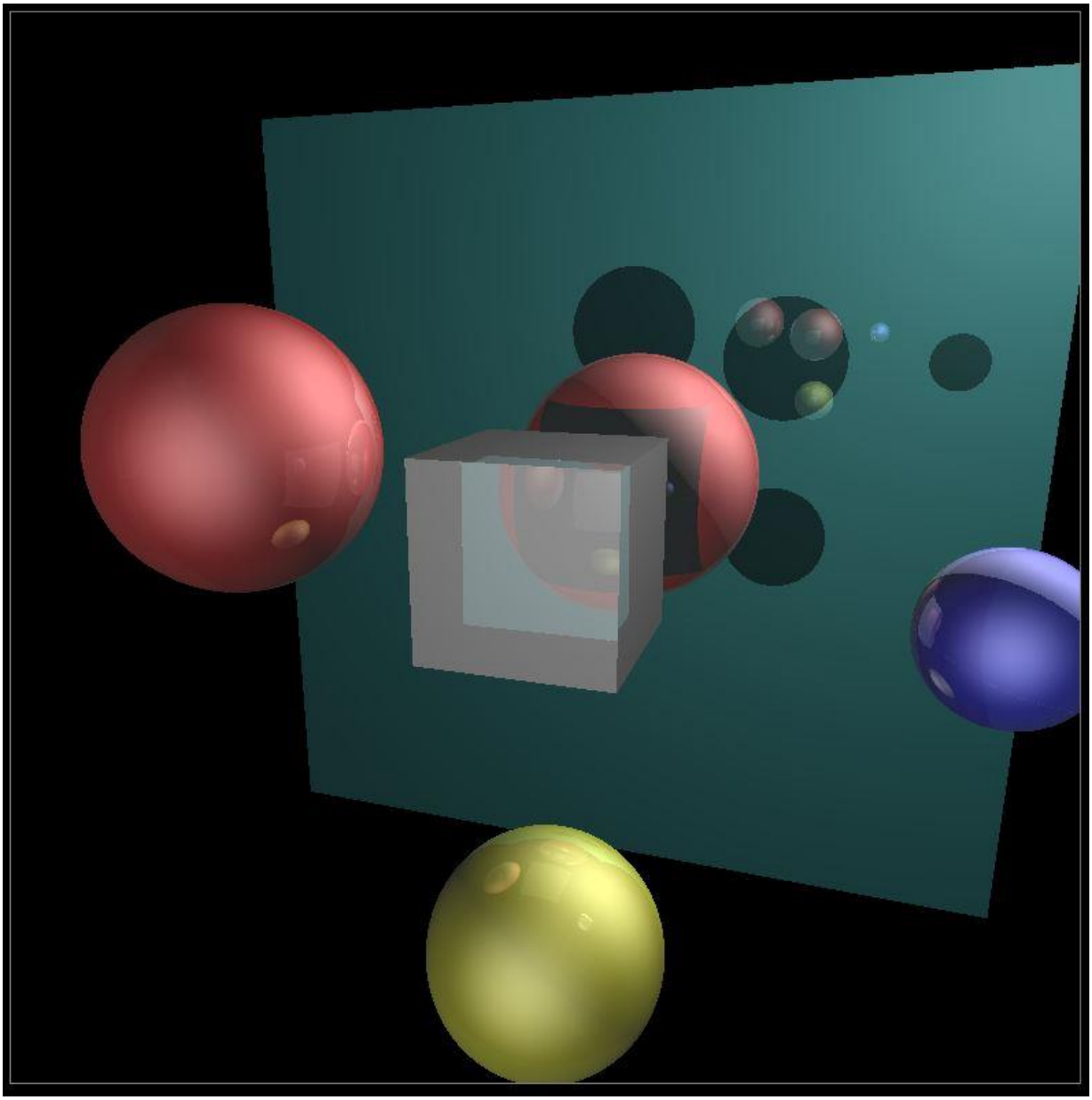
Quits the application. Asks if the user would like to save to a previously loaded/saved file before exiting.

Sample Scene

The program can take a while to render a large scene. For this reason, a screenshot of a sample scene has been included here. Note that the sample scene shows most aspects of the ray tracer (reflection, refraction, shadow feelers, etc.).

The scene consists of 6 objects: 4 opaque colored balls, a transparent cube, and a large mirrored cube (blue) in the back. The scene has 2 light sources: (both white) to the left of the viewer, and above-and-right of the viewer.

This scene has been included in the submission as “scene3.data”.



Extra Credit

1. The program implements a true camera viewing model (not simplified), and CVM movement commands.
2. The program handles intersecting objects (treating the refractive index of intersecting regions as the average of all objects that touch the region).