CSCE 441 - Computer Graphics

Programming Assignment 5

Deadline: Nov. 25th (11:59 pm)

1 Goal

The goal of this assignment is to write a ray tracer.

2 Starter Code

The starter code can be downloaded from here.

3 Task 1

Download the code and run it. You should be able to see a black screen. Make sure you write your name in the appropriate place in the code, so it shows up at the top of the window. Here is a brief explanation of the starter code:

- There is only one folder in the package, which includes all the source files.
- The main function in "main.cpp" first calls the `Init` function.
- The `Init` function performs the followings:
  - First, it initializes the window.
  - Then it creates an instance of the `Scene` class. This class is a container that holds all the scene information including the shapes and light sources.
  - Next it create an instance of the `Camera` class. Currently, only the resolution of the image is passed to the camera. However, you need to modify this and pass all the necessary information.
  - Then we call a function of the `Camera` class to take a picture of the scene using ray tracing. This is where the main loop of the ray tracer should be implemented in. This function takes the scene as the input and takes a picture of it. The picture is stored in a private variable called `renderedImage`.
  - We then have the camera return the rendered image and set it to the `frameBuffer` through `memcpy`.

Note that, you do not need to follow this starter code structure. If you want to write your ray tracer differently, feel free to do so. The starter code is just provided to help you.

4 Task 2

In this part, you will be implementing a ray tracer. Your ray tracer should support the followings:

- **Shadows:** You should compute shadow rays to all the light sources.
- **Reflections:** Your code should recursively call the reflection ray, i.e., you shouldn't hard code the reflection calls. Set the level of recursion to at least 4. DO NOT HARD CODE FOUR REFLECTION CALLS.
- **Lighting:** You should compute basic lighting (ambient, diffiuse, specular) as explained in the shading lectures.

You are expected to demonstrate your ray tracer on the following scene. Here is the camera properties:

- Eye = (0.0, 0.0, 7.0)
- Look at = (0.0, 0.0, 0.0)
- Up = (0.0, 1.0, 0.0)
- FovY = 45
- Width Res = 1200
- Height Res = 800

Your scene should include two planes, four spheres, and two light sources as follows:

- Shapes
  - Sphere 1
    - Position = (-1.0, -0.7, 3.0)
    - Radius = 0.3
    - k_a = (0.1, 0.1, 0.1)
    - k_d = (0.2, 1.0, 0.2)
    - k_s = (1.0, 1.0, 1.0)
    - k_m = (0.0, 0.0, 0.0)
    - s = 100.0
  - Sphere 2
    - Position = (1.0, -0.5, 3.0)
    - Radius = 0.5
    - k_a = (0.1, 0.1, 0.1)
    - k_d = (0.0, 0.0, 1.0)
    - k_s = (1.0, 1.0, 1.0)
    - k_m = (0.0, 0.0, 0.0)
    - s = 10.0
  - Sphere 3 (reflective)
    - Position = (-1.0, 0.0, -0.0)
    - Radius = 1.0
    - k_a = (0.0, 0.0, 0.0)
    - k_d = (0.0, 0.0, 0.0)
    - k_s = (0.0, 0.0, 0.0)
    - k_m = (1.0, 1.0, 1.0)
    - s = 0.0
  - Sphere 4 (reflective)
    - Position = (1.0, 0.0, -1.0)
    - Radius = 1.0
    - k_a = (0.0, 0.0, 0.0)
    - k_d = (0.0, 0.0, 0.0)

        * k_s = (0.0, 0.0, 0.0)
        * k_m = (0.8, 0.8, 0.8)
        * s = 0.0
    - Plane 1
        * Center = (0.0, -1.0, 0.0)
        * Normal = (0.0, 1.0, 0.0)
        * k_a = (0.1, 0.1, 0.1)
        * k_d = (1.0, 1.0, 1.0)
        * k_s = (0.0, 0.0, 0.0)
        * k_m = (0.0, 0.0, 0.0)
        * s = 0.0
    - Plane 2
        * Center = (0.0, 0.0, -3.0)
        * Normal = (0.0, 0.0, 1.0)
        * k_a = (0.1, 0.1, 0.1)
        * k_d = (1.0, 1.0, 1.0)
        * k_s = (0.0, 0.0, 0.0)
        * k_m = (0.0, 0.0, 0.0)
        * s = 0.0

- Light
    - light 1
        * Position = (0.0, 3.0, -2.0)
        * Color = (0.2, 0.2, 0.2)
    - light 2
        * Position = (-2.0, 1.0, 4.0)
        * Color = (0.5, 0.5, 0.5)

Your final rendering should look exactly like the image in Fig. 1.

4.1 Steps

Here are the steps to take to properly implement the ray tracer:

- Fill in the `Camera` class to have the necessary member variables. Currently, the instance of the `Camera` class in the `Init` function of "main.cpp" is created with only width and height information. You should set it up using the camera parameters provided above.

- Set up the scene with the information provided above by filling in the `Shape` and `Light` vectors in the `Scene` class. This can be done in the constructor of the `Scene` class.

  To do this, you need to first add appropriate member variables to the `Shape`, `Sphere`, `Plane`, and `Light` classes. For example, `Light` class should at least have the position and color as member variables. Note that, `Shape` is the base class for the `Sphere` and the `Plane` and has access to all their member variables.

  Once all the classes are properly set up, you can create instances of Sphere, Plane, and Light classes based on the information provided above and then push them into the appropriate vector.

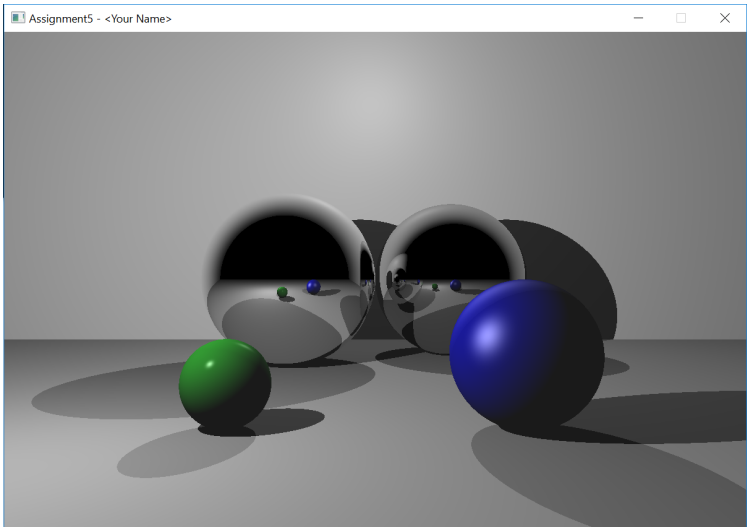

Figure 1: The result of rendering the scene.

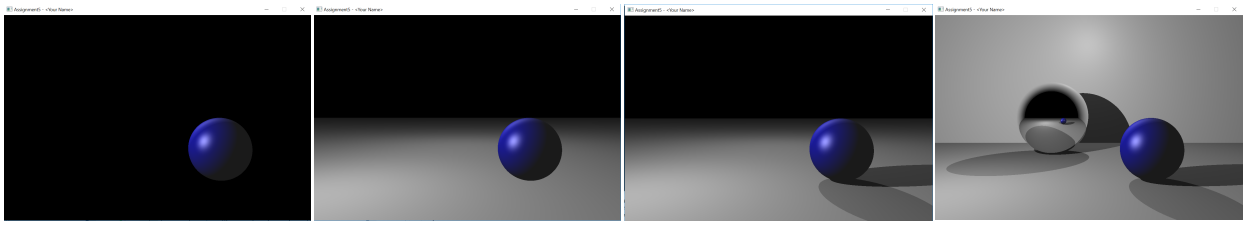

Figure 2: From left to right the outcome of steps 1, 2, 3, and 4 in Sec. 4.2.

- Next, you should fill in the `TakePicture` function of the camera class. In this function you should loop over all the pixels. For each pixel, you first create a ray pointing from the camera to the pixel. You then call a function to get the color of the ray. Most of the computations are done inside this recursive function. The returned color of the ray is then used to set the color of the `renderedImage` array at that pixel.

4.2 Suggestions

Since the ray tracer has several features, it would be better if you build your system incrementally. For example, you can do the following (see Fig. 2 for outcome of each step):

1. Render a sphere without any shadows or recursive calls.
2. Add a plane to the scene.
3. Shoot shadow rays to be able to render shadows.
4. Implement the recursive reflection call and add a reflective sphere to the scene to test it.

5 Bonus

- Add the ability to render triangles
- Add area light and render soft shadows

6 Deliverables

Please follow the instruction below or you may lose some points:

- You should include a README file that includes the parts that you were not able to implement, any extra part that you have implemented, or anything else that is notable.
- Your submission should contain folders "src" as well as "CMakeLists.txt" file. You should not include the "build" folder.
- Zip up the whole package and call it "Firstname_Lastname.zip". Note that the zip file should extract into a folder named "Firstname_Lastname". So you should first put your package in a folder called "Firstname_Lastname" and then zip it up.

7 Ruberic

Total credit: [150 points]

[20 points] - Camera set up and primary rays generated
[10 points] - Plane is intersected correctly
[15 points] - Sphere is intersected correctly
[20 points] - Depth test is done properly
[30 points] - Basic local lighting is computed
    [10 points] - Normals of spheres and planes computed correctly
    [10 points] - Light, view, and reflection vectors are computed correctly
    [10 points] - Shading is computed using Phong model
[25 points] - Shadow rays are calculated and incorporated
[30 points] - Reflection is supported
    [20 points] - Reflection is computed recursively
    [10 points] - Color of reflected vector is combine with local Phong shading

Extra credit: [15 points]

[5 points] - Triangle
[10 points] - Area light