CS3241 Computer Graphics (2023/2024 Semester 1)

Lab Assignment 4

Release Date: 27 October 2023, Friday

Submission Deadline: 13 November 2023, Monday, 11:59 PM

LEARNING OBJECTIVES

Implementing the Whitted Ray Tracing algorithm. After completing the programming assignment, you should have learned

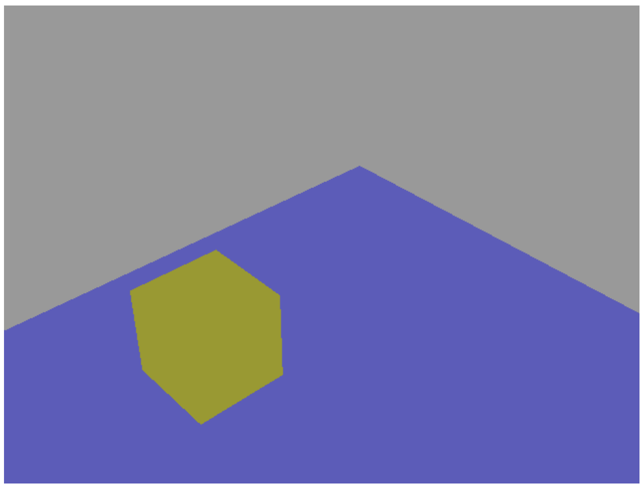
- how to compute ray intersection with some simple implicit-form surface primitive,
- how to do lighting computation,
- how to shoot shadow rays to generate shadows,
- how to spawn secondary rays,
- how to trace rays recursively, and
- how the Whitted Ray Tracing algorithm works.

TASKS

You are to complete an **incomplete C++ program** that implements the Whitted Ray Tracing algorithm. You have to complete the program according to the following requirements. There are altogether **three tasks** in the assignment.

From the **`Canvas > CS3241 > Files > Lab Assignments`** folder, download the ZIP file **`Lab4_todo_(*).zip`**.

A Visual Studio 2017 solution **`Main.sln`** (or Xcode project **`Main.xcodeproj`** on macOS) is provided for you to build the executable program. If you build and run the given incomplete program, it will produce an image as follows (in file **`out1.png`**).



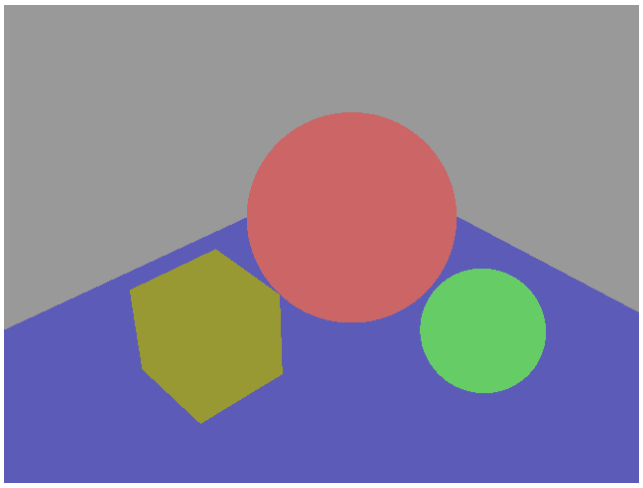
It shows an image of an unlit scene of 3 intersecting planes and a cube made of triangles. Two additional spheres are supposed to be in the image, but the ray-sphere intersection routine has not been implemented yet.

Task 1

You are to complete the **`Sphere::hit()`** and **`Sphere::shadowHit()`** functions in **`Sphere.cpp`** to compute ray-sphere intersection. You must write your code only in places marked "**`WRITE YOUR CODE HERE`**".

You can refer to **`Plane.{h, cpp}`** and **`Triangle.{h, cpp}`** to get some idea how to do it. Other relevant files to study are **`Vector3d.h`**, **`Ray.h`**, **`Surface.h`**, and **`Sphere.h`**.

For this task, you have to **submit** your completed **`Sphere.cpp`**, and the image generated, which should look like the following. You must name your image file **`img_spheres.png`**.



`img_spheres.png`

Task 2

You are to complete the **`Raytrace::TraceRay()`** function in **`Raytrace.cpp`** to perform the recursive ray tracing. You must write your code only in places marked "**`WRITE YOUR CODE HERE`**".

In this implementation, we are assuming that **all objects are opaque**. At each surface point intersected by the ray, the color result is computed using the formula

$$I = I_{local} + k_{rg}\, I_{reflected}$$

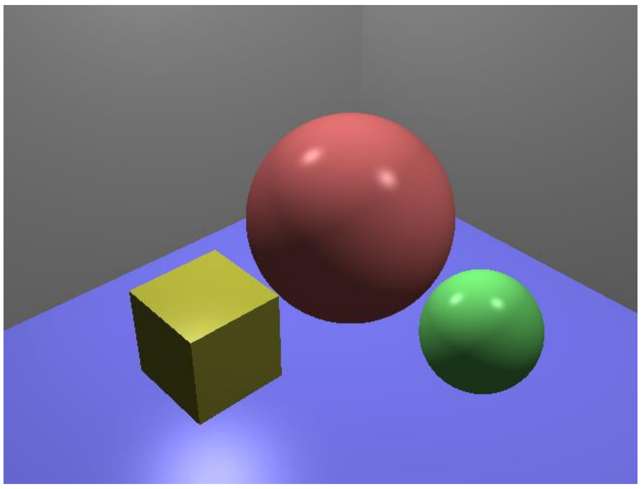
where

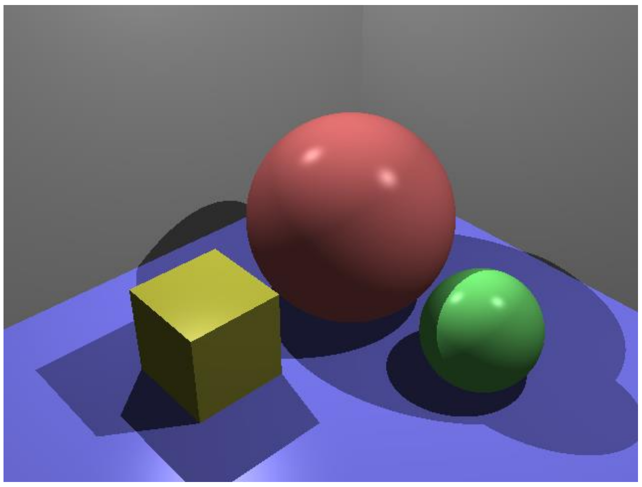
$$I_{local} = I_a k_a + \sum_{i=1}^{M} k_{i,shadow}\, I_{i,source}\left[k_d(\boldsymbol{N}\cdot\boldsymbol{L}_i) + k_r(\boldsymbol{R}_i\cdot\boldsymbol{V})^n\right]$$

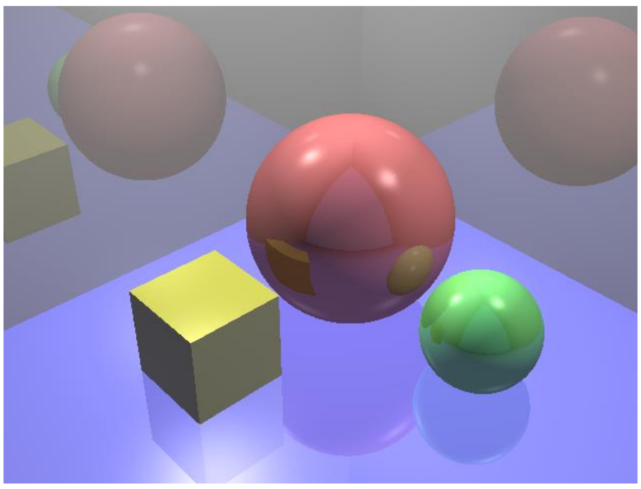
and *M* is the number of point light sources in the scene.

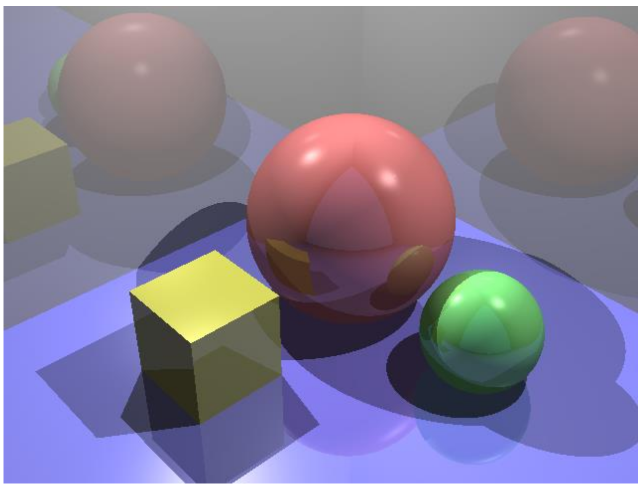
The relevant files to study first are **`Vector3d.h`**, **`Color.h`**, **`Ray.h`**, **`Material.h`**, **`Light.h`**, **`Surface.h`**, **`Scene.h`** and **`Raytrace.h`**.

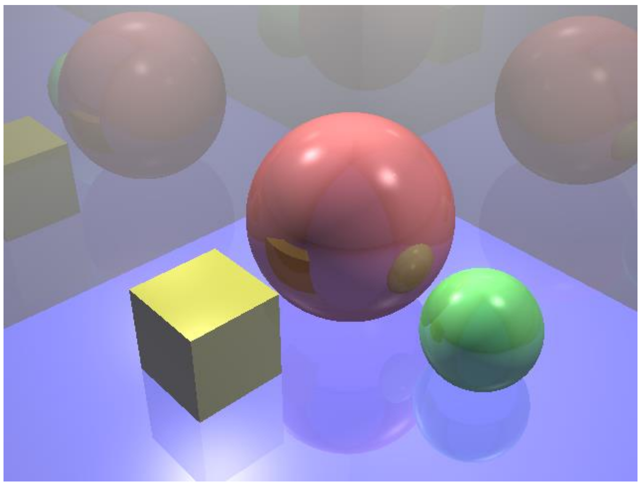
For this task, you have to **submit** your completed **`Raytrace.cpp`**, and the images generated by the program, which should look like the followings. There are **6 images** you need to submit, and you must name them as shown at the bottom of each image shown below.

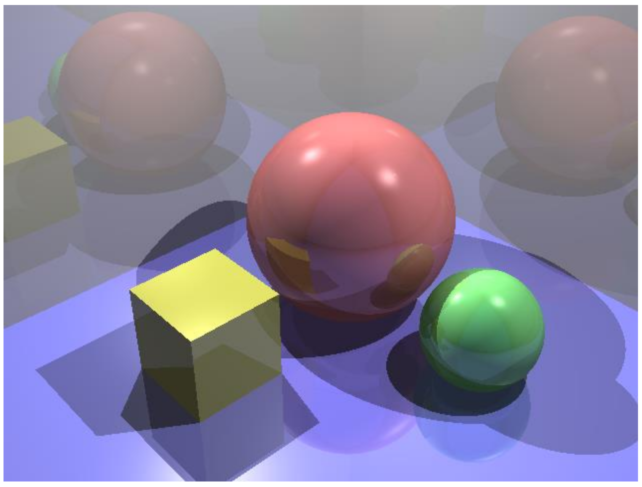

img_r0.png
reflectLevels = 0
hasShadow = false


img_r0s.png
reflectLevels = 0
hasShadow = true


img_r1.png
reflectLevels = 1
hasShadow = false


img_r1s.png
reflectLevels = 1
hasShadow = true


img_r2.png
reflectLevels = 2
hasShadow = false


img_r2s.png
reflectLevels = 2
hasShadow = true

Each of these images should take **less than 10 seconds** to produce. On my laptop, **`img_r2s.png`** (the most time-consuming) took about 1 second. In your IDE (e.g. Microsoft Visual Studio, Xcode), make sure you compile your program using the **Release** configuration.

<u>Task 3</u>

You are to complete the **`DefineScene2()`** function in **`Main.cpp`** to model a new scene for rendering. You must write your code only in places marked "**`WRITE YOUR CODE HERE`**".

You must use all the surface primitive types, namely, plane, sphere, and triangle, in your scene model. This task will be assessed based on the fulfillment of the above basic requirements, on the technical difficulty and object's complexity, and on the aesthetics and creativity. Your scene model should be rendered with `reflectLevels=2` and `hasShadow=true`, and an image resolution of **640x480**. Name your image file **`img_scene2.png`**.

For this task, you have to **submit** your completed **`Main.cpp`**, and the generated image **`img_scene2.png`**.

GRADING

The maximum marks for this programming assignment is **100**, and it constitutes **8%** of your total marks for the module. The marks are allocated as follows:

- **Task 1 — 30 marks**
- **Task 2 — 50 marks**
- **Task 3 — 20 marks**
  - 10 marks — basic requirements,
  - 5 marks — technical difficulty and object's complexity,
  - 5 marks — aesthetics and creativity.

Note that marks will be deducted for bad coding style. If your program cannot be compiled and linked, you get 0 (zero) mark.

Good coding style. Comment your code adequately, use meaningful names for functions and variables, and indent your code properly. You must fill in your **name**, and **NUS User ID** in the **header comment**.

SUBMISSION

For this assignment, you need to **submit only the following 11 files**:

- **`Sphere.cpp`** and **`img_spheres.png`**,
- **`Raytrace.cpp`** and **`img_r0.png`**, **`img_r0s.png`**, **`img_r1.png`**, **`img_r1s.png`**, **`img_r2.png`**, **`img_r2s.png`**,
- **`Main.cpp`** and **`img_scene2.png`**.

You must put it/them in a ZIP file and name your ZIP file ***nus-user-id*****`_lab4.zip`**. For example, if your NUS User ID is **`e0123456`**, you should name your file **`e0123456_lab4.zip`**.

Note that you will be penalized for submitting non-required files.

Submit your ZIP file to **`Canvas > CS3241 > Assignments > Lab Assignment 4`**. Before the submission deadline, you may upload your ZIP file as many times as you want. **We will take only your latest submission**.

DEADLINE

Late submissions will NOT be accepted. The submission page will automatically close at the deadline.

——— End of Document ———