RI3004A 3D Graphics Rendering

Assignment 4

Release Date: 20 July 2016, Wednesday

Submission Deadline: 25 July 2016, Monday, 11:59 PM

LEARNING OBJECTIVES

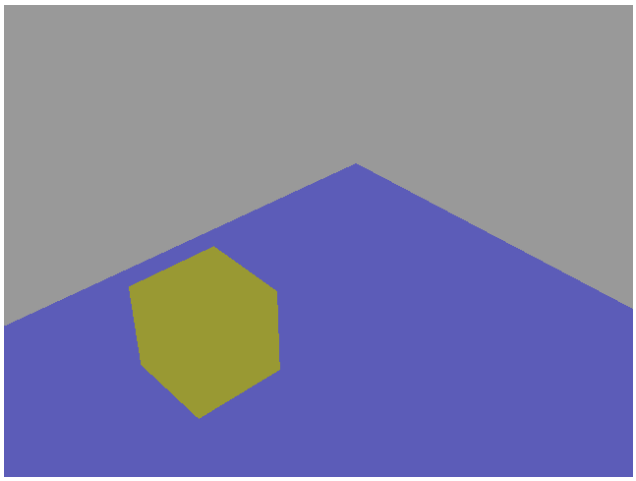
Implementing the Whitted Ray Tracing algorithm. After completing the programming assignment, you should have learned

- 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 **unfinished 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.

Please download the ZIP file **assign4_2016_todo.zip** from the **Assignments** folder in the IVLE Workbin. A Visual Studio 2008 solution file **assign4.sln** is provided for you to build your program. If you build (use Release configuration for better speed) and run the program, it will produce an image as follows (in file **out1.png**):



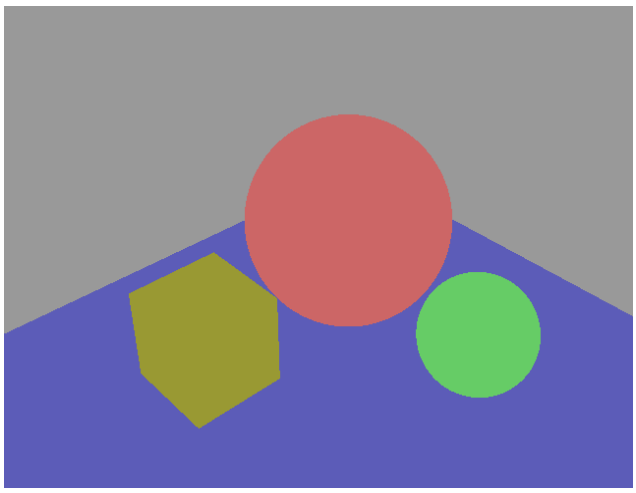
It shows an image of an unlit scene of 3 intersecting planes and a cube made of triangles. There are supposed to have two spheres in the image, but the ray-sphere intersection routine has not been implemented yet.

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

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

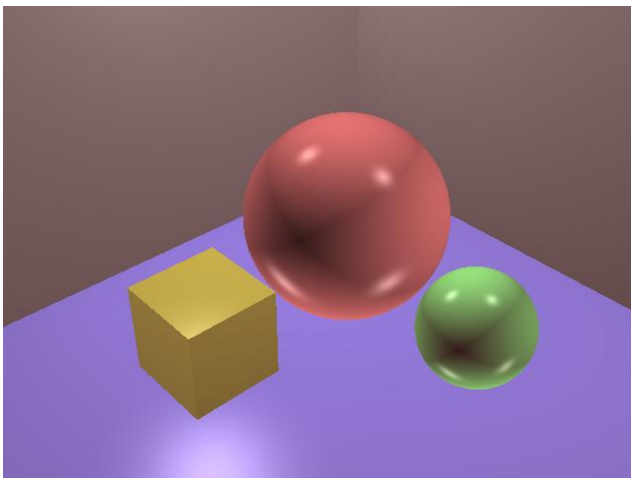
where

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

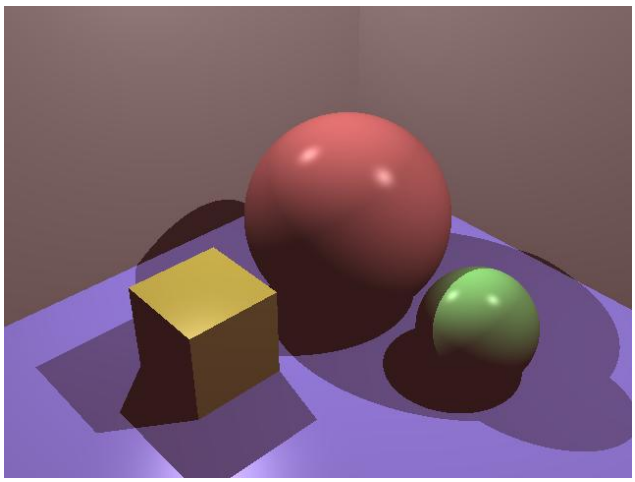
and M is the number of point light sources in the scene.

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

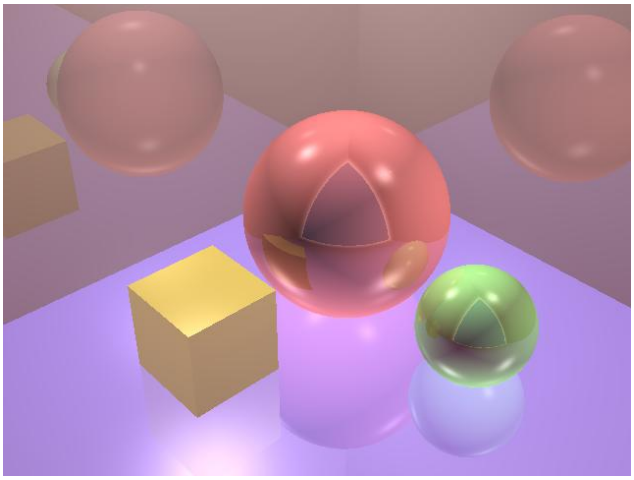
For this task, you have to **submit** your completed **Raytrace.cpp**, and the images generated by the program, which should look like the followings. There are **6 images** you need to submit, and you must name them as shown at the bottom of each image shown below.



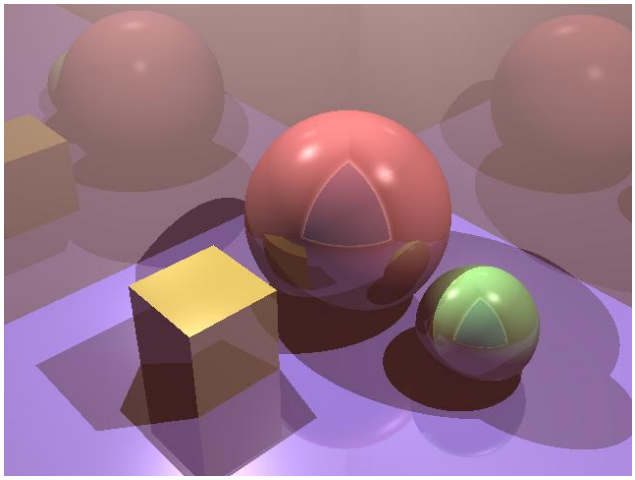
img_r0.png
`reflectLevels = 0`
`hasShadow = false`



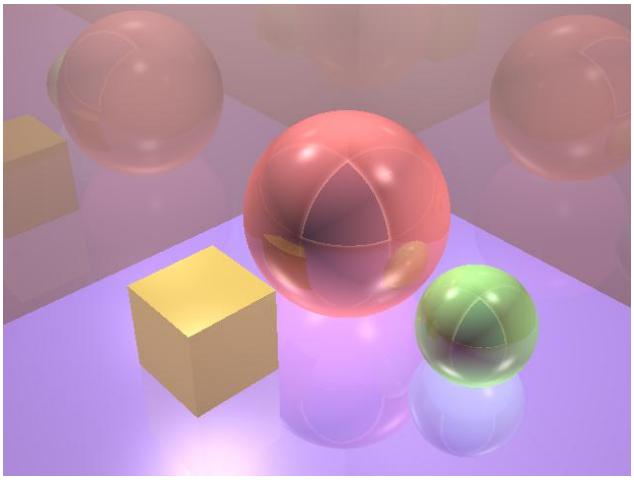
img_r0s.png
`reflectLevels = 0`
`hasShadow = true`



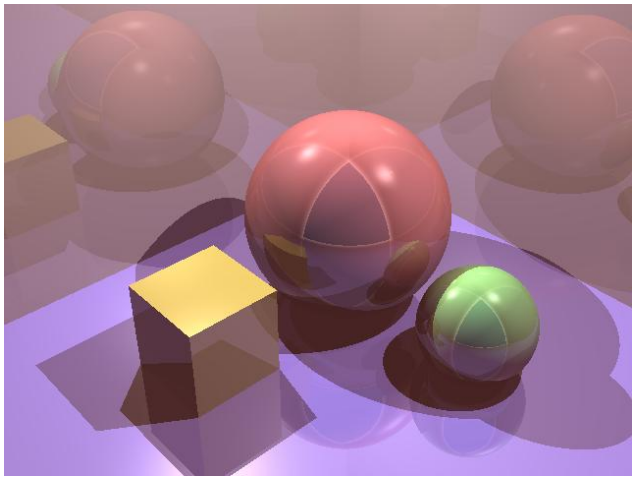
img_r1.png
`reflectLevels = 1`
`hasShadow = false`



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 20 seconds** to produce. On my laptop, **img_r2s.png** (the most time-consuming) took about 7 seconds. Make sure you compile your program using the **Release** configuration.

Task #3

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. Your scene need not be complex. It will be assessed by the aesthetics of the rendered image, which should be produced 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 **25%** of your total marks for the course. The marks are allocated as follows:

- **Task #1 — 25 marks,**
- **Task #2 — 60 marks.**
- **Task #3 — 15 marks.**

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 them in a ZIP file and name your ZIP file ***nus-user-id*_A4.zip**. For example, if your NUS User ID is **NUSRI16-999**, you should name your file **NUSRI16-999_A4.zip**. All letters in the filename must be capitalized.

Submit your ZIP file to the **Assignment 4 Submission** folder in the IVLE Workbin. Before the submission deadline, you may upload your ZIP file as many times as you want to the correct folder. **We will take only your latest submission.** Once you have uploaded a new version to the folder, you **must delete the old versions**. Note that when your file is uploaded to the Workbin folder, the filename may be automatically appended with a number. This is fine, and there is no need to worry about it.

DEADLINE

Late submissions will NOT be accepted. The submission folder in the IVLE Workbin will automatically close at the deadline.

——— End of Document ———