CMPUT 411/511

Assignment 5

©*Herb Yang*

October 29, 2021

Due date: **November 22, 2021** at 23:55 Total marks: 38

Important notes

- A late penalty of 10% per day applies to all late submissions. The maximum number of days late permitted is 1. After 1 day late, your assignment will be given a mark of **ZERO**.
- You are expected to complete the assignment on your own **without** collaboration with others. Discussions at the conceptual level but not at the coding level are permitted.
- You must write your own code. Using codes from the specified text or codes given in this course is permitted. Using any other codes is not permitted.
- Looking at other's codes or letting others to see your codes is not permitted.
- If your codes are used by another student, under the Code of Student Behaviour, you will be dealt with as participating in a plagiarism and cheating offence.
- If you hire a tutor, the tutor **cannot** give you step-by-step or code level instructions to solve the problem. This is defined as cheating.
- You are **not** permitted to upload this assignment to an online site to solicit solutions or to post your solutions to an online site for others to see. Violating this policy will result in severe penalty. These are defined as plagiarism and participation in a plagiarism offence, respectively.
- All your codes will be analyzed for potential plagiarism using MOSS.
- Do not wait until the last minute to work on your assignment. Start as soon as possible.
- Debugging programs and understanding of the materials take time. Debugging is part of the learning experience. You **must** not ask the TA to help you debug your programs.
- During lab, the TA is instructed to give directions but **not** solutions.
- For the programming part,
  - Your code should have sufficient comments to make it readable.
  - Include a README file to document features or bugs, if any, of your program.
- If there are questions, please ask the TA in the lab or via email.

Marking scheme:

- Full assigned marks - no observable bug
- Portions of the assigned marks - observable bugs. The TA has the discretion of assigning a fraction of the assigned marks depending on the number and the severity of the bugs.
- 0 marks - does not work at all

Raytracing: This problem helps you understand raytracing as discussed in class. The assignment is based on Peter Shirley's excellent online text: `Ray Tracing in a Weekend`, `Ray Tracing the Next Week`, and `Ray Tracing for the Rest of Your Life`. In this assignment, use and modify/extend the code developed by Dmitry Brant Link in here. While the code is slow, it is a faithful reproduction of the methods described by Peter Shirley. If your rendering area and the number of samples are small, the rendering speed is still quite tolerable.

1. (2 marks) Read Chapter 3 "Solid Textures" in `Ray Tracing in the Next Week`. Add a constant texture class. Render two spheres, one large one and a small one. If your implementation is correct, you should get the following results (Fig. 1.) The location of the large sphere is at (0, -500, -1), and a small one of radius 1 at (0,1.0, 1).

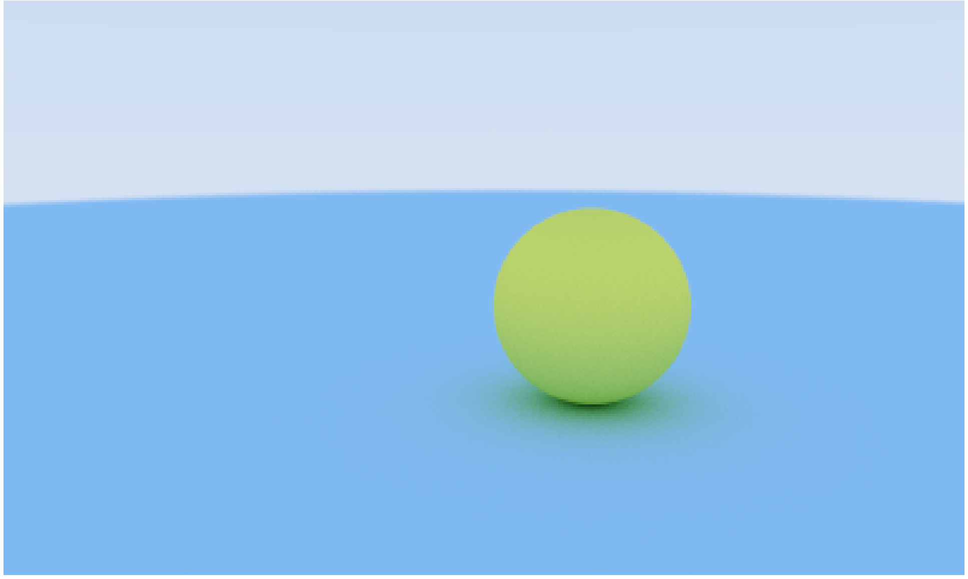

Figure 1: The results of using constant texture to render two spheres.

2. (3 marks) Add the checker texture class and apply it to the base sphere. You should see the result in Fig. 3.

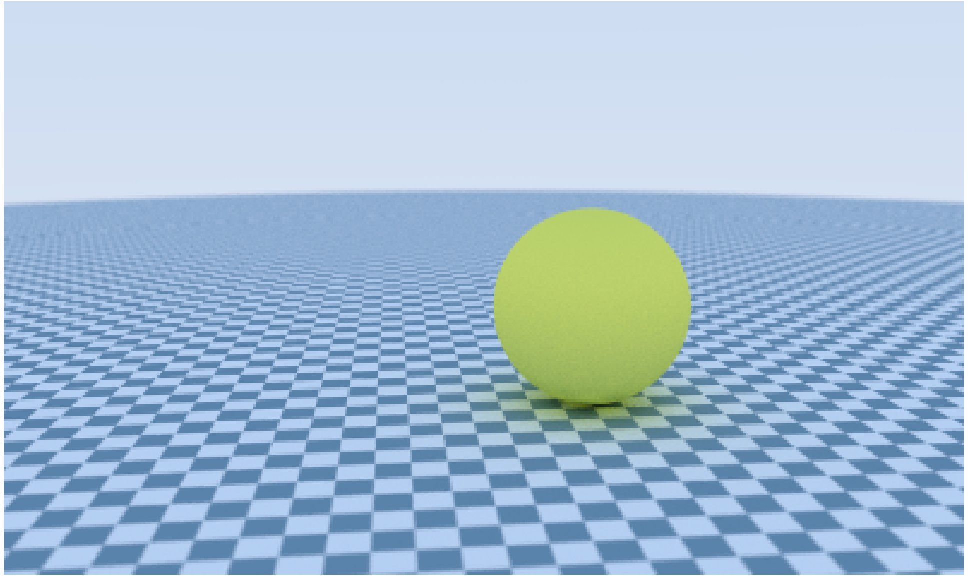

Figure 2: The results of using constant texture to render one sphere and checker texture on the other.

3. (3 marks) Add a disk class and use the metallic material to render the following scene:

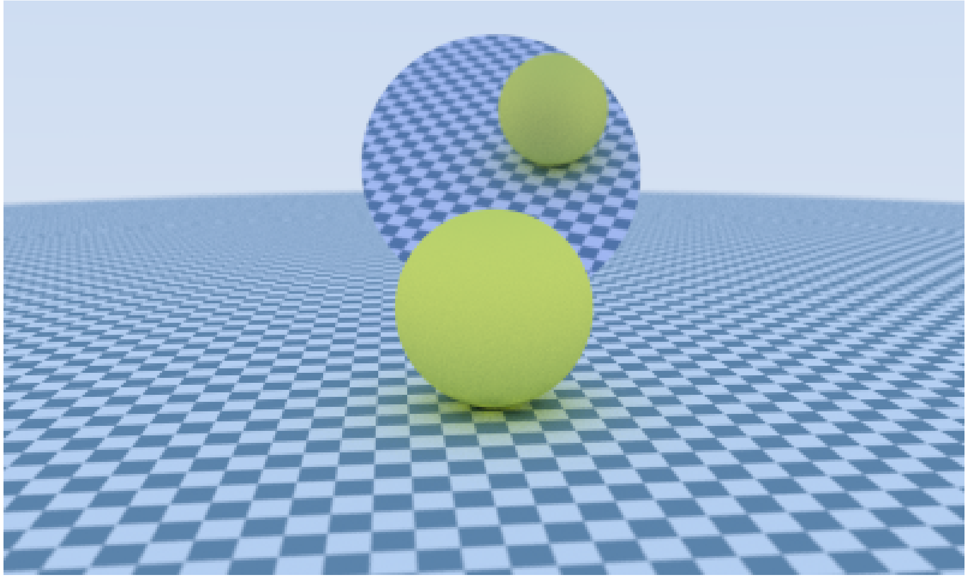

Figure 3: The results of rendering a reflecting disk with radius 2, center at (-0.5, 2.0, -3.0), and normal (-1,-1, 4) and the sphere of radius 1 is at (-1,1, 1).

4. (5 marks) Implement the cylinder class and use the metallic material to render the following scene:

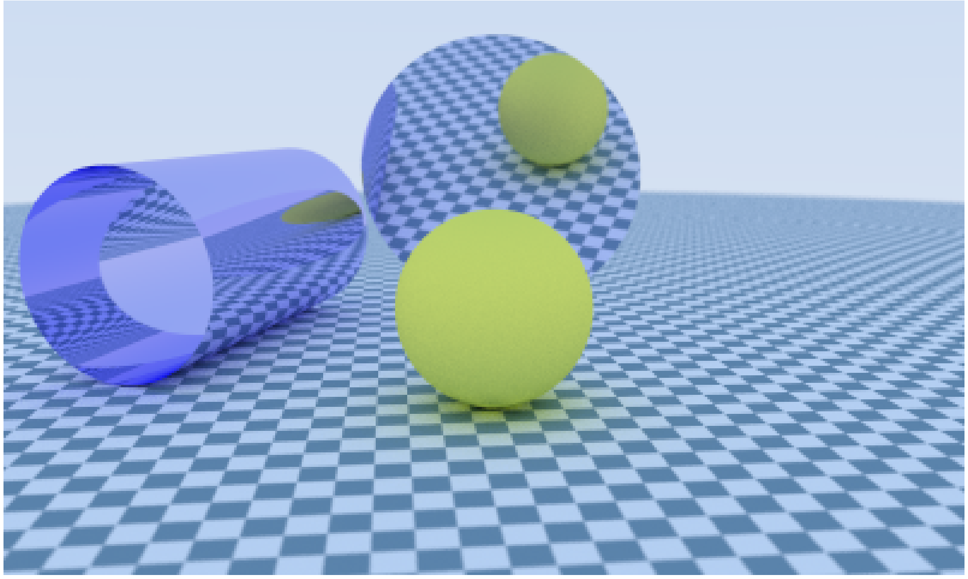

Figure 4: The results of rendering a reflecting cylinder of radius 1.2 with $C_1 = (-5, 1.2, 0.0)$ and $C_2 = (-3.5, 1.2, -4.0)$, a reflecting disk of radius 2 centered at (-0.5, 2.0, -3.0), and with its normal along the direction (-1,-1, 4) and a sphere of radius 1 at (-1,1, 1).

5. (2 marks) Implement the triangle class and render the following scene (Fig. 5):

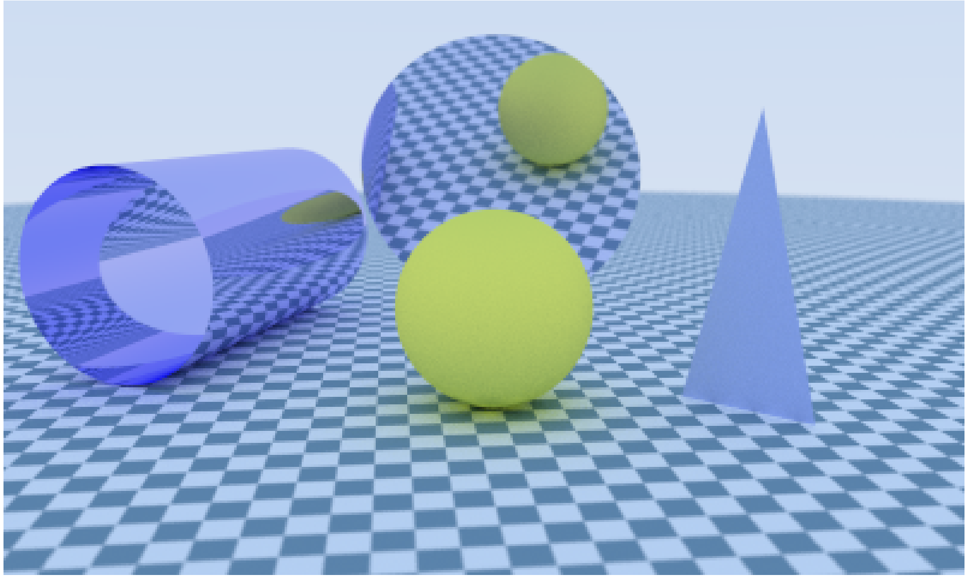

Figure 5: The results of rendering a Lambertian triangle with vertices at (1,0,1), (2,0,2) and (1.5, 3, 1.5). Other scene components are similar to that in Fig. 4.

6. (3 marks) Implement a program to generate a ring of radius 4 of 20 Lambertian spheres each of radius 0.4 of random colors. The lookAt is (0,1,0). The result is shown in Fig. 6.

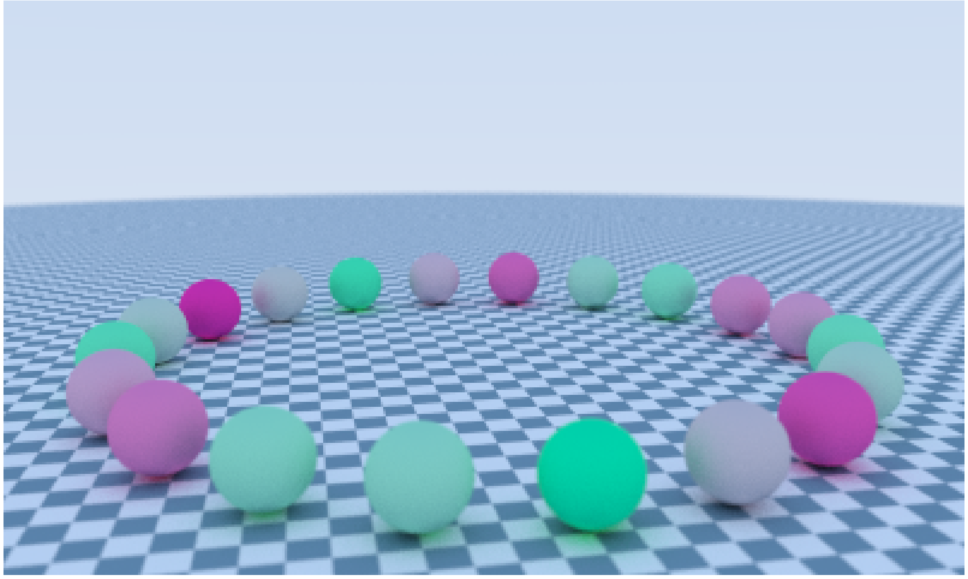

Figure 6: A ring of 20 random colored Lambertial spheres, each of radius 0.4.

7. (5 marks) Implement the equilateral tetrahedron class and render a scene with a Lambertian tetrahedron of side length 4.0 centered at the origin.

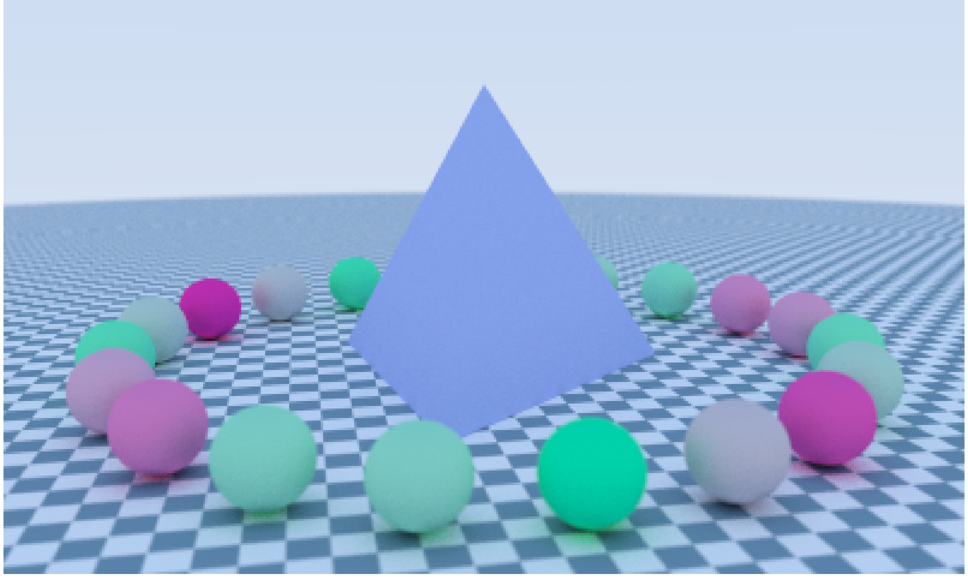

Figure 7: A Lambertian tetrahedron and a ring of 20 random colored Lambertial spheres, each of radius 0.4.

8. (5 marks) Render the same scene as in Q. 7 but change the material of the tetrahedron to metallic.

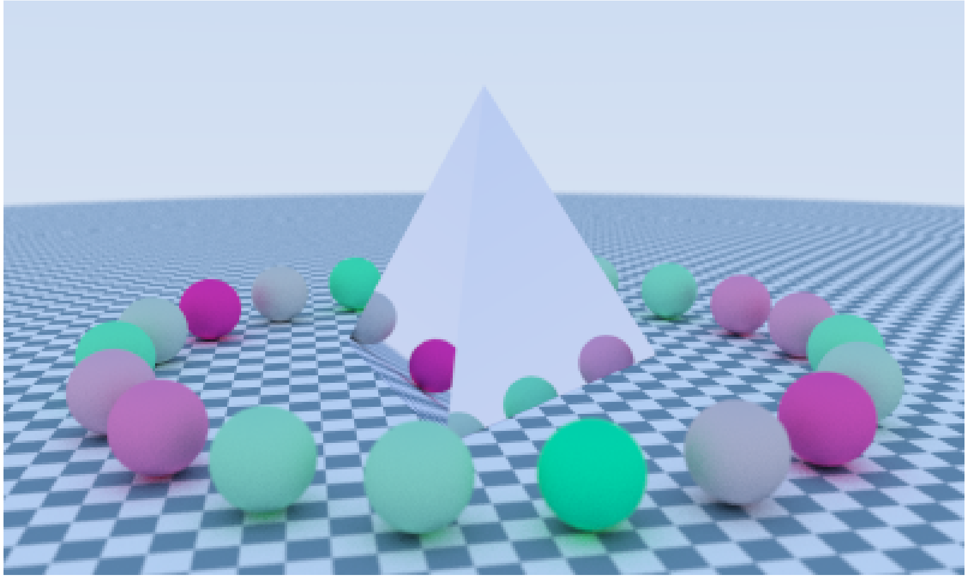

Figure 8: A Lambertian tetrahedron and a ring of 20 random colored Lambertial spheres, each of radius 0.4.

9. (10 marks) Render the same scene as in Q. 7 but change the material of the tetrahedron to dielectric.

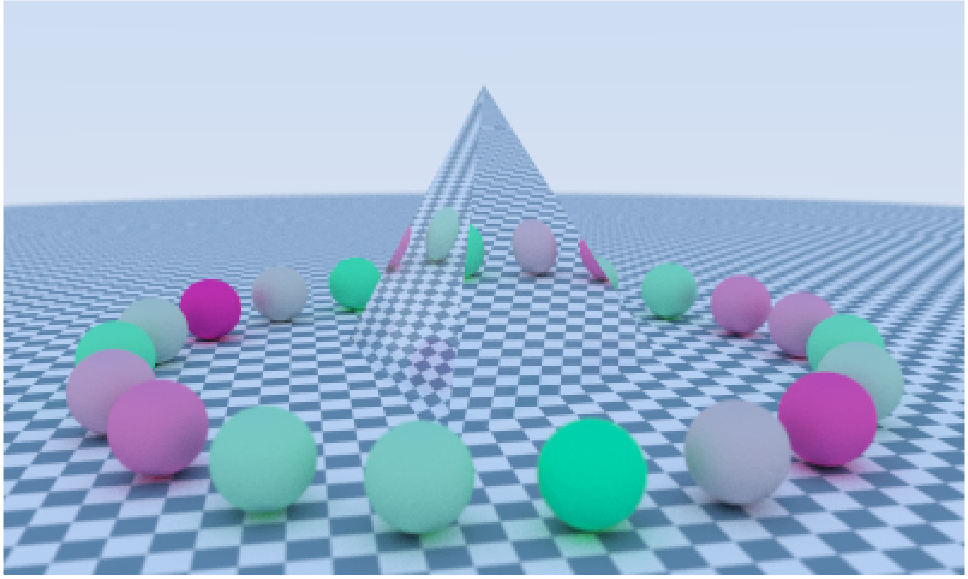

Figure 9: A Lambertian tetrahedron and a ring of 20 random colored Lambertial spheres, each of radius 0.4.

Submission information

- Organize the solution of each question in a separate folder, with a name that corresponds to each question, e.g. q1, q2,...
- Zip up all the folders in one single file
- Include a README file regarding your implementations, if needed
- Upload the zipped file to eClass