CMPUT 411/511

Assignment 6

Herb Yang

November 20, 2021

Due date: **December 3, 2021** at 23:55 Total marks: 10

Submit your assignment electronically to eClass. Organize your files properly into folders and zip up all the folders that correspond to a particular question into one single zip file for submission.

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

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

(10 marks) The goal of this assignment is to get you familiar with using the distance function of a torus to generate a simplified Olympic symbol.
Note: In the supplied `sphere.html` file, you may need to change the paths for the library files and for the `sphere.js` file. In the accompanied olympic.gif, it shows the final result when everything is working properly. You don't need to make your results to look exactly the same as that shown in the animated gif. As long as it is close, then it is fine.
Based on the provided sphere fragment shader, create fragment shaders to generate the required output for each of the following questions. To simplify the work of the TA, provide each shader in a separate file. Name your files as `torus1.js`, `torus2.js`,..., `tours7.js`.

1. (2 marks) Using the method discussed in the lecture for the distance function of a torus, give the shader that generates a torus centered at the origin (Fig. 1) with $R = 0.3$ and $r = 0.05$ with the following surface reflectance properties:

$$\begin{aligned} k_a &= vec3(0.1, 0.1, 0.1); \\ k_d &= vec3(0.1, 0.1, 0.1); \\ k_s &= vec3(1.0, 1.0, 1.0); \\ shininess &= 1.0; \end{aligned}$$

Figure 1: A torus

Figure 2: Two tori

2. (1 marks) Give the shader that generates two touching tori (see Fig. 2). The surface properties of the second torus are given below:

$$\begin{aligned} k_a &= vec3(0.1, 0.1, 0.9); \\ k_d &= vec3(0.1, 0.1, 0.9); \\ k_s &= vec3(1.0, 1.0, 1.0); \\ shininess &= 1.0; \end{aligned}$$

3. (1 marks) Give the shader that generates three touching tori (see Fig. 3). The surface properties of

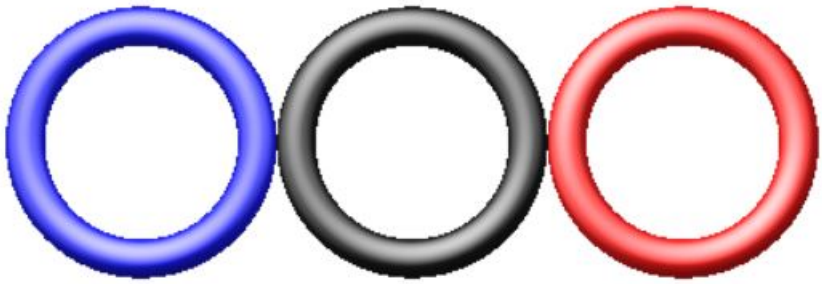

Figure 3: Three tori

the third torus are given below:

$$\begin{aligned} k_a &= vec3(0.9, 0.1, 0.1); \\ k_d &= vec3(0.9, 0.1, 0.1); \\ k_s &= vec3(1.0, 1.0, 1.0); \\ shininess &= 1.0; \end{aligned}$$

4. (1 marks) Give the shader that generates four tori as shown in Fig. 4. The surface properties of the

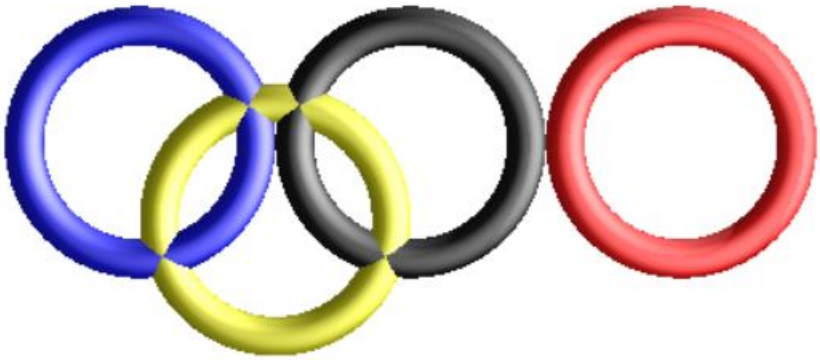

Figure 4: Four tori

fourth torus are given below:

$$\begin{aligned} k_a &= vec3(0.8, 0.8, 0.1); \\ k_d &= vec3(0.8, 0.8, 0.1); \\ k_s &= vec3(1.0, 1.0, 1.0); \\ shininess &= 1.0; \end{aligned}$$

5. (2 marks) Give the shader that generates five tori as shown in Fig. 5. The surface properties of the

Figure 5: Five tori

fifth torus are given below:

$$\begin{aligned} k_a &= vec3(0.1, 0.8, 0.1); \\ k_d &= vec3(0.1, 0.8, 0.1); \\ k_s &= vec3(1.0, 1.0, 1.0); \\ shininess &= 1.0; \end{aligned}$$

6. (2 marks) By appropriately adjusting the orientations and positions of the yellow and green tori, give the shader that generates the simplified Olympic logo as shown in Fig. 6. Your result should

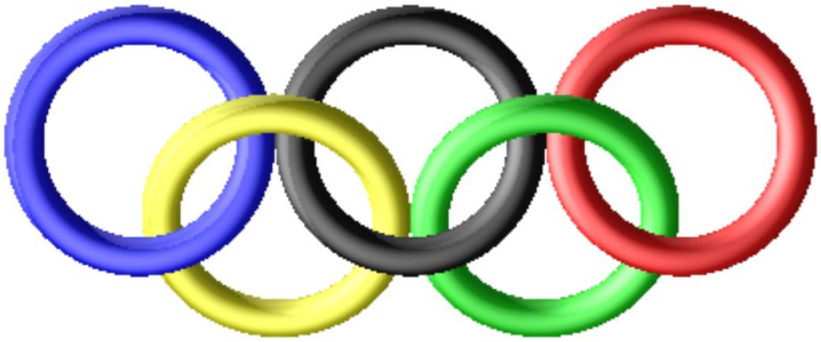

Figure 6: The simplified Olympic logo

not have any torus intersecting another one as shown in Fig. 7.

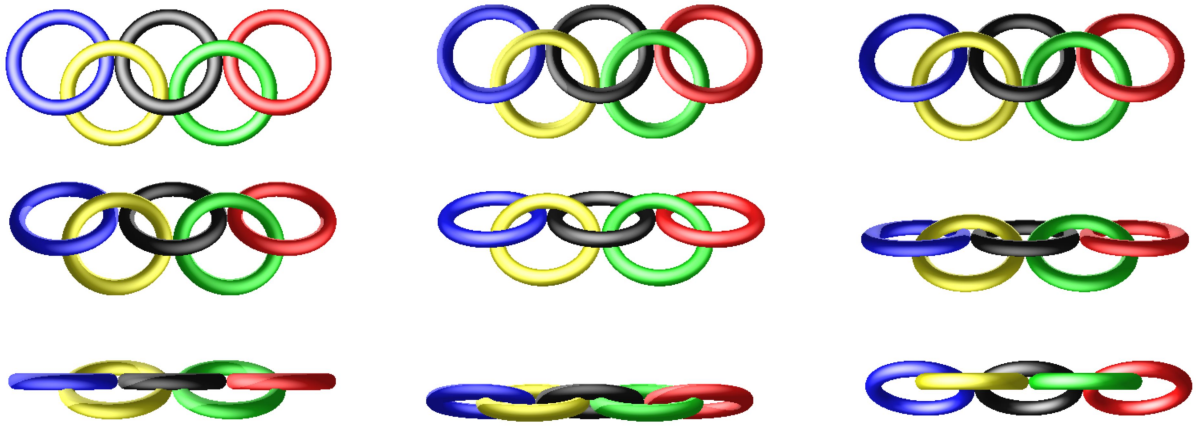

Figure 7: Different views of 5 interlocking tori

7. (1 marks) Use `iTime` to rotate continuously the Olympic logo with respect to the `x-axis` to show different views of the Olympic logo (see Fig. 7).