UOW AUSTRALIA

SCIT

School of Computing & Information Technology

CSCI336 – Interactive Computer Graphics
Autumn 2019

Assignment 1

Due on Tuesday, 9th April 2019 at 23:55

Task 1

Write an OpenGL program to create a simple 2D scene with an animated Ferris wheel. The image in Figure 1 shows an example of the scene (a working program will be shown in one of the lectures).

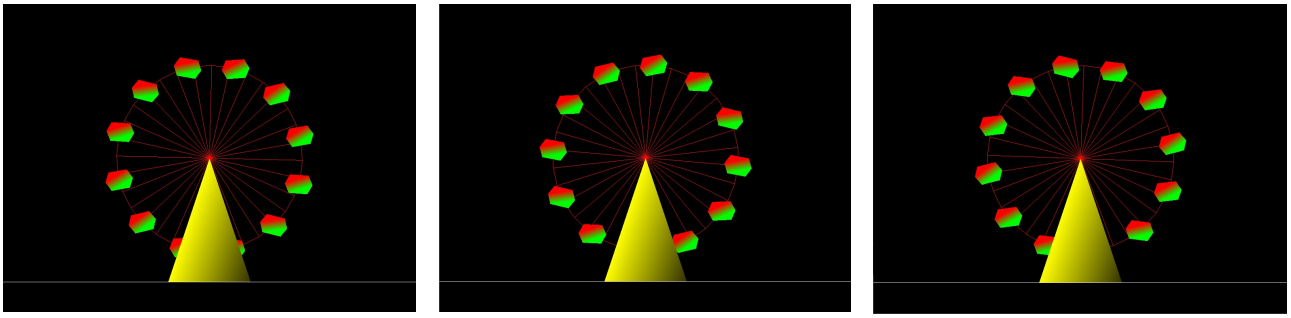

Figure 1: Ferris wheel.

- Ferris wheel
  - Construct a 2D scene like the one shown in Figure 1 (does not have to be identical). The scene should consist of the following objects:
    - A line for the ground
    - A triangle for the base supporting the Ferris wheel
    - A triangle fan for the Ferris wheel spokes
    - Several carriages (Note that your program should only contain the vertex specification of a single carriage, you can reuse this with different transformations)

    Note: the rendering order affects the final image. The "Painter's Algorithm" – things drawn later will overwrite the previous contents.

    (2 marks)
  - Implement basic animation using transformations.
    - Make the Ferris wheel rotate. The carriages should rotate along with the wheel spokes.

(2 marks)

- Allow the user to toggle animation by pressing "p" on the keyboard, and to change the direction of rotation by pressing "d".

  (1 mark)

- As the Ferris wheel rotates, make the carriages sway (rotate a little clockwise, then counter-clockwise). Each carriage should have a different sway angle.

  (1 mark)

Task 2

Write an OpenGL program to create a simple 3D scene of a planetary system using colour cubes to represent the "planets". The image in Figure 2 shows an example of the scene (a working program will be shown in one of the lectures).

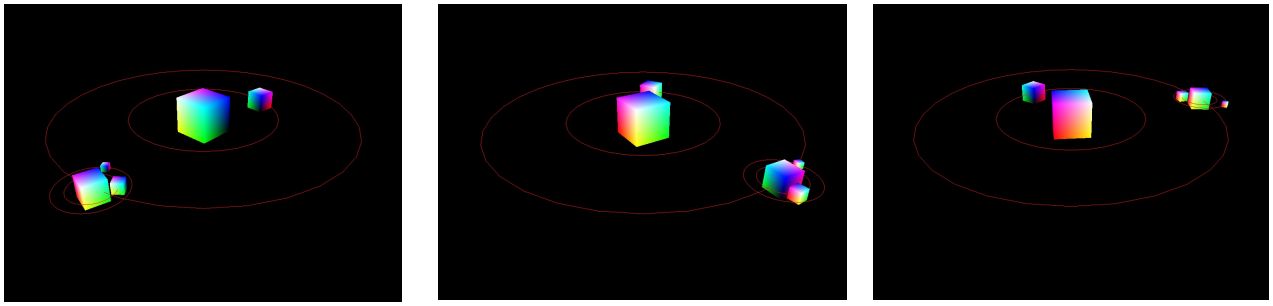

Figure 2: Planetary system.

- Construct the 3D scene using perspective projection.
  - The scene should consist of a large "sun" in the centre and two other "planets" that orbit the sun. Note that your program should contain the vertex specification of a single colour cube. The sun and the planets are to use the same vertex specification, i.e. use transformations to alter the position, size and rotation.

    (1 mark)

  - Implement basic animation.

    The sun and the planets rotate about their respective centres. Each planet also orbits the sun. Each planet's rotation and orbit angle should be different. The planetary orbits and rotation only need to be in the horizontal plane, i.e. rotation about the y-axis.

    (1 mark)

  - Display the orbit paths, i.e. the red circles in Figure 1. You only need the vertex specification of a single line loop. Use transformations to display the orbit path at different positions and sizes.

    (1 mark)

  - Implement two "moons" that rotate about their own centres and orbits the outer planet. Each moon's rotation and orbit angle should be different.

    (1 mark)

Screenshots

In your submission, include two screenshots for each of your programs. The screenshots are to show two different animation frames. Save the screenshots using one of the common image formats, i.e. bmp/jpg/png.

Instructions and Assessment

Zip **all your source files** (.cpp, .h, .vert and .frag) and **screenshots** (.bmp/.jpg/.png) into a single file and submit this via Moodle by the due date and time (do **NOT** zip your entire visual studio project file as this can be very large). Assignments that are not submitted on Moodle will not be marked.

You will have to demonstrate your working program during the lab in Week 6. Do not try to fix your code during this lab, your program will be assessed based on what you submitted. Your program must work on the computers in the lab or you must be able to demonstrate that it works on your own laptop.

The assignment must be your own work. If asked, you must be able to explain what you did and how you did it. Marks will be deducted if you cannot correctly explain your code.

NOTE: The marking allocations shown above are merely a guide. Marks will be awarded based on the overall quality of your work. Marks may be deducted for other reasons, e.g., if your code is too messy or inefficient, if you cannot correctly explain your code, etc.

For code that does not compile, does not work or for programs that crash, the most you can get is half the marks (i.e. 5 marks or less).

