CS324 Coursework Assignment - OpenGL/GLSL 2016

Abhir Bhalerao

November, 2016

1 Introduction

Your coursework assignment task is to write a graphics program using OpenGL and/or GLSL. You should use what you have learnt in the practical laboratories (GLUT, GLU, OpenGL and GLSL). Note that the program **must** compile and run on the department's linux machines and you should test this out before you submit your code for assessment.

Please choose **one** of the problems given below.

You will see that some of them require you to write 2D graphics, some 3D graphics and perhaps use programmable shaders.

I expect that to complete the task well, you need to spend about a week on it (20-30 hours). Please don't spend more than this time, and try to complete the required functionality in the first instance.

If you have any questions please come and see me.

Good luck and have fun!

2 Problems

Please chose only **one** problem to solve.

2.1 Roller Coaster

Write code to simulate a roller coaster ride. It should have a smooth 3D track, perhaps built on a mesh landscape. The roller coaster should have three or more cars which you may want to speed-up and slow down appropriately.

You should allow the user to change the view point from looking ahead in the front car and seeing the whole roller coaster.

The simulation does not need to implement any physics.

2.2 Robot Arm

1. Using the Robot Arm program given in the labs, extend it to allow the (two) fingers of the hand to perform a grab movement. Then make it so that it can pick up simple objects, such as cubes or rings.

2. Program your arm to demonstrate and solve the Towers of Hanoi problem.

Please note that I don't expect you to implement either solid-body physics (but you may want to implement simple collision detection perhaps by dividing the space into small cubic regions); or any form of inverse kinematics (but you may try to restrict arm movements to be 'realistic').

2.3 Pac-Man

Implement a reasonably faithful version of the famous 1980's arcade game, Pac-Man, from first principles in OpenGL.

You should make the game look and give it an appropriate scoring mechanism.

2.4 Rasterization Demo

1. Write a demo to show how Jack Bresenham's circle algorithm works.

2. Write a second demo to show the steps of the polygon scan-line filling method. You might allow either the user to pick a standard set of polygons to test or create their own.

2.5 Clock and Calendar App

1. Make an analogue and digital clock display.
2. Extend your analogue clock to have a choice of alternate hands, faces and a day/date indicator.
3. Implement a stop-watch mode with reset, start and stop buttons.
4. Add functions to the clock to allow the current date/day to be displayed.

You may want to use texture mapping to implement a variety of clock faces, hand and numeral styles.

2.6 Splines and Tubes

1. Write a demo for 2D and 3D spline curves.
2. Use a geometry shader to create splines in 2D and 3D and use it to generate random sets of connected tubes.

2.7 2D Drawing Tool

Write a simple drawing application. It should allow users to draw text, lines, circles, rectangles and optionally fill these objects from a palette of colours and change their transparencies. There should be a tool bar to select the drawing object and drawing mode. The user should be able to move, resize and delete the objects once created. There should be a way to save and load drawings to and from a file.

2.8 Maze

Write a maze application to allow you to navigate a maze shown in 3D. You should maintain a first-person view and clearly mark start and end points. You may want to provide a 2D view (map) mode if a user gets stuck. The walls of your maze should be coloured and/or textured.

3 Assessment Criteria

The coursework marks (out of 20) will be allocated equally to the following 4 areas:

Functionality	How much of the functionality required is implemented by the solution and how well it has been done.
Code	Has appropriate features of OpenGL/GLSL been used in the solution. How readable the code it is.
Look and Feel	The quality of the graphics produced, the ease of use of the solution.
Documentation	Readability and quality of presentation of the documentation. Contents describe well the features of the solution, and instructions given for compilation and running are satisfactory.

4 Collaboration and Plagiarism

As for all pieces of assessment you undertake, the submission must be your **own** work. You may have been working in pairs in the practical labs, but this piece of work must be done individually. Discussion ideas with your colleagues is OK, but the final piece of work should be your own.

Also, you must not copy without attribution from work you may have found on-line. Using any code that is given in the laboratory tutorial examples is permitted.

We will use anti-plagiarism software to look for copying between submission and with external sources, and any breaches of the rules will incur a severe penalty.

5 Deadlines and Submission Formats

The deadline for this coursework is **12pm Monday, 16th January 2017**.

Please submit a **single** ZIP file containing a set of source files (`.cpp`, `.h`), makefiles, and a PDF of a supporting document which describes your code. As mentioned earlier, do make sure your programs compile and run correctly on the DCS Linux systems. We will be compiling, running and testing your code.

Supporting Document Your supporting document must be **no more than 750 words** in length. Please indicate the word count at the end of the text. It should state **clearly and concisely**: features of your solution; main design aspects; how specific OpenGL/GLSL features are employed; how to compile and run it; how to use it. You may include screen-shots to illustrate your supporting document.

Abhir Bhalerao, November 2015