1 Instruction

Zheng’s OpenGL Version is 3.30. Update your driver to make sure your OpenGL is at least as new, otherwise you may not be able to use the start code. You can check your OpenGL version with GLviewer.

You should have already installed QT following the instruction on Prof. Khan’s web page. Zheng’s environment is Windows, VS2013, QT 5.4.0 with QT creator. If you are using Windows, please download the specific version of QT here. The online installation of QT may cause issues. You can wrap the whole QT project and send it to me. Please clear your build before sending it to Zheng to avoid the email being blocking. If your environment is different from Zheng’s, please include a readme file to explain how to configure your project.

The start code is tested in Windows and Mac OSX with QtCreator. Feel free to contact Zheng if you can not successfully run the start code.

Zheng can read C/C++, C#, MATLAB, Python, Java. Try not to use other languages. Otherwise you may spend a lot of time explaining your code to Zheng.

For hardcopy handin:

You only need to print out the core parts of the code. If you print out everything, Zheng may need you to explicitly indicate the core parts. Write comments to explain what you have done.

OpenGL tutorial:

[1] http://www.opengl-tutorial.org/ : a simple tutorial, the basic part may be enough for you to complete the course project.


2 Programming Exercise: Car 101

You will program a simple vehicle that can move around a 3D scene. Think of it as a car that is driving on a plane that also contains some buildings. You will add buildings, and the ability to steer the car and change its speed. You’ll also add a special somersault feature. We are including an executable showing Zheng’s version of the finished project, please make sure QT is added to system variable to run the demo. In Windows, to do this go to Control Panel — System and Security — System. Click on Advanced systems settings on the left.
on Environment Variables on the bottom. Add ’QT_INSTALL_PATH
5.4
msvc2013_64_opengl
bin’ to the Path variable.

Your version need not be identical to this in every detail. For example, you
can vary the speed at which the car moves, how rapidly it steers, or the size
and position of the buildings. However, keep the interface the same, using the
keyboard functions we specify, so that we will be able to easily test the result.

We are giving you skeleton code that will display a textured cube on the
screen. You should be able to open it in QtCreator and run it. If your QT cannot
handle resources correctly, you may need to change RESOURCE_FLAG (change
it to false) and TXT_IMG_PATH (change it to ’PROJECT_PATH/PS2_start/images’) in
mywidget.h to render the texture properly.

You will enhance this code to add more cubes, to change the type of projec-
tion, and to allow yourself to navigate about the scene. You may use features
provided by QT to implement your code. You may edit any part of the code,
but try to keep it neat.

2.1 Add more buildings (15 points)

For this part, enrich the scene that you are looking at by adding at least four
more buildings on the ground plane. You may take a look at the functions:
loadCubes(), moveCube() and addCube().

2.2 Color and Shader (25 points)

Add different colors for buildings (cubes) to distinguish them. You need to change
the color of each cube to make them distinct. For example, your first cube may
be green, second cube may be red. You should keep the textures to distinguish
the six surfaces of one cube, as shown in Fig. [1] You may take a look at the
shader program, and Chapter 2, Chapter 4 in tutorial [2].

2.3 Manual control (20 points)

Add control to the car. Use ’Up’, ’Down’, ’Left’, ’Right’ keys on the keyboard to
control the car, allowing it to move forward, backward, turn left and turn right.
Users should also be able to use the mouse to change the moving direction.
Pressing on the left mouse button should turn the car to the left. The right
mouse button should steer the car to the right. You may take a look at QT class
QMatrix4x4, Chapter 6 in tutorial [2] and Chapter 3 in tutorial [1].

2.4 Automove (20 points)

Now you need to implement another control mode: move forward at fixed speed.
The car should always move forward, in the direction in which it is looking.
Press ‘Space’ on the keyboard to change the control mode between automove and manual control. While moving, you can also steer the car, turn left/right, use the left/right button of mouse. You may take a look at QT class QTimer.

2.5 Change speed (10 points)

While the car is moving, allow keyboard input that will change the speed of the car. Pressing ‘s’ should slow the car down, while pressing ‘f’ should make it go faster.

2.6 Somersault (10 points)

The car also has a special trick. Pressing ‘m’ causes it to do a somersault. During a somersault, the car rotates about a direction that is perpendicular to the direction it is going and that is also perpendicular to the up direction. The somersault trick should work in both manual control and automove modes. The car continues to move forward while somersaulting in automove mode.

2.7 Challenge Problem (up to 20 points)

For extra credit, you can add additional features. Some possible ideas: add objects in other shapes, such as a pyramid or prism (up to 15 points); calculate collisions so that you don’t run into the cubes (up to 20 points); on the side or in another window, show the scene from overhead (up to 5 points); add a curved bridge and have the car’s position follow the bridge, so that as you go over it
you are pointing up and then down (ie, you are not always pointing parallel to the ground plane as you drive, up to 10 points); or add a roller coaster that the car can drive onto (up to 10 points).