**CPSC 3710: Computer Graphics**

**Final Project**

Darsh Thanki, Taranjot Kaur,

Vincent Cote, Zach Nelson

Spring 2019

**Tutorial**

To compile the project, simply type "make" into the console when in the project's folder. This command should create an executable titled "main." Running that main executable should start the game.

**Controls:**

* **z:** moves the robot forward
* **q:** turns the robot left (at intersections only)
* **a:** turns the robot right (at intersections only)
* **F2:** turns the robot’s head to the right
* **F3:** turns the robot’s head to the left
* **F4:** returns the view to normal
* **F5 – F12:** Changes different views of the robot
* **p:** pauses the game
* **r:** brings back the robot to its starting point
* **left-click:** on a building to destroy i

**Distribution of Work**

For our project, we divided the work up as evenly as possible, and each worked on our separate parts before bringing them all together. Our work assignments were as follows:

* Darsh Thanki's main contribution to the project was to create the general city blocks, streets (with Taranjot), as well creating the various building types.
* Taranjot Kaur's part was to work on developing the streets. Additionally, she worked on writing the tutorial and project documentation.
* Vincent Cote's job was to work on the camera angles as well as the various controls in the main file.
* Zach Nelson's portion was to create the robot as well as help create some basic functionality for it, such as turning the head and helping to rotate the entire robot itself.
* After we had each finished our parts, we came back together and worked as a group to bring the entire project together by combining all of our portions. To implement the building destruction feature, we worked on it as a group.

**Special Data Structures**

* Vector – used to store dynamically generated buildings
* Objects – we created individual objects for the robot and buildings.

**OpenGL That We Have Learned**

While working on this project, there were some different OpenGL functions which we had to teach ourselves in order to implement all of the required functions properly.

* We had to learn how to use the "specialKeyUp" functions, as they work differently than the mouse functions. When using the mouse functions, you can check what the current state of the mouse button is so that you can see if the button is released. However, this is not possible for the keyboard functions, so we had to research and implement a way to tell when the keyboard keys were released.
* We had to use glLineStipple which masks fragments produced by rasterization and stops them from being drawn. This function was needed when building the city streets
* glPushAttrib was used to specify the mask for which our building attributes saved.

**Assumptions**

Throughout our work on the project, we had to make a few assumptions about how certain things should work and about how we should implement them. They are as follows:

* When turning the robot, we couldn't find any information about what should happen to the camera when we do so. We had to assume that the camera should always remain behind the robot, so we made it so when the robot turns, the lookAt function changes to accommodate for that rotation.
* In the project description, we were told to make sure that the antenna is always spinning. However, due to it being a cylinder, it usually wouldn't appear to be doing anything if it was just rotating on its Y-axis. We decided to show it by moving the antenna slightly forward on the Z-axis so that it is possible to see that it is rotating.
* Since we were required to choose our own sizes for all of the objects within the game, we had to assume which sizes would work best. Therefore, we decided to randomize the building heights and the probability of the building being created.

**Other Documentation**

The description of various functions used in the program are as follows:-

* **initBuildings()** :- used to initialize the buildings using random heights, random building types, and dynamically creates them based on chance. The buildings are initialized and stored in a vector for easy access and usage in the rest of the program.
* **GL\_SILENCE\_DEPRECATION**:- used for silencing warnings that are deprecated generally on MacOS.
* **glMatrixMode(GL\_MODELVIEW)** :- this applies subsequent matrix operations to the modelview matrix stack.
* **glLoadIdentity()** :- replaces the current matrix with the identity matrix.
* **glClear(GL\_COLOR\_BUFFER\_BIT | GL\_DEPTH\_BUFFER\_BIT)** :- this clears the depth buffer along with the color buffer.
* **glTranslatef()** :- this multiplies the current matrix by a translation matrix which produces translation specified by (x,y,z).
* **glRotatef()** :- this function computes a matrix that performs a counterclockwise rotation of angle degrees about the vector from the origin through the point (x, y, z).
* **glutSwapBuffers()** :- performs a buffer swap on the layer in use for the current window; that is it promotes the content of the back buffer of the layer in use of the current window to become the contents of the front buffer.
* **glViewport()** :- sets the viewport where the first two parameters specify the lower left corner of the viewport rectangle and the other two parameters specify the width and height of the viewport.
* **glClearColor ()** :- specifies the clear values for the color buffers.
* **glEnable(GL\_DEPTH\_TEST)** :- used for enabling the depth test.
* **glutInit()** :- used to initialize the GLUT library.
* **glutInitDisplayMode ()** :- used to set up the initial display mode.
* **glutInitWindowSize()** :- used for specifying the size of the window on the screen and if omitted the window will have a default size.
* **glutInitWindowPosition()** :- used for specifying the position of window on the screen.
* **glutCreateWindow()** :- creates a window with the specified size and position as in the above function and gives it a specific title.
* **glutIdleFunc()** :- sets the global idle callback to be a function so that the GLUT program can perform background processing tasks.
* **glutReshapeFunc()** :- specifies what to do when window is resized.
* **glutKeyboardFunc()** :- used to tell the windows system which function we want to process the normal key events.
* **glutSpecialFunc()** :- used to register a callback handler for special keys.
* **glutMouseFunc()** :-sets the mouse callback for current window.
* **glutMainLoop()** :- enters the GLUT event processing loop.
* **void specialKeyboard()** :-used for adding special keyboard functions to be used in the program.