COMP 5821M: Animation & Simulation 2023-2024

ASSIGNMENT 1: Flight Simulation & Particle Systems **[30 marks]**

You have been provided with a simple interface which already has basic OpenGL model loading and rendering implemented. Your job is to add animation in the form of flight simulation and particle systems. You may assume that the coordinate system and all models are measured in metres.

The programme should take 3 parameters: the initial x, y, z coordinates of the plane, to allow us to test your code effectively. The initial speed should be set to 0.

Note that the FlightSimulatorWidget.cpp file has the paintGL() function working. It simply calls a scene model to render. In order for this to animate, it needs to be called repeatedly, with an opportunity to update positions. This is achieved through the use of a QTimer – a class which takes care of waiting for a fixed period of time. When the timer activates, it calls another function called nextFrame(), which updates the scene, then calls the QT update() function to ask for the widget to be rendered again. This is known as a busy loop, and hogs the processor, but renders frames as fast as possible, although there are no guarantees of timing. You should not need to modify initializeGL(), resizeGL(),paintGL(), or nextFrame().

The keyPressEvent() function implements a very basic user interface, using keys to manipulate the programme. One has been implemented for you so you can see how it is done: the letter ‘X’ (or ‘x’) will cause the programme to exit immediately. You will edit this code to add extra functionality.

You should add the following:

1. Basic flight simulation **[8 marks]**

For this, you will need to implement yaw, pitch and roll, mapped to the WD, AS and QE keys respectively. You will also need to implement speed control, with 0-9 setting speed in m/s directly, and -/+ decreasing and increasing, with a minimum of 0 and a maximum of 9.

You are not allowed to use the OpenGL matrix routines: the matrices should remain as they have been set in resizeGL(). Instead, you should compute the correct transformation matrices yourself, and pass them into the Render() routine of the HomogeneousFaceSurface class.

One matrix has been provided for you to get you started: this is because many modelling tasks assume that z is vertical in the world (WCS), but rendering assumes that z points out of the screen. This is (relatively) easily handled with a 90 degree rotation around X, as shown.

One note: the light direction needs to be modified by the rotation component of the view matrix, so you will have to get this working properly.

1. Particle system **[8 marks]**

Once you have the flight simulator working, you will notice a conical(ish) projection at coordinates (-38500, -4000). You should implement a particle system that ejects lava bombs vertically as shown in class, using Monte Carlo to randomise the direction within 45 degrees of vertical. These lava bombs should be ejected at between 60 m/s and 300 m/s, which will allow them to reach at most 4500m altitude. We \*will\* check to see whether you have set the correct values for this purpose.

You should render the lava bombs using the model and colour provided. They will expire when they reach zero altitude for the purposes of this assignment.

1. Collision Detection **[8 marks]**

Once you have the particle system working, you should address collision detection. There are four types of collision possible: between lava bombs, between lava bombs and the ground, between your plane and the lava bombs, and between the plane and the ground.

If the plane collides with the ground or the lava bombs, you should print a message and exit, because you are dead. To make your life easier, the Terrain class has a function getHeight() which returns the ground height for any (x,y).

If lava bombs hit the ground, they expire – note that this will require modifying the code from b).

If lava bombs hit each other, you may choose to implement any reasonable outcome.

1. Extras **[6 marks]**

**IF AND ONLY IF** you have implemented a)-c) correctly, you may add extra features to your simulation for the remaining marks. Please note that we do not expect most students to do so, as 80% of the marks are available for the other tasks, and marks over 80% are unusual in the UK system.

No advance ruling will be given on the value of the extras, but some possibilities include:

1. Other planes following simple repetitive flight paths, with collisions possible.
2. “Chase cam” – placing the camera above and behind your plane
3. Smoke trails for lava bombs
4. Interactions between lava bombs and the ground

All code should compile on the School's Linux machines without installation of any extra libraries or applications. You should include a makefile and a readme.txt file with any additional instructions.

**FILE NAMING:**

In order to make the marker’s job easier, please make sure that you rename the directory from A1\_handout to xxxxxx\_A1 where xxxxxx is your userID. For example, since my userID is scshca, I would rename it to scshca\_A1.

**PENALTIES:**

Poorly structured or badly commented code may be penalised by up to 25% of the marks available.

Code without a readme may be penalised by up to 10% of the marks available.

Code that does not compile properly will be assigned a mark of 0, but I will usually give the student one chance to correct this.

**DUE DATE:** Thursday, November 16, 2023, 10:00am