CS324 Computer Graphics Coursework

**Introduction**

My solution implements the toy train option. I used WebGL and javascript to complete the task. I used the glMatrix mat4 module to efficiently perform any matrix operations, I have included the module file in the zipped files.

**Functionality**

The train travels in a loop on the circular tracks. There are four different camera views accessed by pressing 1, 2, 3 or 4. The train can speed up, slow down, stop and travel backwards. This is done by pressing ‘>’ to increase forward velocity and ‘<’ to increase backwards velocity. There are 3 speed settings in each direction.

**Development**

Before anything else I had to get the html canvas and set the webgl context. Next I created vertex and fragment shaders and linked them to the program. I also made a position buffer to store the geometry vertex position data; a colour buffer to store the colour data; and a normal buffer to store all of the normal vectors for diffuse lighting.

The code for the vertex and fragment shaders is stored as global constant strings that are passed to a function that compiles them. I used diffuse lighting, getting the brightness of a surface by calculating the angle between the light source normal vectors and the surface normal vectors. This gave me the brightness which I then multiplied by the colour to brighten/darken the surface. In the code I used color rather than colour to keep things neat.

I primarily used gl.TRIANGLES to draw the geometry, although I used gl.LINE\_LOOP to draw the circular tracks. The train track is a simple circular loop which the train follows by being translated and rotated using vertex and angle data of a circle that is stored in a constant called track middle data. The speed and direction of the train is controlled by a tick system: every loop a frame variable is incremented if the direction is forwards or decremented if the direction is backwards. This determines the angle of rotation of each part of the train and also the translation around the track.

Each function that draws part of the geometry follows a general pattern. First, the location of each vertex is defined and loaded into the position buffer. Next, the colour buffer is bound and the colour of each vertex loaded in. Next, the model, view and projection matrices are calculated and multiplied together to create the final environment. The model matrix transforms the geometry. The view matrix places the camera in the world by moving the world around the source. The projection matrix defines the camera’s attributes such as fov, aspect ratio and cull distances.

The main program is looped by calling requestAnimationFrame(main) so the canvas can be constantly updated.

Word count: 458