COSC363 Assignment 1 Report

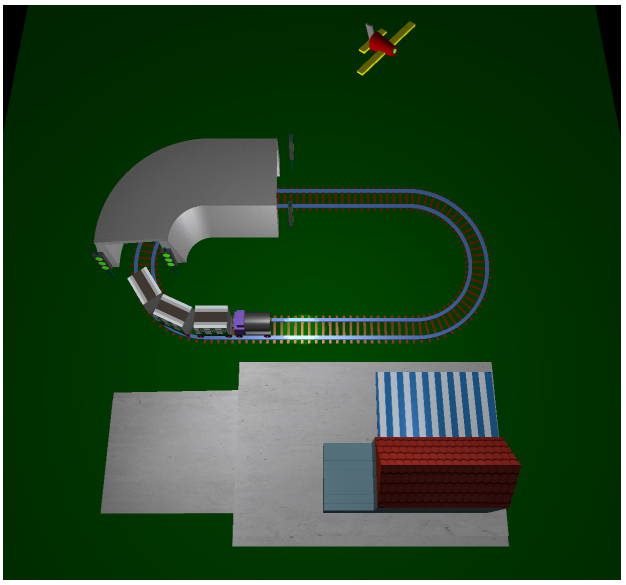
Name: MENG ZHANG
Student Number: 71682325

1. Project Description

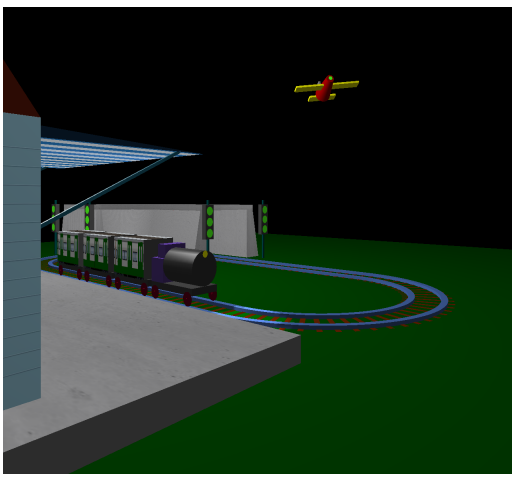
The project creates a 3D OpenGL scene, which mainly includes

- 3 static models:
  - 1 oval railway,
  - 1 tunnel,
  - 1 railway station, which contains:
    - A house with a sloped roof and a sun shade,
    - A floor base connected with a slope.
- 3 animated models:
  - 1 moving train followed by 3 moving wagons,
  - 1 flying aeroplane,
  - 4 flashing signal lights.

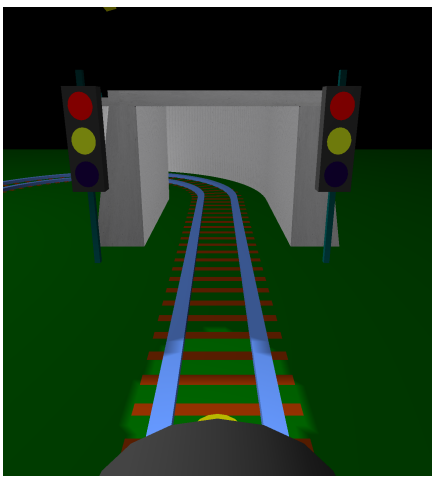
The train moves along the oval tracks, going through the tunnel, and stops a few seconds at the railway station in each lap. The aeroplane circles above the train and railway station.



Scene Overview



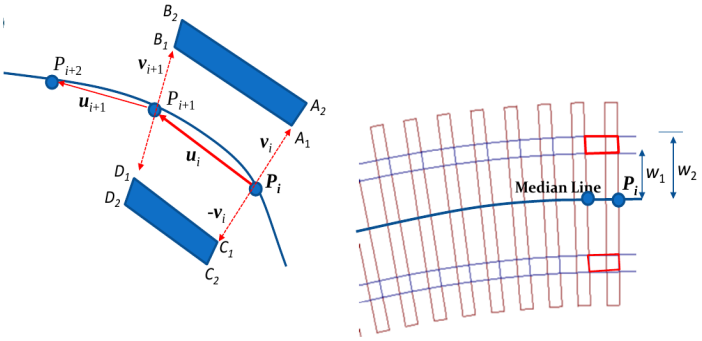
Station View



Cab View

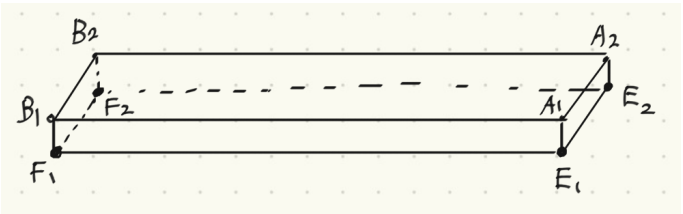
2. Extra Feature Description

- **Track enhancements (oval railway tracks):**
  - Load track coordinates from provided Oval.txt to two array pointers posX and posZ;
  - Create two parallel tracks according to each group of 3 consecutive vertices (P[i], P[i+1], P[i+2]) on the median line:



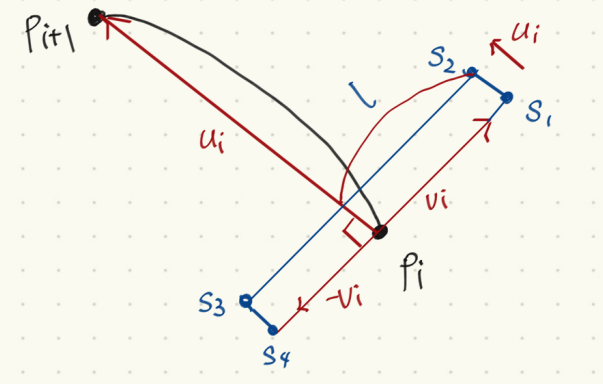
    - Let u[i] be the unit vector of P[i]P[i+1] and v[i] be the unit vector perpendicular to u[i]. Similarly, u[i+1] is the unit vector of P[i+1]P[i+2] and v[i+1] is the unit vector perpendicular to u[i+1].
    - To create the top quad of the two parallel tracks, let w1 = the inner width from median point P[i] to the track and w2 = outer width. The width of the track = w2-w1. Then we can get the coordinates of 4 vertices to create the GL_QUADS for the top:

      A1 = P[i] + v[i] * w1; A2 = P[i] + v[i] * w2;
      B1 = P[i+1] + v[i+1] * w1; B2 = P[i+1] + v[i+1] * w2. Similarly
      C1 = P[i] + (-v[i]) * w1; C2 = P[i] + (-v[i]) * w2;
      D1 = P[i+1] + (-v[i+1]) * w1; D2 = P[i+1] + (-v[i+1]) * w2.
    - To create the 2 vertical quads of the tracks, we can get the related vertex coordinates by using the four vertices of the top. The only difference is the height (y index).



      For example, let A1 = (x, y, z) and suppose the track height is 1, then E1 = (x, y-1, z).
    - Finally, we can create a unit track with 3 quads, i.e. A1A2B2B1, E1A1B1F1, E2A2B2F2, and all the coordinates of the vertices can be obtained according to the process above.

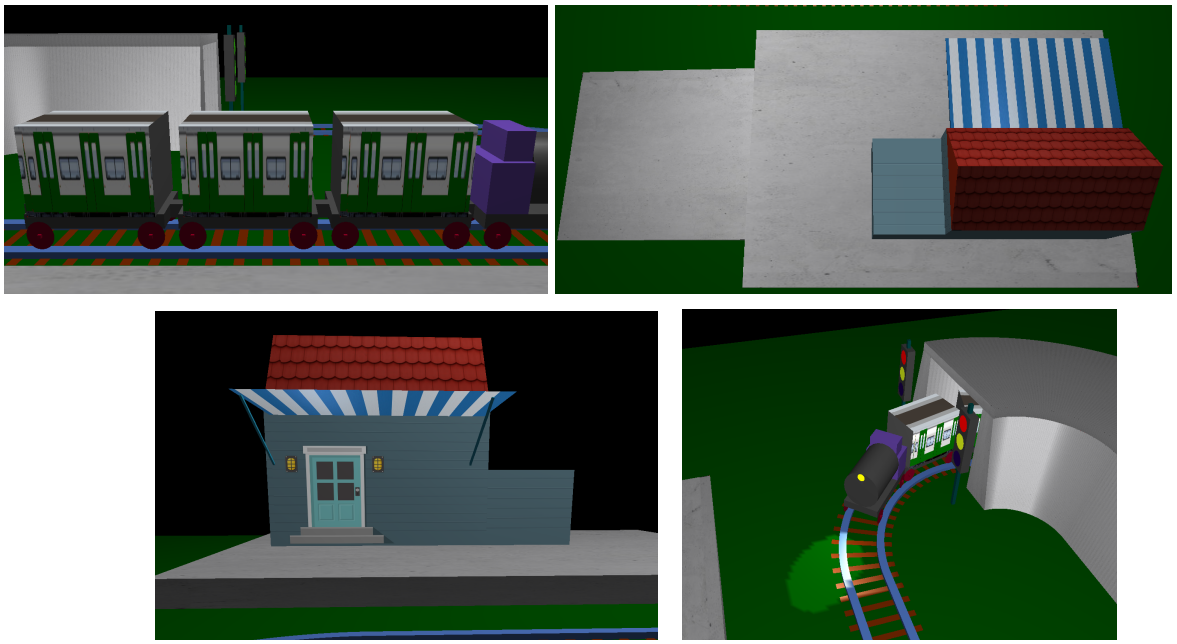
  - To create railway sleepers, let p[i] and P[i+1] be the 2 consecutive vertices on the oval median line, l = the length between P[i] and the vertex S1 of the sleeper where P[i]S1 is perpendicular to P[i]P[i+1].



Then S1 = P[i] + v[i] * l; S2 = S1 + u[i] * scaler;
S4 = P[i] + (-v[i]) * l; S3 = S4 + u[i] * scaler.

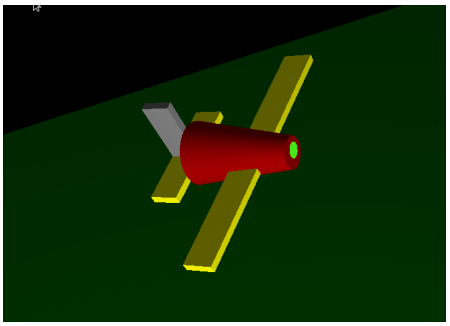
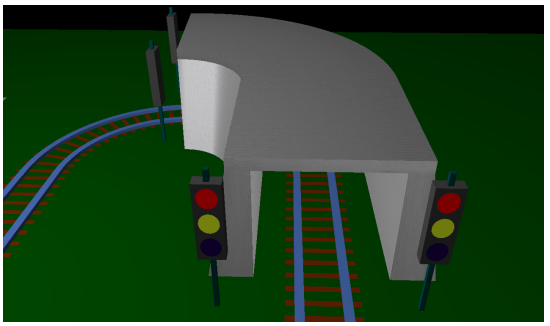
- **Train transformation on provided oval tracks:**
  - Set a variable THETA as the index of the vertices on the oval median line, and create a timer in which THETA increments by 1, unless the train gets to a specific vertex where the train needs to stop for a few seconds.
  - Get the train's current position by posX[THETA] and posZ[THETA].
  - To get the train's current posAngle to y axis, we first get the last and next positions P[i-1] and P[i+1] and calculate the unit vector uv(x, y, z) of P[i-1]P[i+1]. Then PosAngle = atan2(z, -x).
  - Finally we move the train by first rotating it by posAngle and then translate it to (posX[THETA], y, posZ[THETA]). Similarly, we translate the wagons to (posX[THETA - i], y, posZ[THETA - i]).
  - To optimise the process we can cache the posAngle for each vertex on the oval median line, by which we can also quickly get the posAngle of wagons by posAngleCache[THETA - i].

- **Model enhancements**



  - Wagon texturing
  - Station house texturing (including roof, walls and sun shade)
  - Station floor texturing
  - Tunnel texturing
  - Lights on engine

- **Scene enhancements**

  - Aeroplane: Circling above the railway and station
  - Signalling lights: Flashing between green, red and yellow.

- **Camera enhancements**
  There are 3 camera modes:
  - Free navigation view: user-control with keys to move and rotate camera
  - Railway station view: a fixed camera view from railway station
  - Moving cab view: A moving view going with the train. Camera is positioned above the current position of the engine and the look view is located in front of the position of the engine.

3. Control functions

- **"c"**: Switch between 3 camera modes;
- **Up arrow**: Move camera forward in the current direction;
- **Down arrow**: Move camera backward in the current direction;
- **Left arrow**: Change the current direction towards left by a certain angle;
- **Right arrow**: Change the current direction towards right by a certain angle;
- **Page-up**: Increase camera height
- **Page-down**: Decrease camera height
- **Home**: Go back to default view in free navigation

4. Build commands

In a Linux system, unzip the **mzh103_cosc363_assign1.zip** file and choose one of the following options to compile and run the program.

Option 1: Using CMkeLists.txt

- Go to src folder and start a terminal at the location
- Run: ***cmake CMakeLists.txt***
- Run: ***make***
- Start the program by command: ***./RailwayWorld.out***

Option 2: Using vanilla g++

- Go to src folder and start a terminal at the location
- Run : ***g++ -o RailwayWorld.out RailwayWorld.cpp RailModels.cpp -lGL -lGLU -lglut***
- Start the program by command: ***./RailwayWorld.out***

5. **References to sources**
    - [image] Front Door: https://www.freevector.com/front-door-30619
    - [image] Wallpaper Lines: https://www.wallpapersin4k.org/images/861807
    - [image] Concrete texture: https://stock.adobe.com/sk/search/images?k=polished+concrete+texture&asset_id=159146914