Lab02 Report

CS337-Computer Graphics, Autumn 2019.

* Name: Shao Yuming Student ID: 517030910218 Email: 1159326041@sjtu.edu.cn

1 Submitted Files

The developing environment is CLion on MacOS Catalina(10.15.1). The submit project folder contains important files:

- CMakeList.txt (necessary for CLion IDE)
- main.cpp (main program)
- car.hpp (implement a class representing a car and record its information, it can generate transform matrices for shaders accordingly)
- camera.hpp (implement a class representing the camera and record its information)
- loader.hpp & .cpp (load .obj files)
- shader.hpp (load shader files and construct program)
- shaders
  - simpleDepth.vs & .fs (render depth information into a texture, this is for drawing shadow only)
  - draw.vs & .fs (actually render the whole scene)
  - skyBox.vs & .fs (display the sky and groud texture)

2 Satisfied Requirements

Caution: Since the submitted files include a demo which can demonstrate my work to an enough extent, here in this report I will omit some unnecessary figures for simplicity. But I am sure that all the features mentioned here are shown in the video.

2.1 Car and road

In order to obtain .obj files for this lab, I used SolidWorks to build all the models myself. The road is a curved closed runway, as pointed out in the additional questions. As for the car, I refer to one of the models in the game as a template. Figure 1 is the template that I used, with which the .obj file is similar.
To construct the car in my project, I drew some parts of the car seperately

- car base
- wheels $\times$ 4
- steering wheel

Based on this implementation the wheels can rotate correctly while the car is running and changing direction. All the parts have different colors as well.

Figure 1: Template car in the game

2.2 Shadow

I successfully implemented the shadow, as is shown in Figure 2.

Figure 2: The shadow of the car

However, it seems that the shadow is ugly and interferes while we are viewing the objects. I set the function of showing shadow as follow: when the button B is pressed, there will be shadow. If not pressed, there won‘t be any shadow.

2.3 Control

Since the control pattern is not described in detail, I implemented it like this:

- W, S, A, D: move the position of the camera
- mouse: change the angle of the camera
- Up: make the car move forward, speedup until the limit is reached, or slow down while the car is currently moving backward
- Down: make the car move backward, speedup until the limit is reached, or slow down while the car is currently move forward

- Left, Right: if the velocity of the car is not zero, these keys will change the moving direction of the car. Besides, the steering wheel and two front wheels will rotate correspondingly

2.4 Resizing the window

Before and the after the user change the resolution of the display window, the car shape will not be distorted.

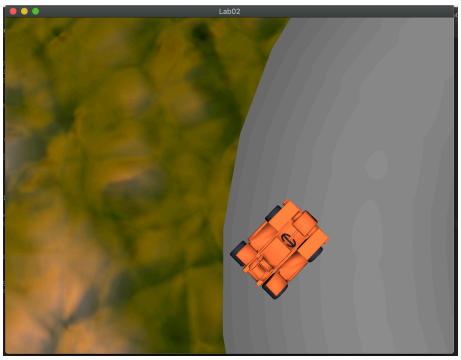

Figure 3: original window

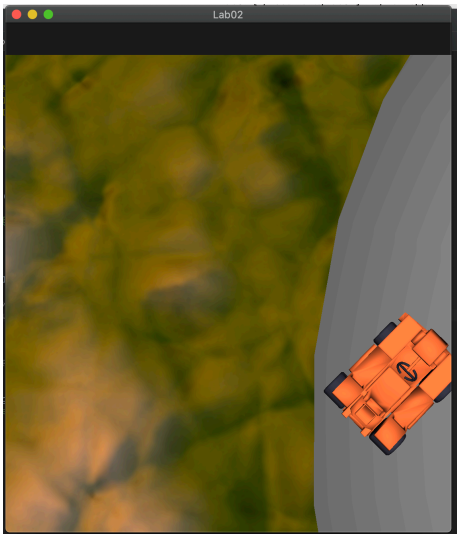

Figure 4: resized window

3 Satisfied Additional Questions

3.1 Curved closed runway

Shown and visible in the demo.

3.2 Texture maping on the ground, sky

Shown and visible in the demo. I used some figures downloaded from websites.

3.3 A Second View

As every who has ever played the game Running Kart knows, while she is controlling the car, she can see the back of the car and the view follows as the car goes. To reconstruct the behavior, I designed a second view mode similar to it, which is shown in the demo.

4 Design Ideas

As this lab(Lab02) is a natural extention of what we have done in Lab01, some of the utilities written while I was doing Lab01 can get used here, the function to load .obj files and calculate their normals, for instance.
However, within the process of this lab, I encountered numerous problems that I spent quite a long time to deal with.

4.1 Create the .obj models

In order to create a car by myself, I learned and used SolidWorks for the first time.

4.2 Assemble the parts into a car

Render the parts all together and make the car run correctly like real ones is quite a difficult task. I guess it is impossible to use absolution position, because of some computational deviation of different parts. This effect is not obvious at the very beginning, but will be annoying when the car runs for a considerable period of time.
The solution to this is to use relative position. Design a class representing the car, just record the positin of the car base. When it is time to render others such as steering wheel, calculate it position accordingly. This kind of computation is not easy as well, because I have to consider rotating coordinate system transformation, using many mathematical formulae.

5 Website

The project has been uploaded to github as well, and here is the address : Running Kart