Computer Graphics using OpenGL Coursework

Li Guo

220060950

# Overview

This project is a game with guns running following the route to shoot objects on the route to gain scores based on how many objects are shot. Player can also change the guns while encounter the other guns on the route. Figure 1 shows the screengrab of the top-down view of the game world. The following Table 1 shows all user interface controls of the game.

绿色的草地上

中度可信度描述已自动生成

Figure 1 top-down view of the game world

|  |  |
| --- | --- |
| F1 | Change game view between top view and free view |
| F2 | Change game view between side view and free view |

Table 1 user interface controls

# Route and camera

## Route

The route in this game is a road track created using Catmull rom, first creating the center line of the whole route, then extend the left and right offset line to create the side of the route, and finally using the vertexes on the sideline to create the texture to the route. Figure 2 shows all the control points of the center line, with the control points set up, the non-linear centerline can be created. Figure 3 shows the function to create the offset curve. The offset curve is created by using the TNB structure to first create the offset curve points for the offset curve, and then using the offset curve points to create the inner curve and outer curve points. Finally, the offset curve can be create successfully. Figure 4 shows the function to create the track of the route. It first connecting all the vertices on the inner curve and outer curve, and then apply the texture of the curve to the lines connected by both of the curve mentioned above.

电脑屏幕的照片上有文字

中度可信度描述已自动生成

Figure 2 control points for the center line

文本

描述已自动生成

Figure 3 offset curve function

文本

描述已自动生成

Figure 4 create track function

## 1.2. camera/viewing

There are three kinds of cameras in this game, including the top view camera, the side view camera and the free view camera. For the top view camera and the side view camera, they are using the function of moving along the center line to make sure they are following the moving object on the center line, and the position of the camera are adjusted as needed to make sure the camera view is at the correct track. Figure 5 shows the function that the objects or the camera can move along the center line in the certain speed. It first gets all the center line points as needed, and then calculate the next position the camera supposed to be based on the current position of the camera. In Figure 6 shows the function of changing the camera view using keyboard. When pressing F2, the camera view will change from the free view mode to the top view mode, and by pressing F3, the camera view will change form the free view mode to the side view mode. When in the top view mode or side view mode, when press F2 or F3, based on the mode of camera view, the camera view mode will change back to free view mode.

文本

描述已自动生成

Figure 5 move along center line function

文本

中度可信度描述已自动生成

Figure 6 change the camera view

# 2. Basic objects, meshes and lighting

## 2.1. basic objects

There are three kinds of basic objects in this game, including the sphere, the cube and the tetrahedron. Each kinds of basic objects are configured in different class. For the sphere, I am using the original template class. For the cube, I add an new class called CCube, in the CCube class, all the interleaved point, texture coord, and normal of the cube are set, then setting all the faces of the cube, and adding all the necessary into the VBO to upload the data into the GPU, and finally using Render() to draw the cube into the game world. For the tetrahedron, the same process is used to configure the tetrahedron data and draw the tetrahedron into the world. Figure 7 and 8 shows the final result of the basic objects in the game world.

图片包含 游戏机, 小

描述已自动生成

Figure 7 cube in the game world

黑白色的标志

低可信度描述已自动生成

Figure 8 tetrahedron in the game world

## meshes

There are four kinds of meshed objects in the game world, including the UZI, the smash, the laser and the pistol. All the models of the meshed objects are loaded from the .obj file and render all the objects into the game world using modelViewMatrixStack function. The following Figure 9, 10, 11 and 12 shows the meshed objects in the new world with proper position, which is on the route, orientation, and scale.

图片包含 草, 武器, 户外, 水

描述已自动生成

Figure 9 Smash in the game world

图片包含 游戏机, 绿色

描述已自动生成

Figure 10 UZI in the game world

图片包含 草, 武器, 户外, 绿色

描述已自动生成

Figure 11 laser in the game world

图片包含 草, 武器, 绿色, 前

描述已自动生成

Figure 12 pistol in the game world

## lighting

There are three kinds of lighting in the game world, including red, blue, and green lighting ball. Each kind of lighting effect is configured in different shader file, for example, for the blue lighting effect, as shown in the Figure 13, the color of this fragment effect is defined as an vec4, and put the required color to the FragColor, the RGB is defined as (0, 0, 1) in the BlueLightingShader.frag file, then this fragment effect is added to the sphere in the game world to render the sphere in the game world when the lighting of the game world is set as black, which is shown in the Figure 14. The Figure 15 shows the final effect of the lighting effect in the game world.

电脑屏幕截图

描述已自动生成

Figure 13 BlueLightingShader.frag file

文本

描述已自动生成

Figure 14 render sphere with lighting effect

黑暗中有圆形

中度可信度描述已自动生成

Figure 15 lighting effect in the game world

# 3. HUD, gameplay and advanced rendering

## 3.1. HUD

There are two kinds of HUDs add in the game frame, including the FPS and the Time passed while the game has started. For the FPS function, the elapsed time and frame count parameters are used to count the framesPerSecond parameter in order to count the FPS while the game is running, and using the elapsed time to calculate the time passed while the game is running, shown in the Figure 16. Figure 17 shows the outcome of the FPS and game time in the game world.

文本

描述已自动生成

Figure 16 Function for FPS and Gametime

文本

描述已自动生成

Figure 17 FPS and Time in game world

## 3.2. Gameplay

There is no game play function for now. There would be the A and D to control the gun on the route to move left and right, when the gun is detected to have collision with other guns on the route, the player can choose to change the gun or not while playing. player can also shoot the gun to fire bullets to shoot the cubes or the tetrahedrons on the route to gain score to have higher result of the final score.

## 3.3. Advanced rendering

There are two kinds of advanced rendering effects in the game world, including the fog and toon shader. For the fog effect, the fog color and density are configured in the mainShader.frag file to add the fog effect to the skybox other then all the other objects in the game world. As shown in the Figure 18, the fog effect fog effect is configured with fogFactor and fragColor while the skybox is rendering, if the skybox is not rendering, the parameters used above will not be used. Figure 19 shows the final effect of the fog effect.

For the toon shader effect, this function is not successful at the time. I have added the ToonShader.frag and ToonShader.vert, which is shown in the Figure 20 and 21, for the fragment file, the toon shading parameters are set to give the final fragment color by using the toonColor parameter. And in the vertex file, the function is to add the toon shader effect to the objects based on the position of the objects and the position of the camera. The final rendering is not successful to the cube which is added with the toon shader effect, due to the flaw to the toon shader function.

文本

描述已自动生成

Figure 18 fog effect in the mainShader.frag

图片包含 户外, 草, 站, 男人

描述已自动生成

Figure 19 final effect of the fog effect

文本

描述已自动生成

Figure 20 toon shader fragment file

文本

描述已自动生成

Figure 21 toon shader vertex file