**Interactive 3D Maze Game Project Report**

**1 Design Outline**

An interactive 3D maze game is implemented based on the skeleton code provided together with the project description. The workflow of the program can be described as follows:

1. load the map: readmap()
2. initialize the game: GL initialization and init(), which includes
   * 1. initializing the player's information: initplayer(),
     2. loading the texture maps using the utility functions TextureLoadBitmap and the provided bitmap.h and bitmap.c,
     3. initializing fog effect, and
     4. initializing the material and lighting effects
3. start the GL main loop, which iteratively
   * 1. display the ground, the walls, the exit door and the player: display()
     2. handle the keyboard events that control the display settings
     3. handle the mouse events that control the direction and position of the player
     4. in the GL idle fallback, move the player based on the player information, and resolve collisions between player and the walls.

My own implementation is all in the file "project.cpp".

**2 Key code fragments of algorithms**

**Load the texture map**

Texture maps and their width and height are defined as global variables by



The texture maps are loaded in the init() function by

****

**Draw the ground: DrawGround()**

The floor is simple a rectangle at y=0 plane. The key code fragment for texture mapping the floor is



**Draw the walls and exit door: DrawWalls()**

The walls are drawn by looping over the map. There are two types of walls: parallel to x or parallel to z, so two types of rectangles are created for them respectively.

****

The exit door is drawn in a similar way.

****

**Draw the player**

The player is drawn using a sphere to represent to mouse's body and two nearly flat cones to represent the ears. When the player is moving forward, the ears are rotated by -30 degrees randomly to reflect the movement.

Similarly, in the wired (debug) mode of DrawSphere()



**Collision Detection**

Collision is detected by checking if the player's position after the move (dx, dz) overlaps with the wall. Four coordinates in the map are checked: (lx, z), (ux, z), (x, lz), (x, uz). If the move in x overlaps map[lx][z] or map[ux][z], then we set the move dx to 0. Similarlly, if the move in z overlaps map[x][lz] or map[x][uz], we set the move dz to 0.

****

**Material and Lighting**

Material and lighting are set up in the init() function



**Fog Mode**

Fog effect is initialized in init() by

And it can be turned on and off by the key "f" or "F" through the handling in the keyboard() function:



**3 How to use the program**

The program should be launched with a maze map file. For example,

"project.exe map.dat"

The map .dat file stores the maze information as described in the project description. After launching, you can use the mouse to control the direction and movement of the mouse. Clicking left or right button alone means changing the direction of the player to the left or right. Holding the left and right button together means moving the player forward.

When you find and cross the exit door, the program will pause for one second and print out "You Win" in the command console. After that, the player will return to the initial position and you can play the game again.

The key "S" or "s" can be used to switch the display of the player between solid mode or wired mode.

The key "F" or "f" can be used to turn on and off the fog effect.

**4 Experimental results**

We show 9 screenshots of the game to demonstrate our experimental results:

1. At launch
2. Pressing "s" shows the wired player
3. Go to the corner of the wall, the player stops
4. The exit door
5. By default we have the fog effect turned on
6. Pressing "f" turns the fog effect off
7. Increasing the fog effect's brightness, this requires changing the source code and recompile.
8. Changing the color material of the player, this requires changing the source code and recompile.
9. Changing the position of light0 and its diffusion property, this requires changing the source code and recompile.

|  |  |
| --- | --- |
| 1 At launch | 2 Wired player |
| 3 Collision detection | 4 Display of the exit door |
| 5 Fog effect (default) | 6 Without fog effect |
| 7 Fog effect with brighter fog color |  |
| 8 Different player material (color) | 9 Changing the light position and diffusion property |

**5 Feeling and opinions about the project**

This is a very interesting project. It demonstrates how a 3D game is programmed from scratch. Although it is a very simple game, it excites me a lot when I added the features piece by piece following instructions and eventually make the game look real.

The skeleton code demonstrates very well the OpenGL game workflow. Without the skeleton code I cannot imagine finishing such a project a possible task. By working with the skeleton code, extending it, and referring back to the lecture notes, I got much better understanding of the OpenGl workflow, design of the scenes in 3D, texture mapping, collision detection, keyboard and mouse interaction, camera, lighting, and materials. All the knowledge I learned from the lecture now links together and the whole picture becomes clear. The practice and debugging helps a lot in the learning process and make the knowledge solid.

Sometimes the explanation on certain GL functions from the internet is not very easy to understand, for example, the combination of lighting and material. In those cases, I experimented different combination of functions and parameters to see which one can achieve the effect I wanted. Through those debugging process and tests, though I could not say I have fully understood the topic, I am proud to say that I am able to achieve the desired effect I want.

The game itself might be a simple one. But it clearly shows the room for extension. For a 3D game player like most of us students, it could not be more exciting to extend it by adding more features beyond the project. On the top of my mind we can improve the control of the mouse movement by the keyboard and allow the user to change the camera position and view angle interactively, to mimic the majority of the 3D online games. Secondly, we could add timer to the game. A random generation of the maze map would also be interesting.

In summary it is a very nice project that allows us to learn knowledge from practice. And it rewards us fun while learning!