Bruno Veiga - bsv259

Xavier Barrios - xb375

Maze of Doom

**Overview**

The project is a 3-Dimensional maze walkthrough. A player starts at one of the corners of the maze, and can walk through the maze by navigating around the many walls. The walls are setup in a way that makes it easy enough for the player to walk around, and in the center of the maze there is a fountain of water. There are 3 different views that the player may utilize to move his character around. The “player” mode sets the camera right behind the character figure. The “bird’s eye” mode sets the camera at the center of the maze, well above the scene, but the center constantly follows the character figure. Lastly, the “top-down” mode sets the camera directly above the character figure and follows him around.

**Implementation**

There are four major parts to this implementation.

1. The construction of the maze walls and floor
   1. In order to construct the walls we first created a function to draw a single 10x10x1 rectangular prism. Then we used that function in our draw\_maze() function to create the multiple walls while rotating and translating pieces as we needed them.
   2. For the floor we used a flat square shape laid out as 90x0x90
2. Building the hierarchical model
   1. Our hierarchical model consists of 12 separate shapes that move together when we move the character figure. We have a head, torso, left upper and lower arm, right upper and lower arm, left upper and lower leg, right upper and lower leg, and left and right feet. While the head is made from a sphere shape, all the other pieces of the hierarchical model are made from rectangular prisms.
3. Texturing the figures
   1. In order to give a more realistic feel to our project, we needed to apply textures to our scenery. To do that we used a function to load the textures from a directory inside our project folder into openGL, and then apply them to their corresponding shapes. In our draw\_maze() function we load and set the texture to be mapped right before we draw the maze walls and then load another texture for when we draw the maze floor.
4. Setting the camera and figure movement
   1. Perhaps one of the toughest challenges in building our project was creating character movement and then setting the viewpoint in such a way that it would follow the character and turn just as he does. To complete this milestone we mostly changed how character movement affected the gluLookAt() function. We used three different view modes.
      1. Player Mode - In this mode the “eye” is an imaginary camera set a few units behind the character figure with the center as a point on the character’s head, that way the camera would always follow the character. As the character turns we reset the starting eye position to a point behind the character again.
      2. Bird’s Eye Mode - In this mode the “eye” position is permanently set to the middle of the maze from a higher point of view with the center as the character’s head again. This allows the player to see the character moving around, but from a slightly farther position.
      3. Top-Down Mode - In this mode we set the “eye” position directly above the character figure but a few units away. The center in this case is the character’s head, just as in the other modes. This allows the player to follow the character around, but also view a much larger portion of the maze as well.

**Compiling and Running the Program**

Compiling the program may depend on your your operating system. All is compiled and run through the shell.

In Linux we use the following commands to compile then run:

g++ -o maze maze.cpp -lglut -lGL -lGLU

./maze

In Mac OSX we use the following to compile then run:

g++ -framework GLUT -framework Cocoa -framework OpenGL maze.cpp -o maze

./maze

**Bugs and Issues**

Some Bugs that we are aware of with the program are:

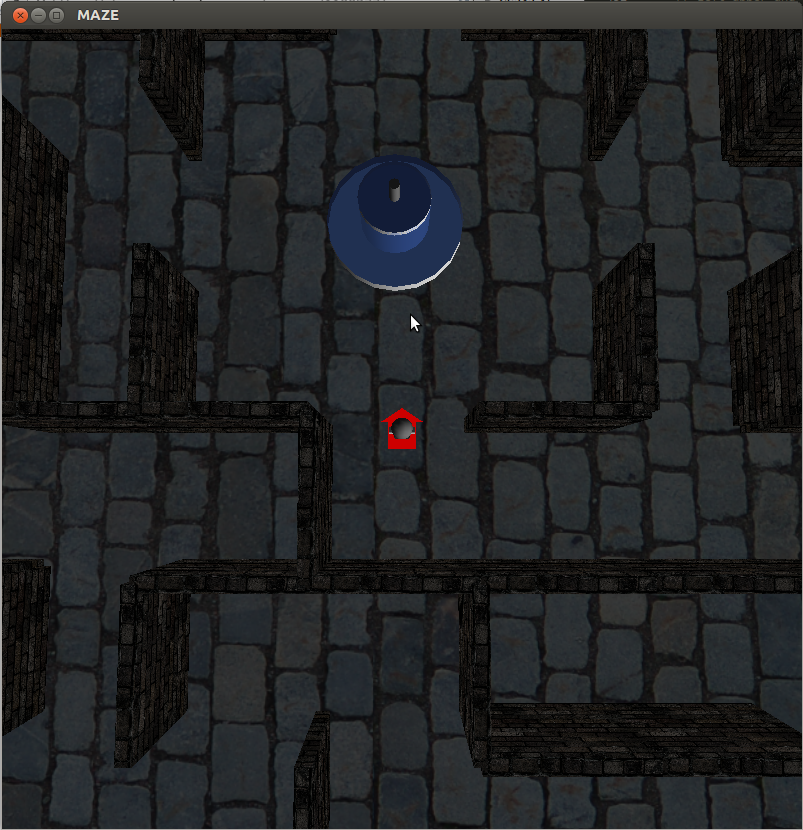
1. The lighting in our project does not reach all walls in the maze. After much trial and error we could not get a light positioned correctly in such a way that would light up the entire scene, even with multiple sources of light.
2. The fountain in the middle doesn’t display part of one of the shapes. We have a gray cylinder in the center of the fountain that is partly not displayed. It has to do with blending and transparency, but even though transparency is disabled before that shape is drawn, it simply doesn’t appear.

**Improvements and Future Work**

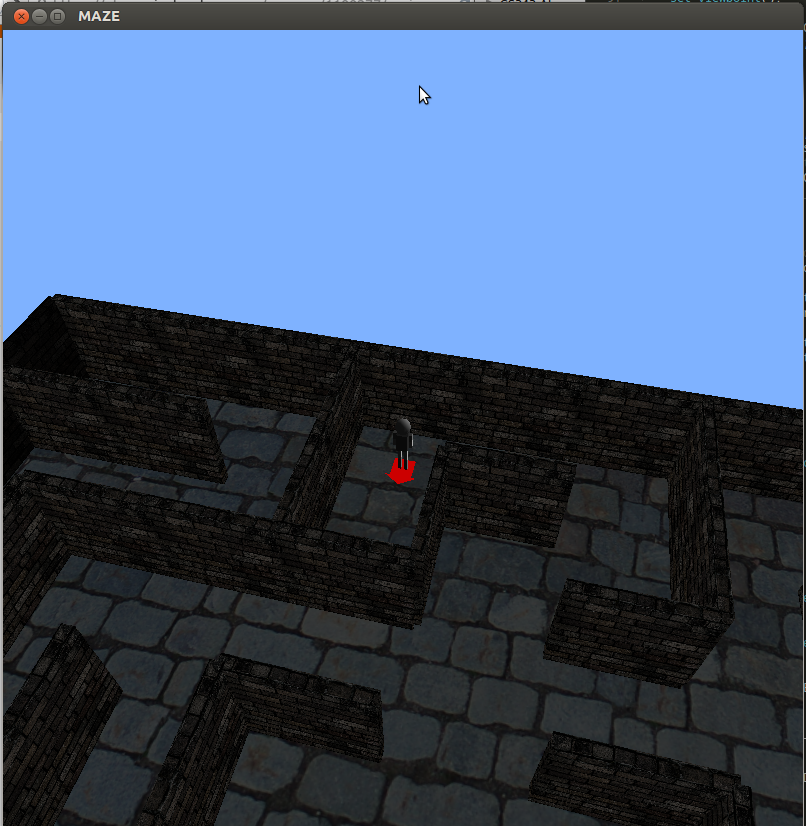
Some things we could add to the program in order to improve it are:

1. Wall collision - In the future we’d like to add wall collision to the project. that way the character can’t go through walls to navigate through the maze.
2. Walking animation - Another big piece that would improve our project even more would be to add animation for each body part. Instead of having the whole character slide forward, we would have the legs and arms move just like a person would.
3. Texturizing each shape in the character figure - this addition might be a little bit tougher to add, specially on the head of the character, but it would certainly improve the look of the project and give it more realistic feel.
4. Rotation of the character and camera - Currently the camera position and character are rotated immediately by 90° which is very jarring and disorienting when the camera is in “player mode.” A smoother transition would improve the illusion of the character “turning.”

**Screenshots**

PLAYER\_MODE view TOP\_DOWN\_MODE view



BIRD\_MODE view

**Bibliography**

OpenGL Software Development Kit. <https://www.opengl.org/sdk/docs/man2/>

CS354 Computer Graphics. <http://www.cs.utexas.edu/~amontoya/>

OpenGL Programming Guide. <http://www.glprogramming.com/red/chapter05.html>

Wooden’s OpenGL Tutorials. <http://www.falloutsoftware.com/tutorials/gl/gl8.htm>