**Labyrinth Specifications**

Christine Johnson

Eric Klukovich

Thomas Rushton

CS 480/680 – Computer Graphics

November 24, 2014

# Overview

## Program Requirements

The objective of this assignment was to create a 3D simulation of Labyrinth that is as realistic to the actual game as possible, yet still being a playable and an enjoyable experience for the end user:

* Load game board, and ball
* Basic physics for ball motion
* Load realistic textures for each object
* Movable camera
* Keyboard controls
* Multiple menu items - Restart, Pause, Resume, and Exit
* Display scores with text on the screen
* Multiple balls
* Multiple levels

## Team Members

Eric Klukovich – CS 680

Christine Johnson – CS 680

Thomas Rushton – CS 480

## Extra Credit

### Undergrad Only Extra Credit (Thomas Rushton)

#### Multiple balls

The ability to add an additional ball to the game is an option in the menu. If one of the balls fall into the hole then the game will reset from the beginning. Only one ball has to go in the finish hole in order to move to the next level.

#### Multiple Levels

Once the user completes the first level, then the second level will be automatically loaded. The user also has the option to change the level at any time in the menu.

### Undergrad and Graduate Extra Credit

#### Sound Effects

Sound effects are implemented using the irrKlang audio library. Sound effects include: background music and sound clips for when the ball falls into a hole or makes it to the goal.

# User Manual

## Compiling and Running the Program

## Open a terminal and change to the Assignment11 folder.

## Navigate to the "build" folder (cd build)

## Once in the build folder, simply type "make".

## Navigate to the "bin" folder. (cd ../bin)

1. Once in bin, simply type "./Labyrinth"

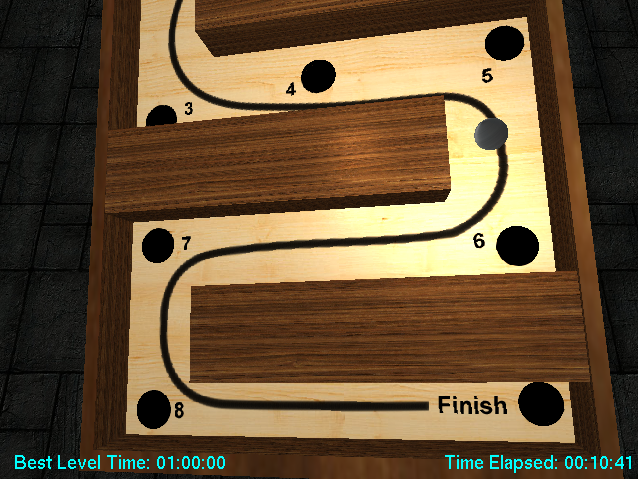
## Controls

## Keyboard

|  |  |
| --- | --- |
| **Key** | **Description** |
| **w** | **Tilt Board Forward** |
| **a** | **Tilt Board Left** |
| **s** | **Tilt Board Down** |
| **d** | **Tilt Board Right** |
| **Up Arrow Key** | **Zoom In** |
| **Left Arrow Key** | **Move Camera Left** |
| **Down Arrow Key** | **Zoom Out** |
| **Right Arrow Key** | **Move Camera Right** |
| **ESC** | **Exit the Program** |

Press keys 1-5 toggle different lights on/off:

|  |  |
| --- | --- |
| **Key** | **Description** |
| **1** | **Toggle Ambient Light On/Off** |
| **2** | **Toggle Diffuse Light On/Off** |
| **3** | **Toggle Specular Light On/Off** |
| **4** | **Toggle Point Light On/Off** |
| **5** | **Toggle Spotlight On/Off** |



**Figure 1: Level 1**



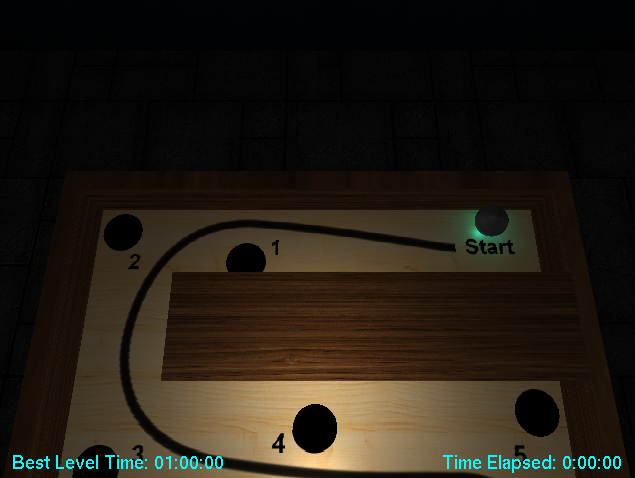
**Figure 2: Ambient Light Turned Off.**



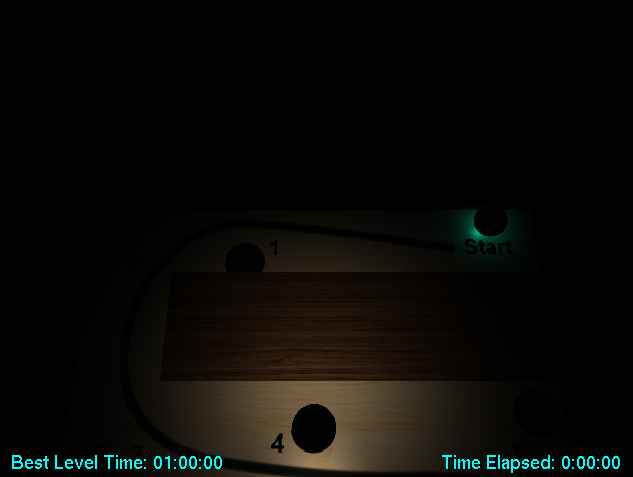
**Figure 3: Diffuse Light Turned Off.**



**Figure 4: Specular Light Turned Off.**



**Figure 5: Point Light, Spotlight and Specular.**

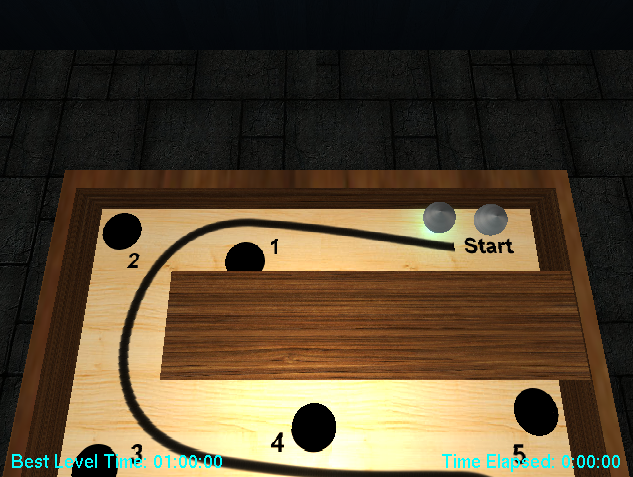


**Figure 6: Spotlight and Specular Light.**

## Menu

To open the menu, press the mouse right-click button.

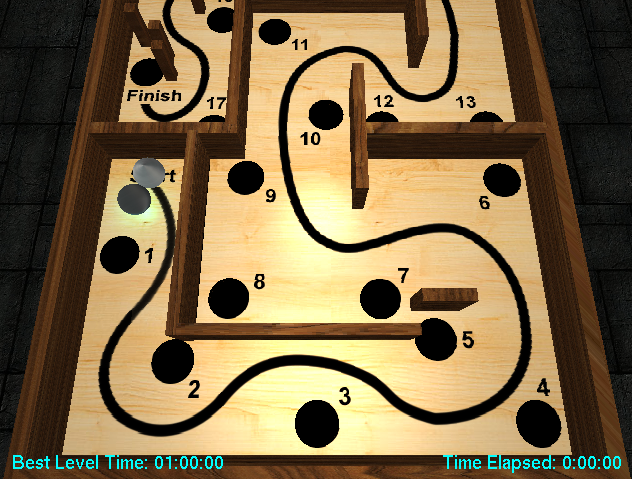
|  |  |
| --- | --- |
| **Menu Option** | **Description** |
| **Toggle Between One/Two Balls** | **Switch between one or two balls in the board** |
| **Pause Game** | **Pauses the current game** |
| **Resume Game** | **Resumes the current game** |
| **Restart Game** | **Starts a new game** |
| **Change Level** | **Toggle between level 1 and level 2** |
| **Quit** | **Exit the program** |



**Figure 7: Level 1 - Multiple Balls.**



**Figure 8: Level 2.**



**Figure 10: Level 2 - Multiple Balls.**

# Technical Manual

## Difficulties Encountered

### Tilting the Board

When we first tried to tilt the board, our first approach was to apply torque forces in order to tilt the board. The issue with this method is that the board had to be a moving body, which changes how Bullet creates the mesh. A moving body mesh does not construcd the collision object correctly. As a result, we used a static body and applied angular velocities to it and limited the rotation angle.

### Rotating the “Reset Ball” Plane with the Board

After we got the board to tilt correctly, our next challenge was to make the ball reset when the ball falls through a hole. In order to do this, we created a collision plane underneath the board and used a collision callback to reset the ball to the initial position. We had an issue because we were trying to use a static plane object, which did not respond to any forces applied to it. We then used a collision box object that allowed for forces to move it. Once we made this change then everything worked correctly.

### Camera

We wanted to create a dynamic camera that would allow for the user to always have a clear view of the ball and the holes when they are playing. We initially had an overhead view of the board, but some of the holes were hard to see. The camera adjusts it position when the ball exceeds some boundary. This involved tedious adjustments of the camera position in order to get the camera to work properly.

## What we would do differently

### Make a Class for Labyrinth

If we had more time and planned our project out better, then we would have liked to have made a labyrinth class because we had to implement multiple levels. This class would have made resetting, storing all the level data, and other setting much easier.

### Advanced Camera

If given more time, we would have liked to create a more flexible camera with multiple viewing options that would be applied dynamically. One option would be to follow the ball and would turn as the ball turns.

### Add advanced features

If given more time, we would add more advanced features such as a splash screen, more themes, and game replay.

# Video Game Party Poster

