# Labyrinth Guide: Graduate Student 680

Liesl Wigand, Khoa Tran

November 25, 2013

# 1 Overview

A Labyrinth, sometimes called a maze, is a set of passages or paths one must traverse to reach a goal. However, this description of a labyrinth isn't completely accurate. A labyrinth is a non-branching path that leads to the center (figure 1). A maze, although similar, consists of non-branching or branching path which may contain dead-ends or alternate paths to a goal.

Figure 1: Labyrinth design. `http://en.wikipedia.org/wiki/File:Con\%C3\%ADmbriga_minotauro.jpg`

For this project, 2 mazes and 1 labyrinth were created for the game Labyrinth. In this game, the objective was to design a maze and guide the ball(s) to a goal by tilting the entire maze. While trying to guide the ball through the maze, one must dodge the pitfalls that hinder the progress towards the goal. To move the ball across the board, the user can use the knobs to tilt the board with respect to the center of the board. Other requirements for this project are listed in the following section.

## 1.1 Features

**Labyrinth Requirements:**

1. Board, ball, and knobs
2. Texture on board and ball
3. Maze on board
4. Holes/winning position marked
5. Basic physics
6. Multiple lights
7. Movable viewpoint (camera)
8. Display score
9. Menus (start, quit, pause)
10. Colors and textures

11. Controls to tilt the board
12. Clock (for scoring)
13. Multiple balls
14. Multiple levels

**Extra Credit:**

1. Sky box (the background).
2. Swap lighting between dark and light.
3. Play with or without glass top on board.
4. Contain both a maze and a labyrinth board.
5. Adaptive preset overhead camera.
6. History of top 5 scores.

# 2 User Manual

This section contains a list of software requirements, an installation guide, and directions for compiling and running the program. It also includes screenshots and controls to be used in the game.

## 2.1 Getting Started

1. **Software Requirements** If any of the libraries are missing, please run the commands or follow the instructions listed below each set of libraries.

   (a) cmake, GLM, glew, glut:
   *sudo apt-get install freeglut3-dev freeglut3 libglew1.6-dev*
   *sudo apt-get install libglm-dev*

   (b) devIL:
   *sudo apt-get install libtiff-dev libdevil-dev*

   (c) Assimp 3.0, and Bullet.
   For Assimp, go to: *http://clanlib.org/assimp-linux.html*
   For Bullet physics, go to: *http://bulletphysics.org/mediawiki-1.5.8/index.php/Installation*

2. **Compiling and running Labyrinth**

   (a) Run cmake and make in the build folder:
   cd build
   cmake ..
   make

   (b) Execute the program in the bin folder:
   cd ..
   cd bin
   ./pa

After executing Labyrinth, the game will start in default mode (Figure 2).

1. **Default Mode:** Camera fixed to board center, dark mode (no sky box, spot lights only), and square maze with 1 ball.
2. **Spot Light:** On top, there is a spot light drawn as a sphere, that will follow the ball on the board.
3. **Goal Box:** The goal is the gray box on the other side of the board.
4. **Knobs:** The knobs and rotation are controlled from the keyboard.
5. **Menu:** To change to a different mode, press the right mouse button to bring up a menu.
6. **Clock:** The clock across the upper half of the screen is constantly running until the game is won or reset. The time is displayed as: (Hours:Minutes:Seconds.milliseconds).


Figure 2: Startup screen upon running Labyrinth

## 2.2 Game Controls

The game controls are mostly implemented using the keyboard. The mouse right button is used to access the game menu.

**Board Tilting Control Keys:**

1. **a or A:** Tilt the board left.
2. **d or D:** Tilt the board right.
3. **w or W:** Tilt the board up.
4. **s or S:** Tilt the board down.

**Other Keys:**

1. **Up/Down Arrow (↑↓):** Changes camera height from the board center.
2. **Up/Down Arrow (←→):** Changes camera rotation from the board center.
3. **Page Up/Page Down:** Changes camera distance from the board center.
4. **Home:** Reset camera to default position.
5. **o or O:** Switch to overhead ball adaptive camera mode (figure 3). **Note:** This camera mode will adapt to the first ball loaded onto the board. For multiple balls on the board, it's best to play in default camera mode.
6. **p or P:** Return to default camera mode (figure 2).
7. **Mouse Right Click:** Access the game menu.


Figure 3: Overhead camera adaptive to ball location

## 2.3 Menu Options

1. **Changing the Light Settings**
   There are 2 different lighting settings that can be accessed from the menu (Right mouse button).

   - **Bright Mode** - The sky box becomes visible, the ambient lighting is increased and the spot light(s) remain on the ball(s).

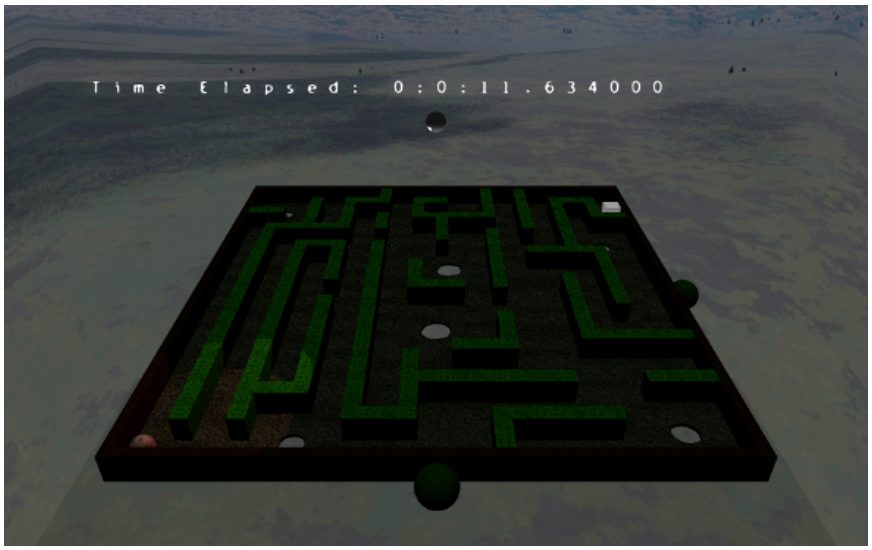


Figure 4: Bright Mode

   - **Dark Mode** - Darkens the sky box, decreases ambient lighting and makes the spot lights more obvious.


Figure 5: Dark Mode

2. **Changing the Sneeze Guard Settings**
   By default the board has an invisible top on it to prevent the ball from escaping the maze. This top can be removed from the menu (Right mouse button).

   - **With Top Mode** - The top fits over the whole maze and the ball can only escape through holes (or errors).
   - **Without Top Mode** - The ball can drop over the edge or be tossed through the maze. This provides an interesting way to speed progress through the maze, but is also more dangerous.

3. **Switching Game Boards**
   There are 3 boards in this game and they can be accessed from the menu.

   - **Maze 1:** Maze 1 is the default square maze board. The goal is to get the ball to the goal marked by a gray box on the right side of the board. The ball begins in the lower left; the second ball appears in the middle of the far side of the board.

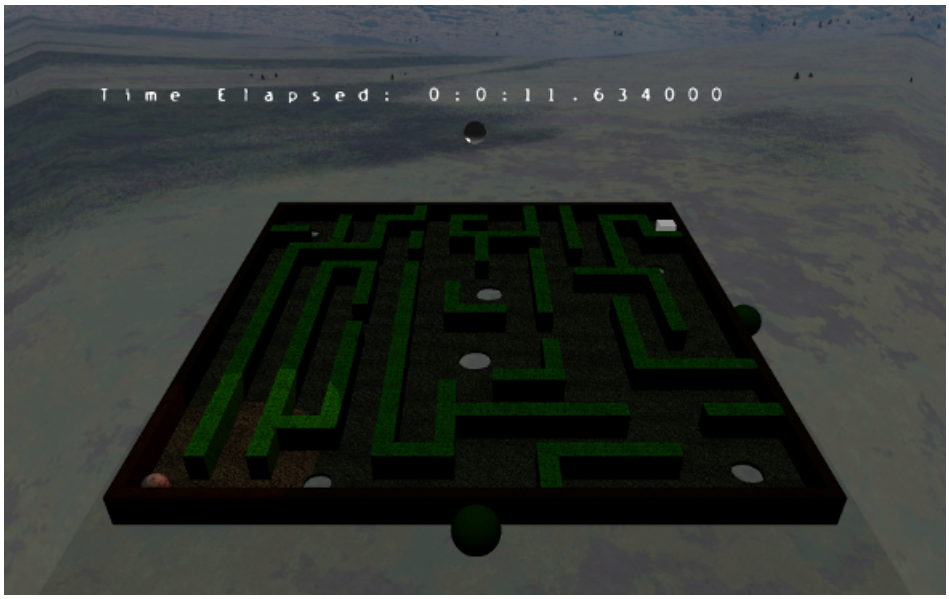


Figure 6: Default Maze or Maze 1

   - **Maze 2:** Maze 2 is a round maze board. The goal is the gray box at the center of the board, and both balls appear near the far side of the board.


Figure 7: Labyrinth - Maze 2

- **Maze 3:** Maze 3 is the "easy" board which is without walls. This makes the goal both easier and harder to reach.


Figure 8: Easy Maze - Maze 3

4. **Changing Number of Balls on the Board**
   The default mode will only have one ball on the board. Adding a second ball to the board can be done from the menu. With two balls on the board, both balls must make it to the gray box goal on each board to win the game. Once a ball has reached the goal, it is removed from play, which makes it much simpler to reach the goal with the second ball. This was a design choice, which is discussed further in the technical report.

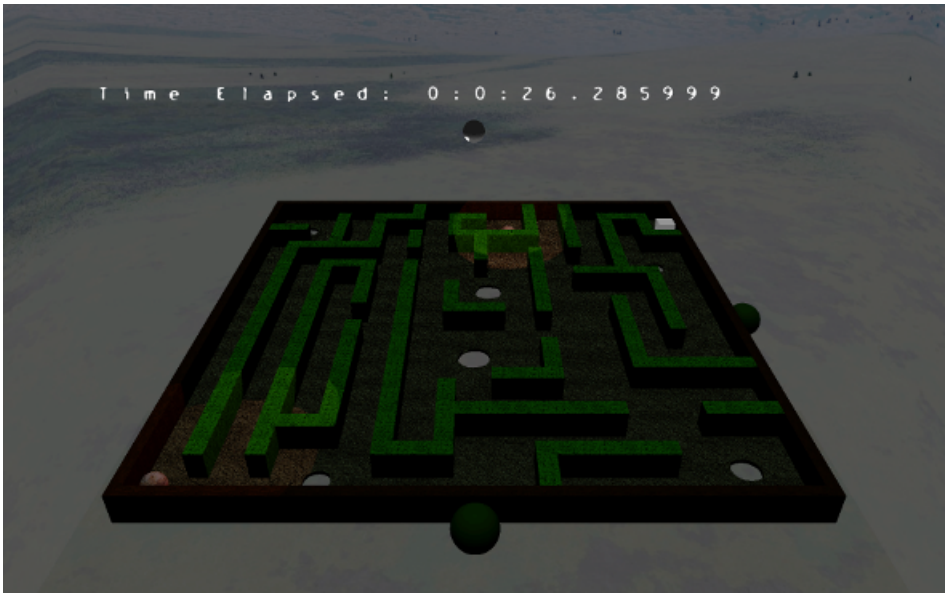


Figure 9: Board contain 2 balls

5. **Game Status**
   After the user has beaten the game, a score is display in addition to the top 5 scores including the current score ranked. The score is the time taken to reach the goal, displayed as (Hours:Minutes:Seconds.milliseconds). The clock will continuously run until the game is reset for a new game.


Figure 10: Game Over screen with score

# 3 Technical Manual

1. **Game Physics:** Moving the board in the physics world quickly would cause the ball to fly through the board or off of it. The solution to this problem was placing another invisible board on top. This makes the game much easier.

2. **Board Movements:** Including the goal block introduced an interesting problem with the board movement. The goal block didn't move in sync with the board when the board was tilted or moved from the user's input. The problem was, rotations were set as quaternions then switched to Euler, and this caused some issues with the speed of rotation, causing the goal to move at a different rate than the board.

3. **Textures:** There were some texture images that didn't load correctly. The problems with loading some of the textured images were inconsistent across size, color, and format. For example, the sky box loaded perfectly in Blender, but when loaded in OpenGL, the top and bottom faces were swapped. There were also certain images that appeared as a colorful rainbow pattern rather than, say, a grey brick wall. It is unclear what caused these errors, since the switching in a different image fixed this error, but the images were the same resolution and type.

4. **Resetting:** Resetting objects in the physics world continued to cause problems. If objects were not all reset at once and the world updated to reflect their new locations, the physics world would behave as though the objects remained in their old positions for several steps, or for some objects but not others. There was some confusion about whether the position of these objects should be set in the RigidBody or the MotionState, and whether after this was set, the object needed to be removed and added back into the PhysicsWorld. There were also issues with scaling of the objects: Bullet expects scale to be set in the CollisionShape, but if the btTransform is set using setFromOpenGlMatrix(), and the provided matrix was the complete model matrix (including scale), this causes odd problems where scale is sometimes applied twice, and sometimes not at all depending on whether an object is dynamic and set from the world, or static, and being set by the user's input.