Labyrinth

Bandith Phommounivong
Eric Bryant
Michael Mills

25 November 2013
CS480: Computer Graphics

Overview:

Welcome to Labyrinth, made possible through the open source libraries of OpenGL, Assimp, and Bullet. Our Labyrinth game is a fully functioning labyrinth simulation game in which you race against the clock while navigating your ball through a complex maze.

Extra Credit completed:

* * Multiple Balls
* * Multiple Levels
* * Top 10 Scoreboard

User Manual:

How to begin:

To compile: From the terminal, navigate to the build directory of the project. Type make. The program should compile without errors.

To run: Navigate to the bin directory. Type ./Labyrinth 3LetterPlayerName. The program should start and you're off to the races!

Controls:

To pause: f or F or select menu option through right click

To quit: ESC or q or Q or select from menu

To reset the camera: p or P

Board Controls:	Arrow key up tilts board back
	Arrow key down tilts board forward
	Arrow key left tilts board left
	Array key right tilts board right

Camera controls:	J/L to rotate left/right
	I/K to change pitch
	U/O to zoom in/out

Light controls:	Light1	t/y	moves light 1 along z-axis
		g/h	moves light 1 along x-axis
		b/n	moves light 1 along y-axis
	Light 2	T/Y	moves light 1 along z-axis
		G/H	moves light 1 along x-axis
		B/N	moves light 1 along y-axis

	1 turn ambient light off
	2 turn ambient light on
	3 turn distant light on light 1
	4 turn point light on light 1
	5 turn spot light on light 1
	(Press shift and any combination of these keys to apply to light source 2)
Menu Controls:	Quit – quit the game
	Resume Game – Resume from a paused state
	Pause Game – Pause the game play
	Add Ball – Adds an extra ball to the labyrinth board
	Swap Levels – Swaps between the two available levels

How to win:

Successfully navigate your ball from the start of the maze to the end of the maze without loosing all 3 lives. If all lives are lost, you are notified and the game is paused. Press F to un-pause and restart the game. Similarly, if you win, your total score and time are displayed, and the game is paused. Use the menu to un-pause and continue to next level or ress F to un-pause and restart.

You have a max time limit of 5 minutes, so don't delay!

In Game Play:

When you first load the program, the ball drops onto the labyrinth board and you can then begin navigating your ball by tilting the board with the arrow keys.

If you fall through one of the holes before reaching the finish mark the ball is reset back to the start. If you successfully reach the finish line, the game is paused and you are alerted of your win.

The menu can be used to change game play. You can swap which level you are playing by right clicking, and selecting “Swap Level.” To add a second ball, right click and select “Add Ball.” At the moment, there is no way to remove the second ball once it is added, so good luck! When you add ball or swap levels, at this time your score/time does not restart. The clock is always running, so be sure to act fast!

We implement a subtracting scoring scheme, the score that you start with is the highest possible score, and as time goes by your score goes down. The more lives you have during each level, the more points you will gain.

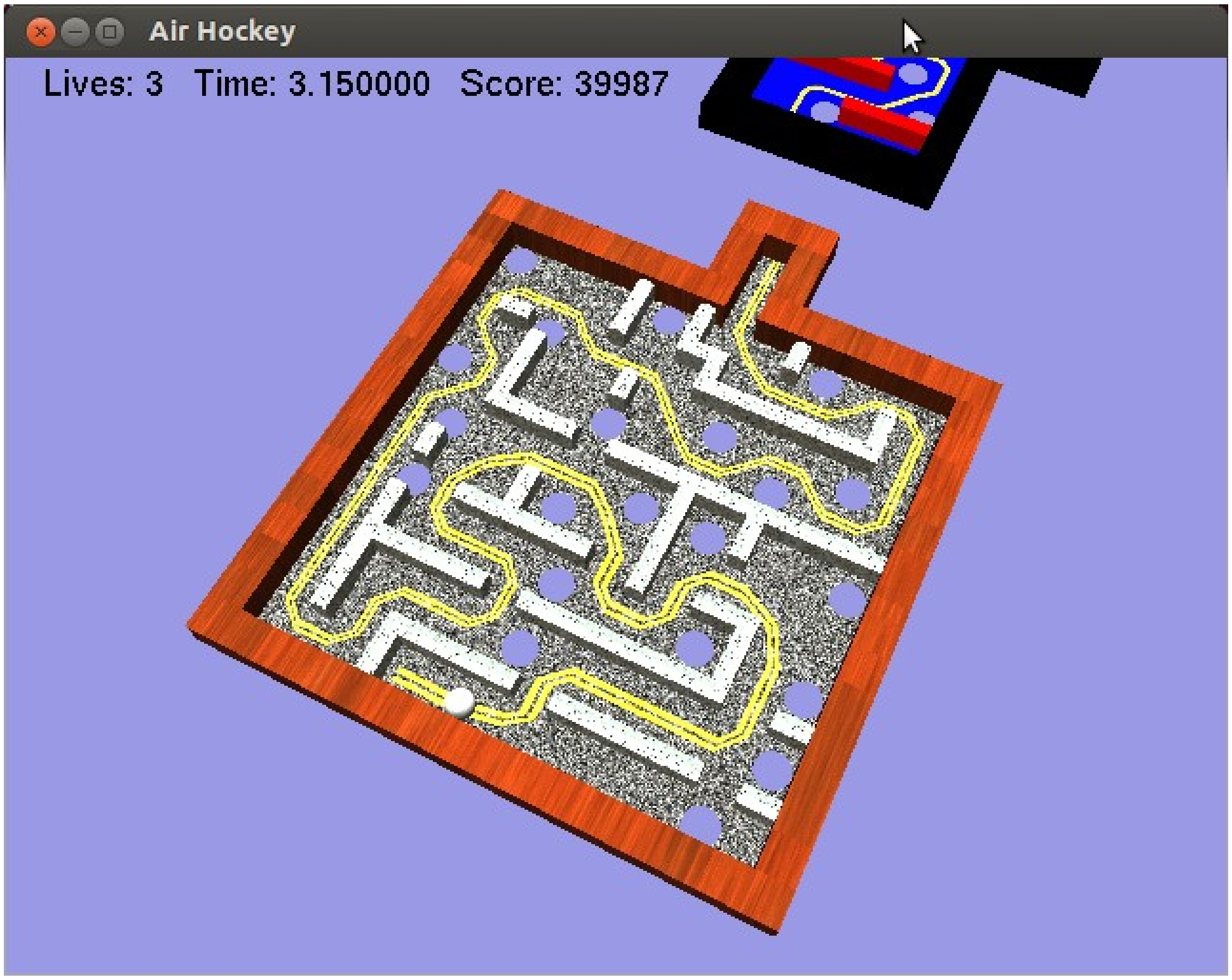


Illustration 1: Game at startup, with distant light enabled and playing on big board

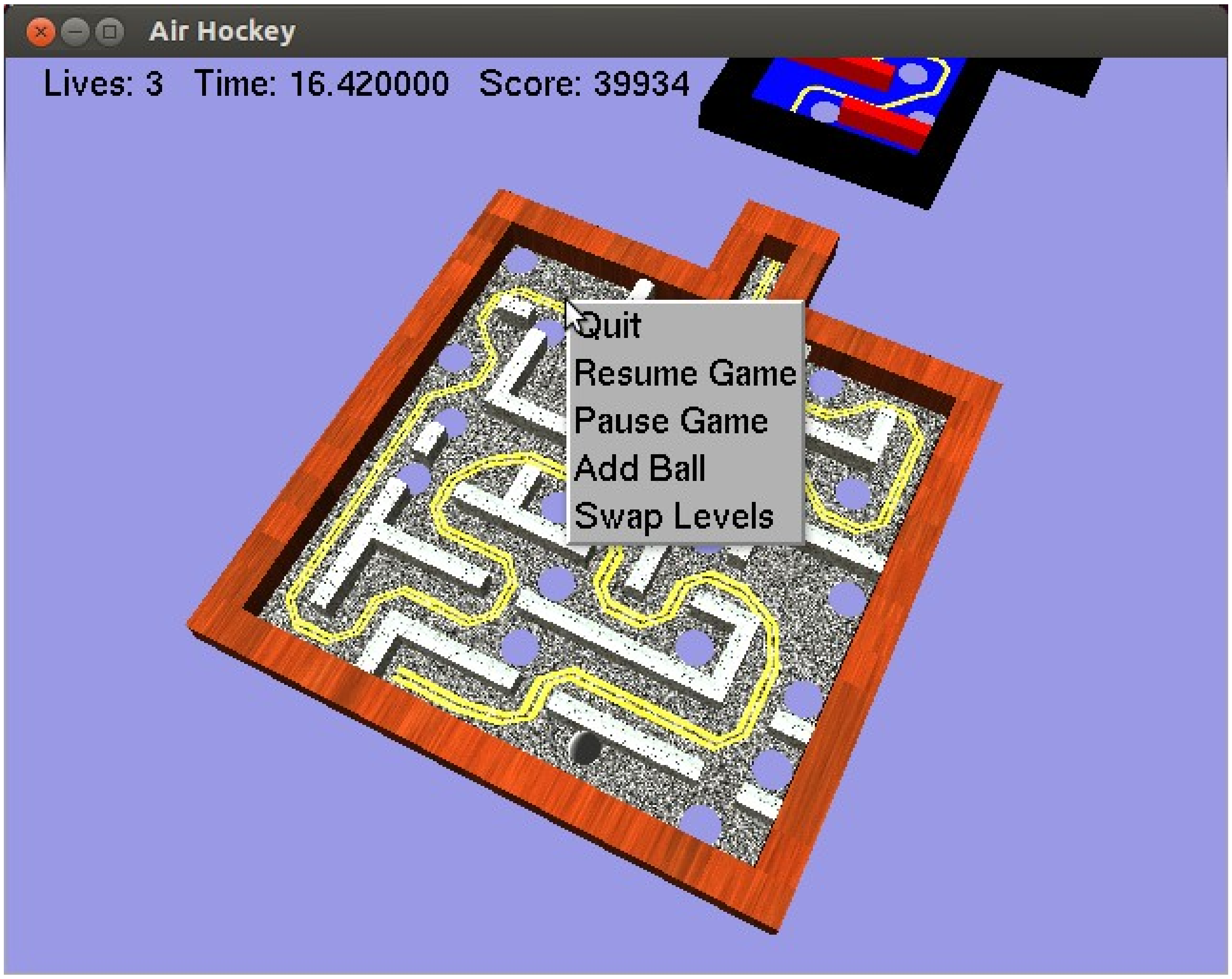


Illustration 2: The menu options, you can add ball, swap levels, pause, resume, and quite game.

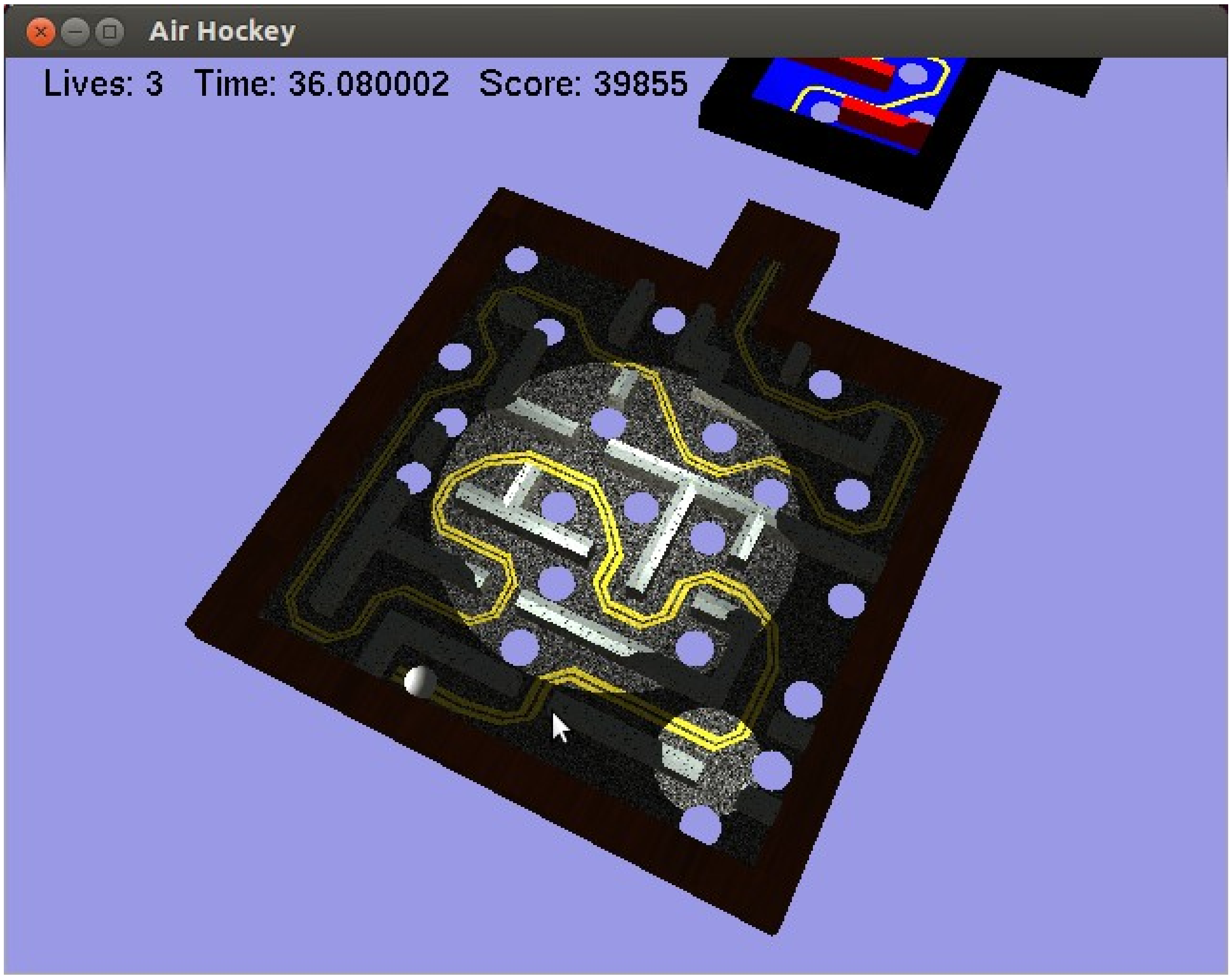


Illustration 3: Spot lights enabled on the big board, with both lights visible

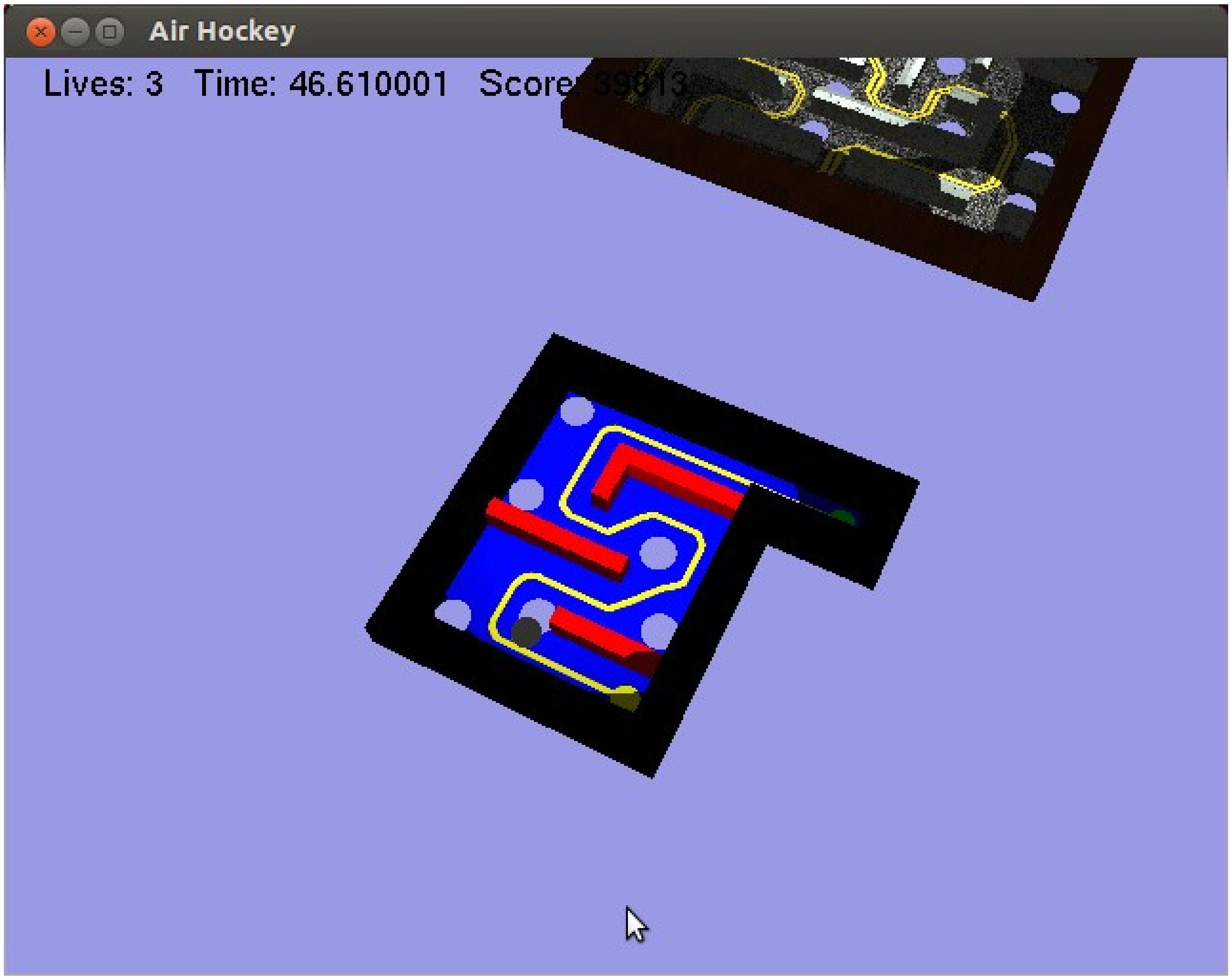
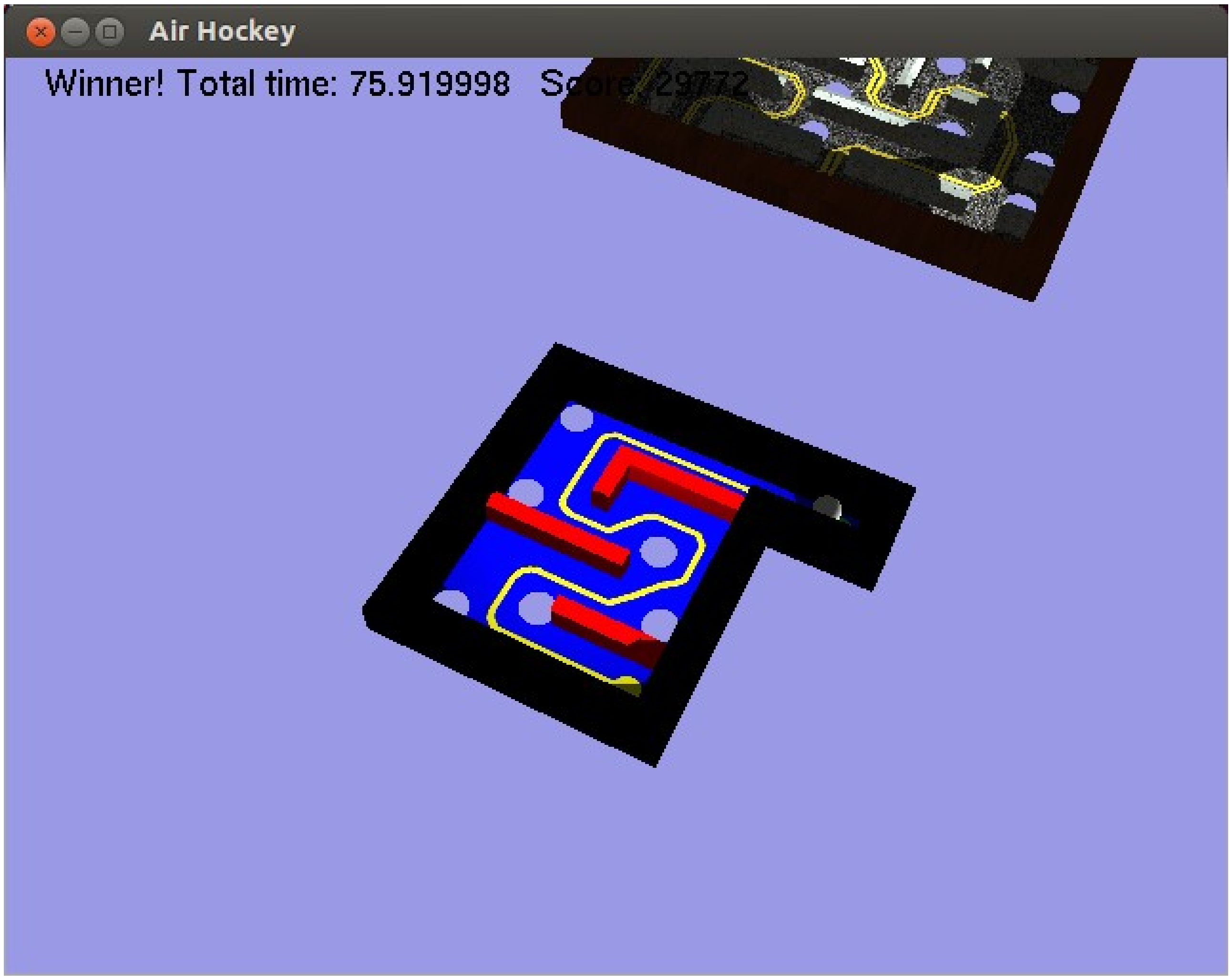
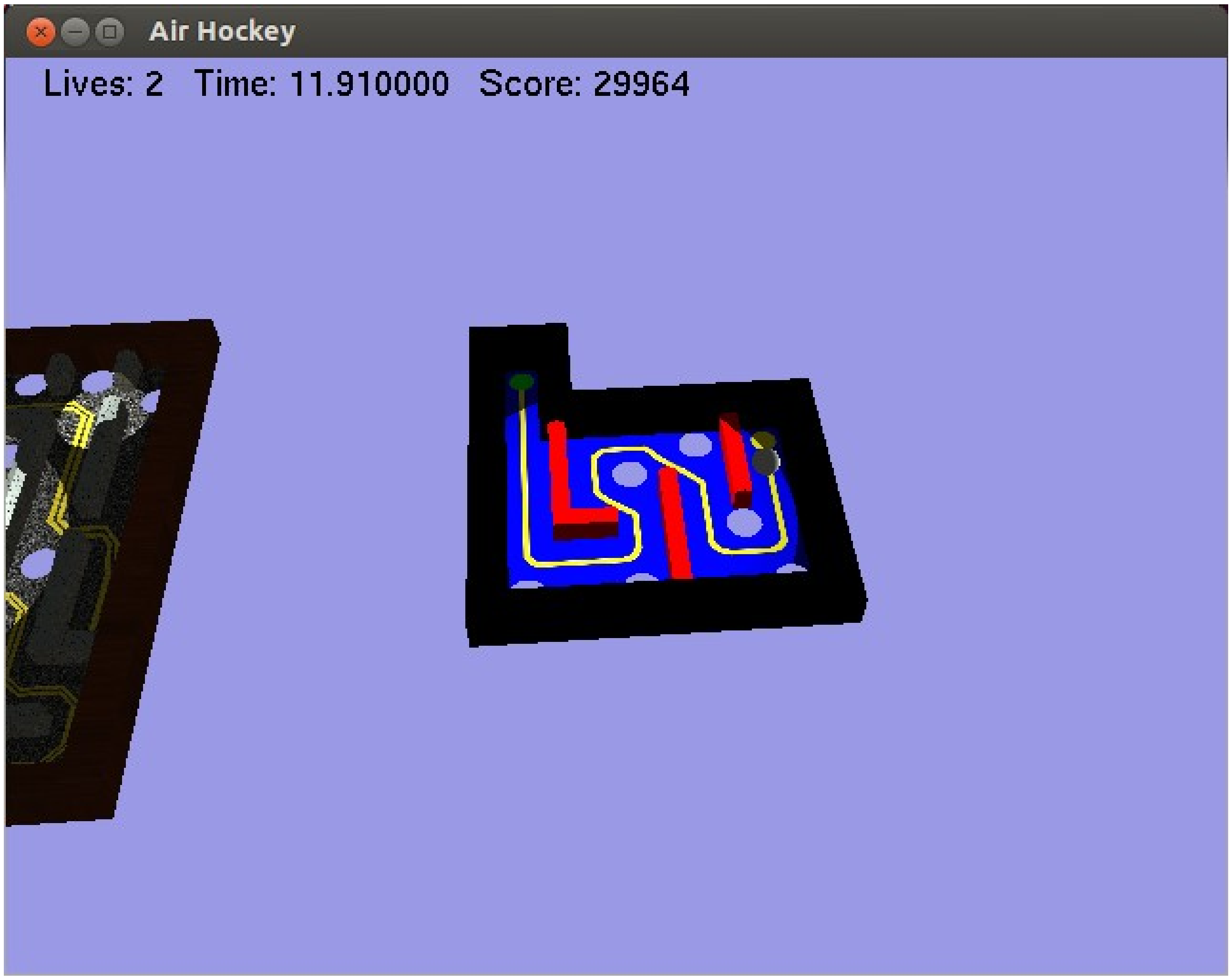


Illustration 4: The screen after swapping levels



Winner! Total time: 75.919998   Score: 29772

Illustration 5: Winning conditions with total time and score displayed



Lives: 2 Time: 11.910000 Score: 29964

Illustration 6: You can rotate and move the camer with the j/l, i/k, and u/o keys

Tech Manual:

Integrating Assimp (our model loader), Bullet (our physics engine), and opengl (for the graphics) has been a fun and interesting project. Some aspects of the implementation have been harder than anticipated, while others went much smoother than originally anticipated.

For labyrinth, we use a collision mesh shape for the labyrinth boards and a built in primitive for the ball (or balls if you have added the second ball to the screen). This ended up being one of the many large headaches with the game. Originally, our model ball was not centered at zero, leading us to believe our collision mesh shape was not properly centered. After much struggling we figures out that the ball was the culprit and our collision mesh had be correct the entire time.

We also discovered that you can get a 4x4 matrix out of the MotionState object in the physics world and apply this to the drawn models. This made translating the models much easier (at first we were attempting to manually rotate the drawn models to match the collision object, but this quickly proved to be far too difficult).

We were able to implement multiple lights (two to be exact), but we did run into issues with getting the lighting to be dynamic. Despite rolling around on board, the light of on the ball will not change, this is likely due to light and some property of the ball being in different spaces. After extensive testing we think it maybe the normals of each model that is responsible for this bug. However, we were unable to fix this bug at this time. The both light are movable, with the controls discussed at the start of this document.

We still have troubles running the code on different machines, in the ECC linux lab and on a Toshiba Ultra Book, when we display text it causes the program to lag terribly. We have determined that this is because of the call to glutBitmapCharacter(). The program runs smooth on all tested machines when we omit this call. Given more time we would like to figure out this interesting bug.

Also, in the event that the user is able to “flick” the ball off the board it is possible to get a false win by getting the ball far enough along the y-axis before it drops below the threshold check which tests if the ball has fallen off the board.

For future additions we would like to get the swap levels functionality working a little better, having the time and lives reset when you swap between. Given more time, we would make it so you start out playing on the small level, and are smoothly pan to the next level after a win. If you've gotten this far in the tech manual, we'll let you know about some secret controls. You can use a/d and w/s to manually “drive” the ball around the board by applying a linear velocity to it. This makes it easy to skip over holes, but be careful...it's easy to fly off the board by going too fast.

Overall Labyrinth was a fun, interesting, and very difficult project.

Source Code:

Source code can be found here: https://bitbucket.org/ericbskis/cs480/overview.

Please email Eric at ericb.skis@gmail.com in order to gain access rights.