"So we beat on, boats against the current, borne back ceaselessly into the CUBE."
--F Scott Fitzgerald (with apologies to)

Enter the Cube

Team 12

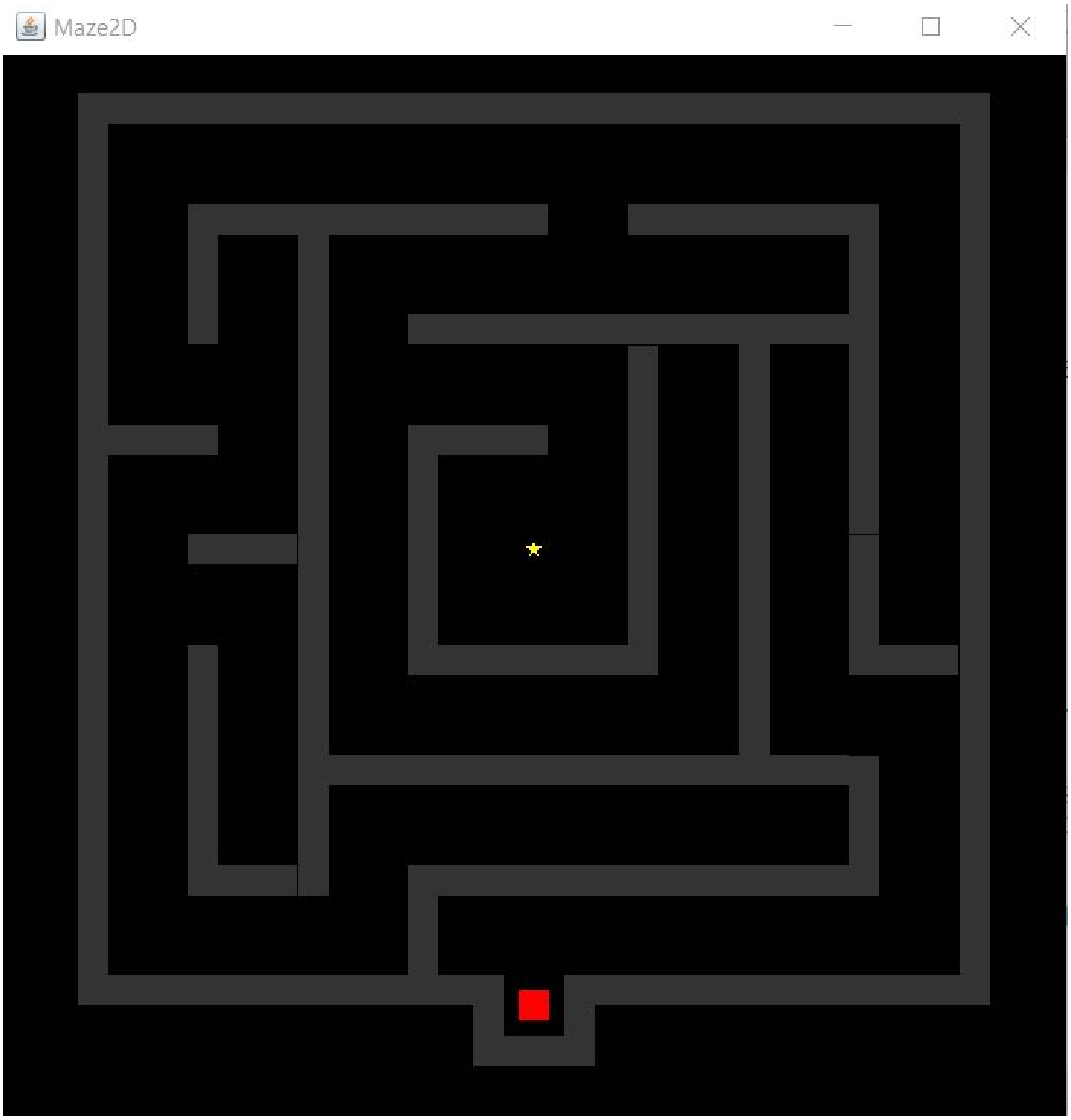
Wyatt S. Carpenter, Nathan Thomas, Samantha Crosley

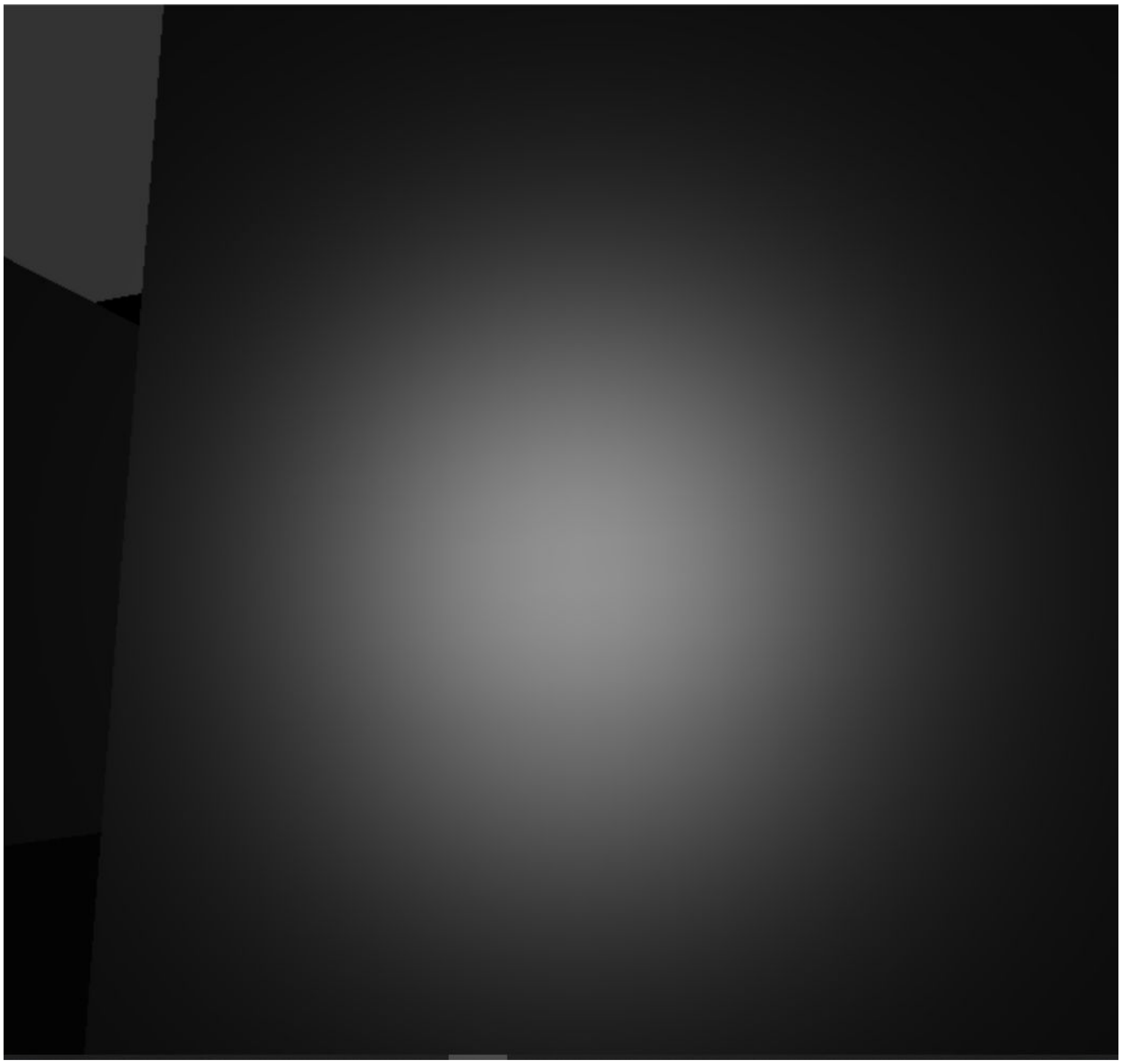
Introduction

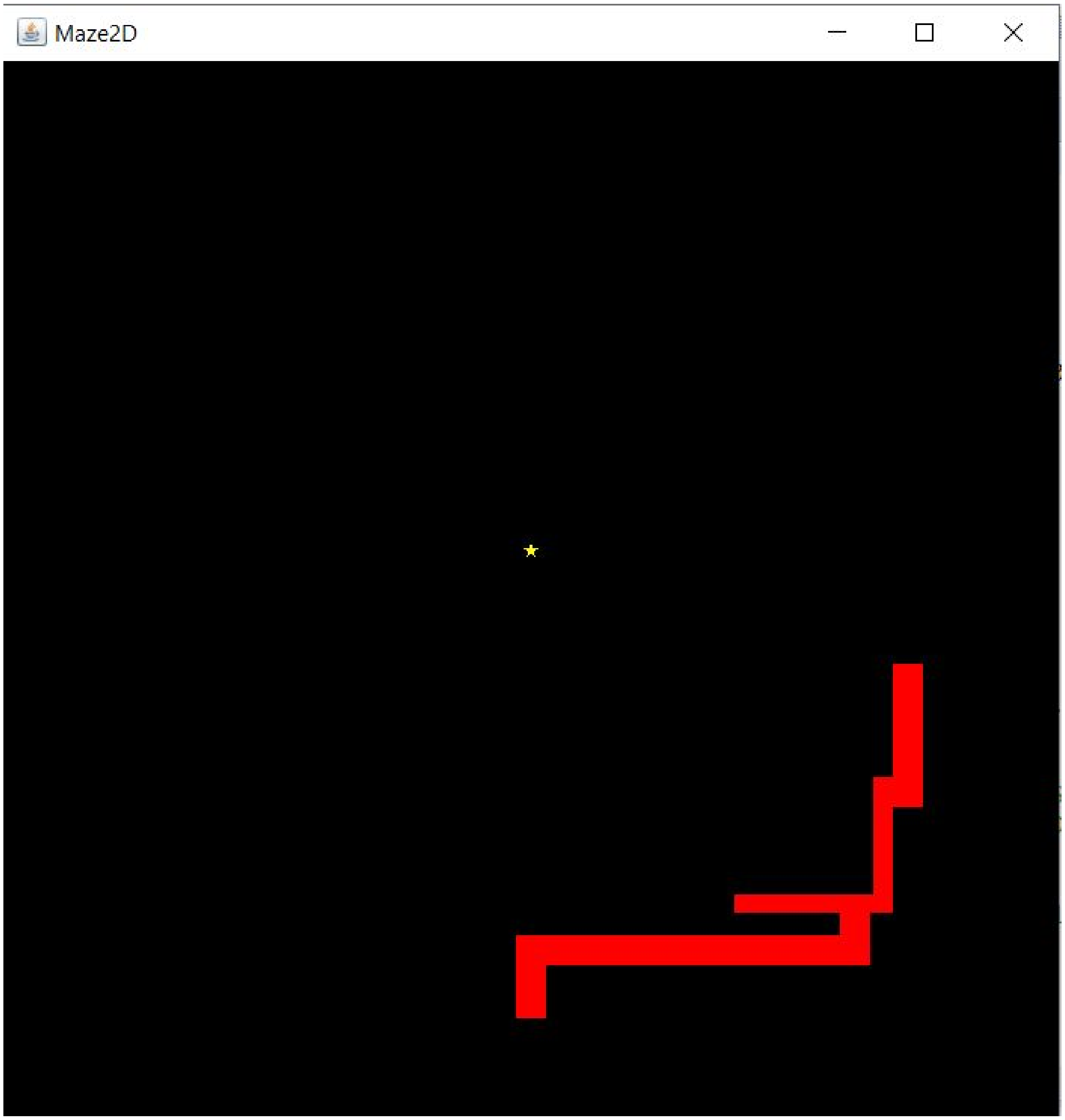
Our intended product is to have developed a video game, Enter the Cube, that features multiple 3D mazes. After the user reaches the floating object at the end of the current maze, the game takes the player to the next maze. The next maze in the game has a different 3D shape and scenery. The first maze in our game is set on a plane while the second maze is set on a cube. Each maze will have its own lighting and gravities. The lighting in the first maze will come from the players point of view, simulating a flashlight. Interaction with the walls, via light or touch, will cause different effects. In the second maze, the lighting will come from the center of the cube. We have succeeded in creating both mazes in 3D as well as adding the various lighting and gravities.

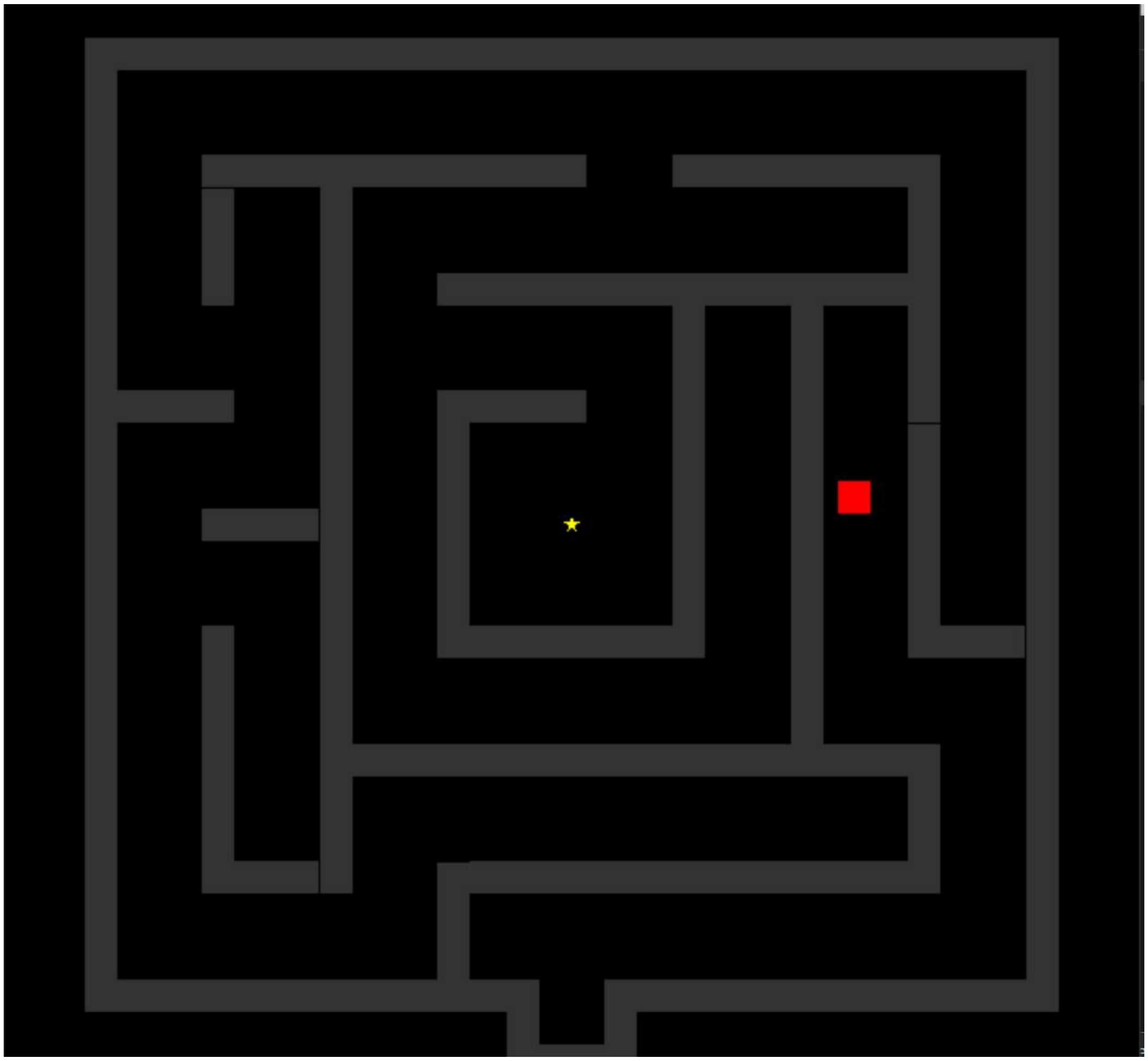
Example

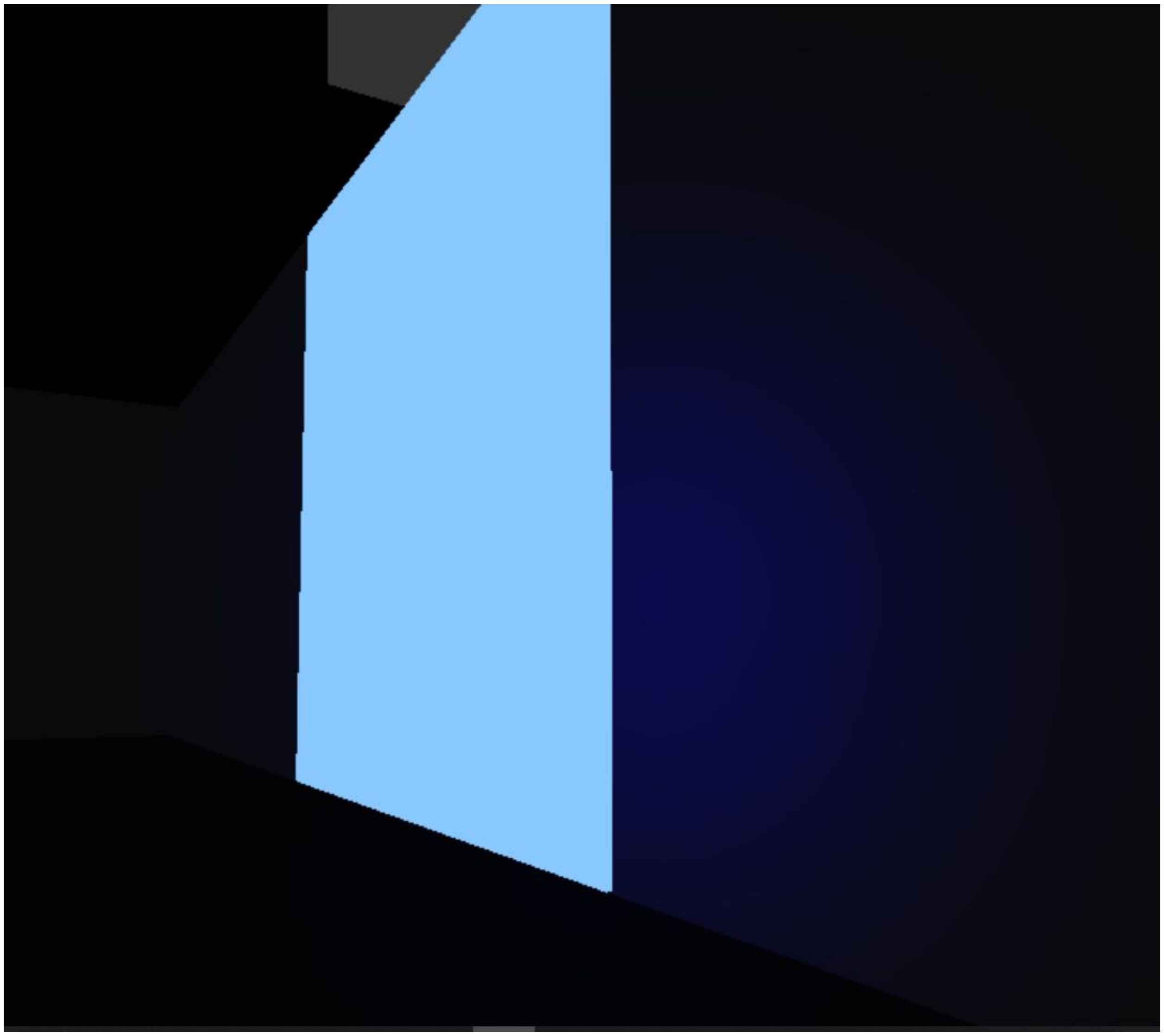
Upon loading the game, the player will see a 2D plane maze, as in the first image below. Pressing tab will toggle to the first-person view of the maze, as in the second image below, which is navigated with mouse and keyboard. Pressing C in the 2D plane maze will make the walls invisible and highlight the players trail, as shown in the 3rd image below.

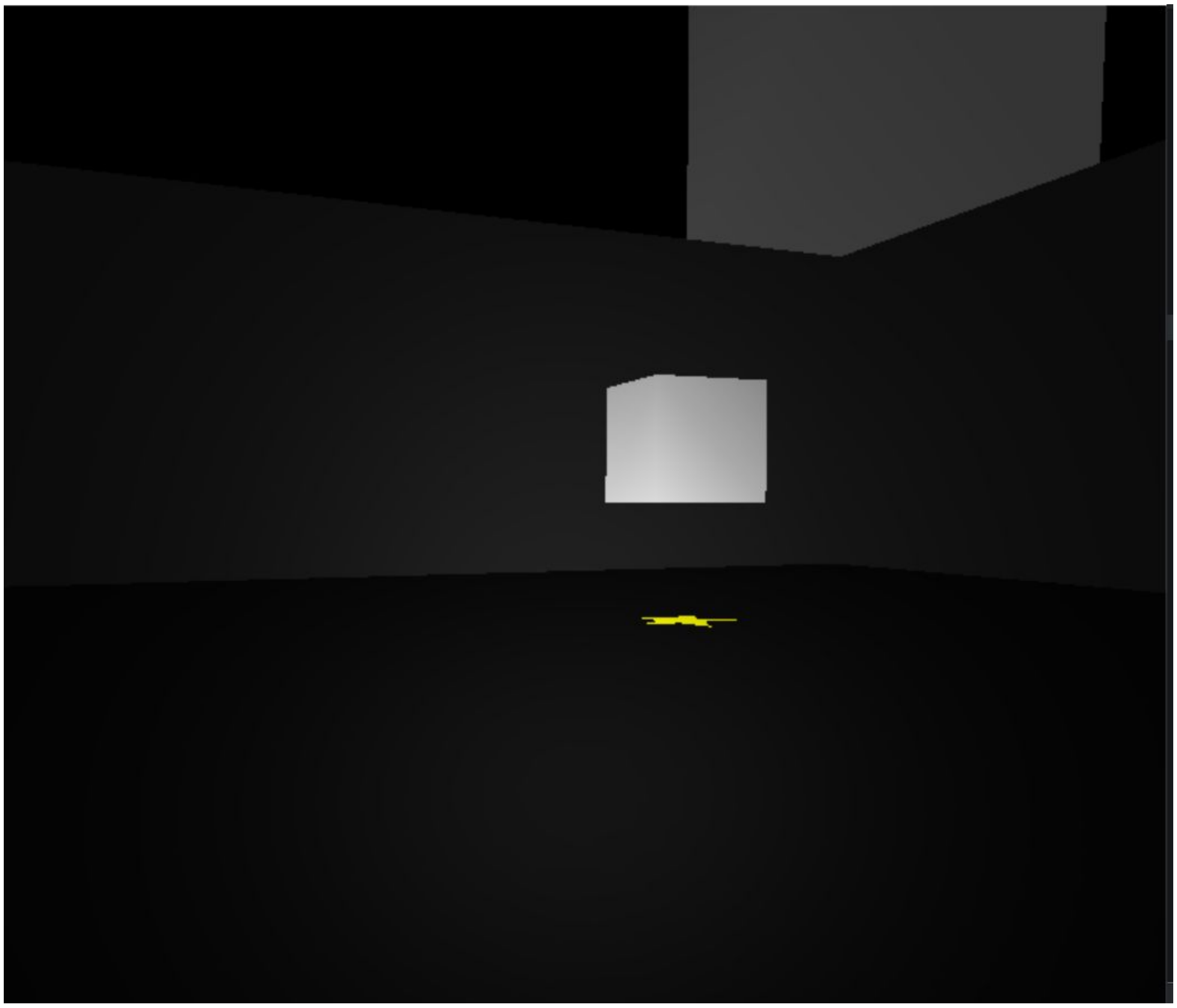

Maze2D




Maze2D







Process

Our approach to developing our game was to implement features based on their importance to achieve a working game. We decided that it was more important to implement collision detection and a basic maze before implementing added actions, lighting, and gravity. After the layout of the plane maze and collision detection was implemented, we then started implementing increase in speed, gravity, and lighting. After the first maze was completed, the second maze started being implemented.

Our original timeline of activities is outlined in the table below. The green highlighted boxes are goals that we achieved on time. The orange highlighted boxes are goals that we achieved late. Yellow highlighted boxes are partially completed. Unhighlighted boxes were not completed.

	Wyatt	Nathan	Samantha	Team Goal

Feb 21st - Feb 27th	Navigable 2D space	Navigable 2D space	Navigable 2D space	2D Prototype
Feb 28th - March 12th	Implement plane maze	Implement plane maze	Implement plane maze	Project Show and Tell 2D Plane Prototype Progress Report
March 13th - April 2nd	Implement 3D plane maze / Begin cube maze	Implement 3D plane maze / Begin cube maze	Research and add lighting and scenery	Have working 3D plane maze
April 3rd - April 16th	Begin individual research & development	Begin individual research & development	Implement cube maze Lighting and scenery	Have working game Prototype
April 17th - April 30th	Final fixes	Final fixes	Final fixes	Final Report Due 04/23 5053 Mini Project Due May 1st

Because of the switch to online classes and the pandemic, it has been a little more challenging to find time to communicate about and allocate time for the project. It also meant that we needed to assess what was a priority for our final prototype. Because of the current circumstances, we decided that scenery wasn't a priority for our finished product.

Detailed Results

Enter the Cube is a first-person maze game that allows the user to run and jump, look, and *feel*. It features wasd/arrow key movement, real Newtonian gravity over fixed heavenly bodies, mouselook based on that gravity, cubes, and a hudless graphical user interface (the game).

The game starts in an overhead view of a gray maze, but a tap of the 'Tab' key swaps the view to first-person (this first maze can be completed in either perspective). There is also an easter egg feature (pressing 'C') that allows the user to make the walls of the maze invisible to add challenge for the user. In this mode, the user is able to see their trail, but cannot see the walls of the maze.

For the 1st maze, taking place on a plane, the light comes from the players point of view to simulate light from a flashlight. Throughout the maze, there are wall(s) that will glow blue when the player shines the light on it. If touched by the player, the wall(s) grow and spin wildly, eventually disappearing. At the end of every maze, there is a floating object that, when touched, will transport the player to the next maze. In the first maze, this is a cube. It floats above a symbol of a star.

For the 2nd maze, taking place in a white cube, the lighting comes from the center of the cube. This maze features real gravity, allowing the player to walk every interior surface of the cube. The maze covers all six faces of the cube. The user encounters a rotating plane at the end of this maze, transporting them back to level 1. This is a metaphor for college.

The player can jump against the direction of gravity by pressing the spacebar. Holding the spacebar jumps higher. The player can move perpendicular to the direction of gravity using the ‘WASD’ or arrow keys, as you would expect. Tab and ‘C’ keys have already been described. Shift and Control sprint and crouch respectively, which increase and decrease your speed respectively.

The player can move the mouse (“mouselook”) over different parts of the play screen to rotate and see other angles on the world.

The code is all in one package: enter.the.cube. There is an application class that serves as a wrapper to run all the other classes, a keyboard handler, a mouse handler, a model class that tracks all the variables and locations of things, and a view class which makes JOGL calls to draw the graphics. There is also a Point3D class in which Wyatt once again implemented vector math, as all Java projects seem to demand of him somehow. The program is written in Java, and uses OpenGL with JOGL. We all used Eclipse to complete this project, and the project uses gradle to build itself somehow.

Playing our game will require a computer of some meager graphical ability from the last decade, an ability to run java and JOGL (Java OpenGL) on this computer, a keyboard, a mouse/trackpad/nub/complete-lack-of-visual-curiosity, and an iron will. These components must all be connected properly.

Future Directions

Fortunately, the team was able to achieve all of our major goals for Enter The Cube, except for the goal of making the experience of playing “exceptionally pleasant”.

Another, though minor, goal left by the wayside was the FMV portion. Initially, we had intended to record a full motion video segment (of, it must be admitted, very little significance) and play it back on some surface in Enter The Cube. However, due to cancellation and delays on the part of various sources we approached to be subjects of fmvs, nontriviality of implementation, as well eventual lack of interest and time on the part of the team, we decided to scrap this feature. We considered instead playing an audio segment to provide a similar point of interest, but for similar reasons eventually decided to scrap this feature as well.

Some major features that we weren't able to implement for our maze include scenery. Our original plan for the maze was to add unique scenery to each level of the game, but because of time and communication restraints, we weren't able to add more scenery to the different levels. As evidence of this communication restraint, only one member of our team believed scenery was a major planned feature of the project, whereas the others had believed all along that "scenery" meant "more cubes".

Originally, early in the planning stage, we had intended to implement many levels, of many shapes and sizes. We were advised to pare down our ambitions to two, but if we had time we would add more levels, such as: a second plane, a sphere, and the OpenGL teapot.

Besides the additional levels and FMV described above, expanded development of this project would mainly focus on polish. The project fell short of its goal of extreme pleasantness, and expanded development could rectify this, and other aspects of the project, by focusing on the below areas:

Input Capturing: Our ability to capture input was extremely suboptimal. Instead of capturing cursor like a real game we just track the mouse in our window. Instead of keeping track of keyboard input like a real game we just take the keyevents as they come. These contrivances can result in awkward maneuvering, such as the fact that you have to wait a second to continue walking in a direction after the first step if you are holding the button, because the OS must decide to begin repeating your keyevent.

Joyful Physicality: Unfortunately, the mind-blisteringly realistic Newtonian gravity system may be a little too mind-blisteringly realistic. For example, generally people want to be able to walk into slopes. However, you will note this is impossible, and in real life is only accomplished because you move *above* the slope on purpose, but lifting your legs. Enter The Cube compromises slightly on this, allowS players to walk up gentle slopes, but the realism would probably be sacrificed for pleasantness if more development was to be had.

Bigger Levels: Our mazes were small and thus insufficiently amazing.

Looking Good: Our mazes are visually dull and their interior parts are hard to distinguish. To combat this, we might try adding textures, shapes, programmatic depth perception, and shading.

Resolution: For some reason, we never bothered to adjust away from an internal 700x700 resolution. While the game can be played in 1920x1080 (or any other resolution) by adjusting the window, it would probably look better (less stretched) in this (our canonical target resolution) if we used an internal resolution of 1920x1080.

Code Quality: During this project, we exercised a design philosophy known as "continuous deprecation"[1] in which working code was purposefully never changed. This means when a new function was written to deal with something, any old code was not refactored to use this function. This allowed the project to proceed at a rapid pace, secure in the knowledge that progress was never lost. This may seem intuitively like a bad strategy, so consider the anecdote of the one time Wyatt got annoyed with his own strategy on this project: he removed Nathan's carefully crafted maze bounds checks to refactor them all into a newly revised moveTo function. While Wyatt had his own valid reasons for doing this, it means that in the project the red square can now walk half into walls and the player can occasionally see behind surfaces. What a set-back! That's why the philosophy of continuous deprecation was adopted. However, there is a significant amount of code that could be removed and/or enhardied by refactoring, so continued development would probably include some of that.

Conclusion

Our intended product of a 3D maze video game with multiple levels on different shapes was, overall, executed successfully. We were able to create the mazes as well as lighting and gravities. In addition to this, we were also able to implement jumping, sprinting and crouching. Unfortunately, we weren't able to add textures or pleasing scenery, because it was deemed lower priority than a working maze. Aside from our main goals, we were able to add some things that weren't intended in our original plan for the game. We added functions to toggle between 2D and 3D perspective for the plane maze as well as to toggle between the walls visibility. However, we fell short on being able to add scenery and the planned video segment.

Team Summary

Wyatt S. Carpenter and Nathan Thomas' roles were to work on the implementation of the mazes while Samantha Crosley's role was to write reports as well as work on the scenery for the game. Wyatt spent a lot of time working on the gravity, the camera, the controls, the cubes, and level 2. Nathan spent a lot of time working on the lighting, the walls, the non-cube shapes, and level 1. Samantha wrote a large portion of the reports throughout the semester, but was unable to work on scenery due to everyone agreeing a working maze was higher priority. She also was able to design the maze for the cube.

Masters of Computer Science

[1] Unrelatedly, we also practiced continuous integration.

Nathan and Wyatt especially implemented a magic wall segment that glows when a blue flashlight is shone on it and disappears when run into. When the blue light is not shown on the wall it appears as any other wall. We placed this wall in between two other walls in the maze to make it appear as there is no entrance until the player switches to the blue light.

Nathan implemented the lighting features. For the plane maze he created a flashlight effect. The user can click ‘f’ to cycle between white flashlight, blue flashlight, and no flashlight. The flashlight effect was created by setting the position of the light to be at the player's location and setting the light direction to be the direction the player is facing. To make the flashlight effect he limmited the spread of the light and the distance the light travels. The spread was limited setting the spotlight cutoff and spotlight exponent. The distance was limited by setting the lights attenuation. In order for the light to appear he set the lights ambient, diffuse, and specular properties. When the light is white specular is set to white, and ambient and diffuse are set to gray. When the light is blue specular is set to blue, and ambient and diffuse are set to dark blue. In order for the walls to appear in their intended colors and not just the color of the light the material of all objects in the world were set to their real world color. He also created the glowing lighting effect for the special walls. This was done by setting the emission of the special wall to a light blue color when the flashlight is blue. This makes it appear as though the special wall is glowing when the blue light is on. He also created the lighting effect for the cube maze. This was done by placing the light in the center of the cube. The light spreads in all directions of the cube. The attenuation was set to make the light look appealing.

Wyatt implemented the wall disappearing animation. While implementing, he realised that the drawWall function he wanted to use was extremely bound to the orthogonal axes of the world. He therefore had to learn the gl matrix system, and transformations and rotations, to simply accomplish this animation. The animation causes the magic wall to rotate around its center and expand. Since the player must run into the wall to trigger the animation, the player is usually inside the wall when this animation happens, which produces an unusual and disorienting effect that Wyatt thought was aesthetically pleasing. While the animation is running, the player can observe regular walls that are encompassed by the magic wall and thus execute the maze like normal. After 1000 frames, the animation ends and the wall disappears. Wyatt thought this turned out well.