Maze Database
for UTM CSCI 352

Grace Looney
Matthew Pugh

Abstract

The goal of our project is to create a maze game using WPF applications. We are planning for our game to be able to randomly cycle through at least five pregenerated mazes. Our target audience is gamers who enjoy solving puzzles and are looking for a short, fun diversion.

1. Introduction

For *Maze*, we accomplished an interactive WPF application that involves computer generated mazes. We are using a Binary Algorithm to create the mazes due to its ability to create infinite sized mazes very quickly. The maze will be seen in an isometric view where the viewer will only be able to see a portion and the screen will move with the player.

Maze is intended for people who enjoy puzzle games. It offers a simple distraction to the constant boredom most humans suffer from. We hope our audience will achieve a sense of fulfillment out of our game because they solved a maze or multiple mazes.

1.1. Background

We want to our maze to be 'perfect' and 'orthogonal' [2]. A 'perfect' maze means that the maze lacks loops and closed circuits. So the maze does not have any unreachable areas. In addition, 'perfect' implies that there is exactly one path. There are multiple ways to do this. For example, one of the easiest ways to achieve a perfect orthogonal algorithm is to a Binary Tree algorithm. Although the Binary Tree algorithm is simple and fast, there is one con it is very biased due how it creates the maze. Each cell carves the next cell by randomly choosing a cell that is either two directions which causes the product to have passages that lead diagonally from upper left to lower right. We overlooked this downside because of the ease of implementation.

We wanted to do this project because of the implementation of an algorithm that we learned in CSCI 325. In addition, this application allows us to gain an understanding of how game applications work.

1.2. Challenges

There were many challenges to this project. With the CONVID-19 outbreak, we lost the ease of the communication which was the first step of our mighty downfall. There were multiple challenges: implementing a way to view the maze, player movement, and the settings menu. The biggest challenge implementing a way to view the maze, because the algorithm only returns whether the cell is linked. Although we never accomplished the player movement. The Settings menu's challenges were diverse because we had to understand how to pass on the settings values that the user selected to other pages.

2. Scope

Our project is done when we have a 'perfect' 'orthogonal' maze application with a top-down view. In addition, the user will be able to view a menu with buttons that'll allow them to make binary tree maze and format it to be viewed top-down.

2.1. Stretch Goals:

We completed 1 out 4 of our stretch goals. The timer is half-way implemented. It shows a timer but when the user pauses the maze it does not stop the timer. We did not complete the *scored* stretch goal, because of the timer issue and the inability to gather points. We minimally completed the *music* stretch goal. The music does not continuously play, but it does restart every time the user opens a new window. The *secret buttons* implementation was not attempted.

- Timed: The user can see how long it took them to complete that maze.
- Scored: The user can earn points by collecting items across the maze.
- Music: The user can have music while completing the maze.
- Secret Buttons: The user can access special items or areas when standing on secret button

Use Case ID	Use Case Name	Primary Actor	Complexity	Priority
1	Start New Game	Gamer	Med	1
2	Customize Maze	Gamer	Med	2
3	Save Game	Gamer	Med	3
4	Load Game	Gamer	Med	3
5	Calculate High Scores	Computer	Med	3
6	Play Game	Gamer	Hard	1
7	View Credits	Gamer	Easy	4

TABLE 1. USE CASE TABLE

2.2. Requirements

For our requirements, we met a portion of them. We were able to complete all of the nonfunctional requirements. The application uses the binary tree algorithm to create a maze within one second on worst case scenario 2.2.2. For the *scalability* 2.2.2, there is a global maze variable that a power user can adjust when they want a bigger or smaller maze. The functional requirements that we met are 1,2,3, and4. The functional requirements that we did not meet: 5, 6, and 7.

2.2.1. Functional.

1) User needs to be able to view the GUI
2) User needs to be able to interact with the GUI
3) User needs to view the maze
4) User needs to be able to interact with the maze
5) Application needs to be able to save 3 games
6) Application needs to be able to load any of the three saved games
7) Application will record high scores

2.2.2. Non-Functional.

- Performance- Application will create mazes in a short amount of time
- Scalability - Application will create mazes of different sizes

2.3. Use Cases

We have seven use cases. 2.3 is medium complexity because all this case required was transferring of the data values: size, settings choices. In addition, it needed to show a game screen after the *New Game* button was clicked. 2.3 is medium complexity and 2_{nd} priority because it only requires an small understanding of how GUI works and how to transfer values from one page to the next. In addition, Customize does not impede any other use case, so this is why it has a low priority. 2.3 is medium complexity and low priority because it's implementation only effects 2.3 because Load calls upon the data that 2.3 saved. 2.3 is low priority and medium difficulty because it does not affect any other use case. 2.3 is high priority and very complex. Playing the game is the sole purpose of this project so its complexity and priority demonstrate this. 2.3 is listed as an easy task and low priority because it does not impede any other use case and its just an information window.

Use Case Number: 12.3
Use Case Name: Start New Game
Description: The gamer on our WPF application wants to play a new game. They will click the "New Game" Button, and this will start the off the Binary Tree algorithm to create a new maze. In addition, the maze will display after algorithm is completed.
Pre-conditions:
- Maze is downloaded
- Maze has been opened

Post-conditions:
- New game will be started

Invariants:
- New game function is working properly
- Game runs as it is meant to

Process:
1) Select New Game from the main menu
2) Play game

Use Case Number: 2
Use Case Name: Customize Maze
Description: The gamer on our WPF application wants to edit their maze appearance or the sounds of the maze. The gamer clicks the "Settings" button. This will take the gamer to the settings menu, so they can customize the maze appearance.

Pre-conditions:
* Maze is downloaded
* Maze has been opened

Post-conditions:
* Current maze theme will be updated

Invariants:
* Theme function is working properly
* Music function is working properly
* Customize function is working properly

Process:
1) Select settings from home screen
2) If desired theme already exists, then select it from the Theme menu
3) If desired theme does not exist, then move to the Customize menu
4) Select desired color for maze walls
5) Select desired color for maze floor
6) Select desired music setting from the Music menu
7) Save maze settings

Use Case Number: 3

Use Case Name: Save Game

Description: The gamer on our WPF application wants to save their unfinished maze. The gamer selects in the ”save” button from the game view. They will be able to select one of the save spaces to save their maze to that memory spot.

Pre-conditions:
* Maze is downloaded
* Maze has been opened

Post-conditions:
* Current game is saved in desired save slot

Invariants:
* Pause function is working properly
* Save function is working properly

Process
1) Click pause button
2) Click save button
3) Select a save slot
4) If save slot is currently being used, then decide whether or not to overwrite the previous save
5) Return to pause menu
6) End current game

2.4. Interface Mockups

Figure 1. This is the Menu, There are

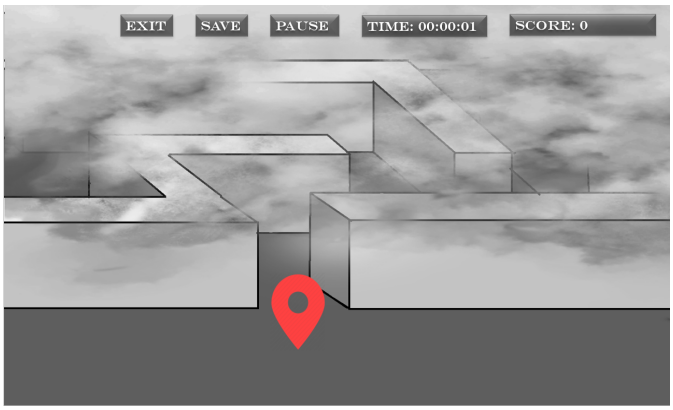

Figure 2. This is the screen where the ”New Button” transports the user

Figure 3. This is the Load Screen that will appear after they select the button ”Load Game”

Figure 4. This is the screen that will be shown after the Maze game has finished

Figure 5. This is the Customize Menu

Figure 6. This is the High Score Screen

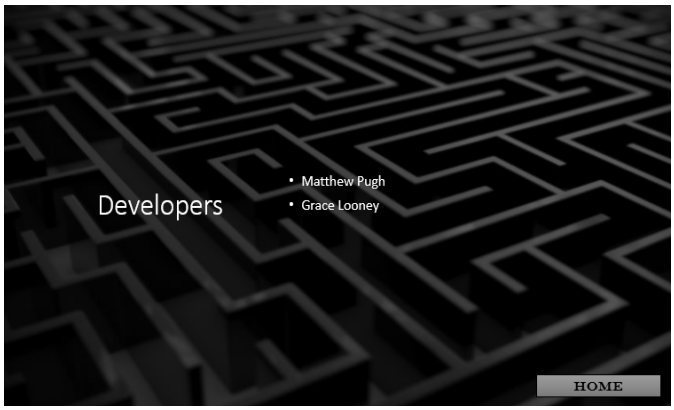

Figure 7. This is the Credits

3. Project Timeline

Maze Game Timeline: (refer to Table 2 for a more detailed schedule)

1) Requirements: January 20 - February 27
2) Design: February 28 – March 31
3) Implementation: April 1 – April 20
4) Verification: April 21 – April 26
5) Maintenance: April 26 - onward

Explanation:

- Requirements
  - Teams formed. Began planning what we wanted the game to do and look like. Decided what features we need/want to implement.
- Design
  - Planned out the programming structure of the maze game. Selected the Singleton and Observer design patterns to use in implementation of the game.
- Implementation
  - Full scale implementation of designs.
    1) April 1 – April 7: Begin implementation of the main menu as well as the mazes themselves. Focus on completing the new game and load game options. If new game and load game are completed, begin implementing the high score page and settings menu.
    2) April 8 – April 14: Continue implementation of the mazes and finish implementing the high score page and settings menu if they are not yet complete.
    3) April 15 – April 20: Finish implementation of the mazes. Begin working on stretch goals (timer, music, secret buttons, hidden objects) if enough work is completed on the mazes.
- Verification
  - Maze should be completed at this stage. Prepare for the final presentation on April 26.
- Maintenance
  - Review goals that were and were not accomplished. Discuss new features that may be implemented in the future.

Activity	Start	End	Notes	Priority
Requirements:				
Team Formed	1/9/2020	1/11/2020	A team of two exists and they tolerate each other	1
Project Idea	1/10/2020	1/15/2020	Both members agree upon an idea	1
Purpose	1/21/2020	1/15/2020	Both members agree upon the purpose of the project	1
Features	1/21/2020	1/15/2020	Both members agree upon an what features to include	1
Use Cases	2/11/2020	1/15/2020	The use cases are listed in the latex file	1
Non-functional and functional	2/11/2020	1/15/2020	The requirements are listed in the latex file	2
Interface Mockup	2/11/2020	2/21/2020	The mock-up slides are added to the Latex file	2
Design:				
Timeline	2/26/2020	2/27/2020	The timeline is added to the latex file	2
UML Outline	2/28/2020	3/31/2020	The outline is added to the latex file	2
Researched Design Patterns	2/27/2020	3/31/2020	The team reasearches possible design patterns to implement	3
Researched Implementation Ideas	2/29/2020	3/31/2020	The team researches possible ways to implement project	3
Updated Latex File	3/31/2020	3/31/2020	The team updates the Latex file with the changes	2
Implementation:				
Main Menu	4/1/2020	4/8/2020	Main menu implementation should be complete	2
Maze	4/1/2020	4/14/2020	Maze implementation should be complete	1
High Score	4/8/2020	4/16/2020	High score implementation should be complete	2
Settings	4/1/2020	4/14/2020	settings menu implementation should be complete	2
Stretch Goals	4/15/2020	4/21/2020	Stretch goals should implementation should be attempted	3
Verification:				
Testing	4/15/2020	4/20/2020	Test project	1
Preparation for Presentation	4/20/2020	4/20/2020	Prepare	1

TABLE 2. TIMELINE TABLE

4. Project Structure

4.1. UML Outline

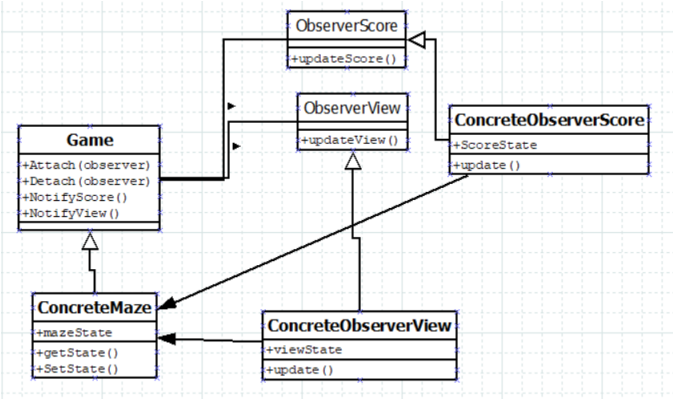

Figure 8. How we will update the GUI

We chose the observer pattern because it allows us to customize the maze immediately on change. We chose the Singleton

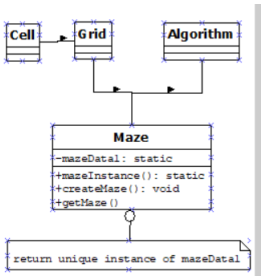

Figure 9. How we will handle the three instances of the maze

pattern to control the number of instances of the maze.

4.2. Design Patterns Used

1) Singleton
2) Observer

5. Results

The result of our project is sad. We barely accomplished half of the work. We were unable to meet the minimum requirements. In summary, we have a half implemented maze game This section will start out a little vague, but it should grow as your project evolves. With each deliverable you hand in, give me a final summary of where your project stands. By the end, this should be a reflective section discussing how many of your original goals you managed to attain/how many desired use cases you implemented/how many extra features you added.

5.1. Future Work

Future work, we will delete all copies of the game, and forget this catastrophe ever existed. In addition, this partnership will never happen again.

References

[1] H. Kopka and P. W. Daly, *A Guide to LaTeX*, 3rd ed. Harlow, England: Addison-Wesley, 1999.

[2] Walter Pullen. *Think Labyrinth*,(2012).