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 and databases. 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. Currently, we are still researching and planning out how to implement our project design.

1. Introduction

For Maze, we are trying to accomplish a interactive WPF application that involves databases and computer generated mazes. We are using Eller's Algorithm to create the mazes due to its ability ro create infinite sized mazes very quickly. The mass of mazes will be stored as a 2d array in a database. 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 fullfiillment 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 cricuits. 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, The best way to achieve this is to use to achieve an 'perfect' 'orthogonal' maze is to use Eller's Algorithm. Eller's Algorithm is incredibly fast and very memory efficient because it only requires storage proportional to only a single row. So in the future, we aim to get the application to make simple mazes on user-demand.

We decided to do this project because it allows us to use databases and algorithms. Algorithms hold a special place in our hearts. They offer a way to implement something amazing efficiently. On the otherhand, databases are something new that we have learned in CSCI 352.

1.2. Challenges

We will most likely get stuck in implementing a database, Eller's algorithm, and the isometric view. We are both neophytes concerning databases, and we need to find a way so that we limit the amount of space that an array takes up. Eller's Algorithm requires implementers to understand sets and inner-relations of sets to arrays. The isometric view will require us to understand 3D graphics which none of us have experience in.

2. Scope

Our project is done when we have a 'perfect' 'orthogonal' maze application with a view. It will be a functional application that calls on a database holding minimum of five mazes and be able to call on them at random. In addition, the user will be able to view a menu with buttons that'll allow it to randomly select a maze and format it to be viewed isometricly.

2.1. Stretch Goals:

- 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

As part of fleshing out the scope of your requirements, you'll also need to keep in mind both your functional and non-functional requirements. These should be listed, and explained in detail as necessary. Use this area to explain how you gathered these requirements.

2.2.1. Functional.

- User needs to be able to interact with the maze
- Application needs to be able to save 3 games
- Application needs to be able to load any of the three saved games
- Application will record high scores

2.2.2. Non-Functional.

- Perfomance- Application will create mazes in a short amount of time
- Scalability - Application will create mazes of differnent sizes

2.3. Use Cases

This subsection is arguably part of how you define your project scope (why it is in the Scope section...). In a traditional Waterfall approach, as part of your requirements gathering phase (what does the product actually *need* to do?), you will typically sit down with a user to develop use cases.

You should have a table listing all use cases discussed in the document, the ID is just the order it is listed in, the name should be indicative of what should happen, the primary actor is typically most important in an application where you may have different levels of users (think admin vs normal user), complexity is a best-guess on your part as to how hard it should be. A lower number in priority indicates that it needs to happen sooner rather than later. A sample table, or Use Case Index can be seen in Table 1.

Use Case Number: 1
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 Eller's algorithm to create a new maze. In addition, the maze will display after algorithm is completed.
Preconditons:
- Maze is downloaded
- Maze has been opened

Postconditons:
- New game will be started

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

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 appearence 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 appearence.
Preconditons:
- Maze is downloaded
- Maze has been opened

Postconditons:
- Current maze theme will be updated

Invariants:
- Theme function is working properly

- Music function is working properly
- Customize function is working properly

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.
Preconditons:
- Maze is downloaded
- Maze has been opened

Postconditons:
- Current game is saved in desired save slot

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

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

You will then need to continue to flesh out all use cases you have identified for your project.

2.4. Interface Mockups

Figure 1. This is the Menu

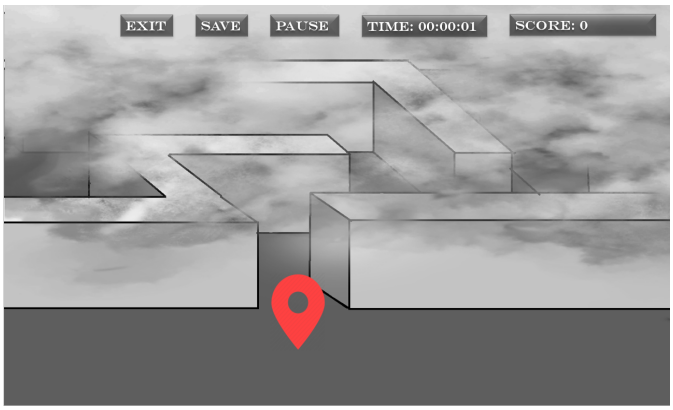

Figure 2. This is the screen where the ”New Button” transports the user

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

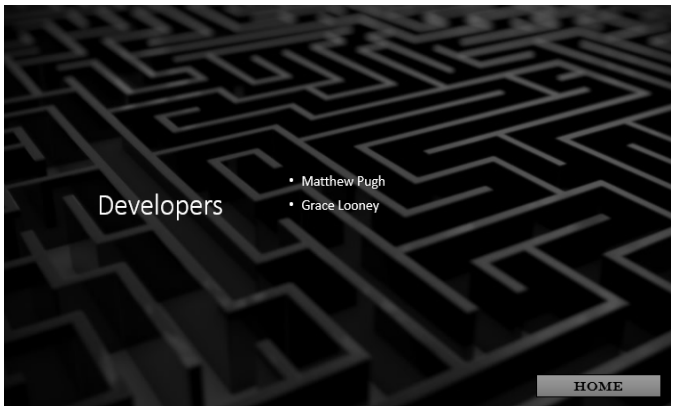

Figure 7. This is the Credits

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 mockup 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 reasearches 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

At first, this will be a little empty (it will need to be filled in by the time you turn in your final report). This is your chance to discuss all of your design decisions (consider this the README's big brother).

4.1. UML Outline

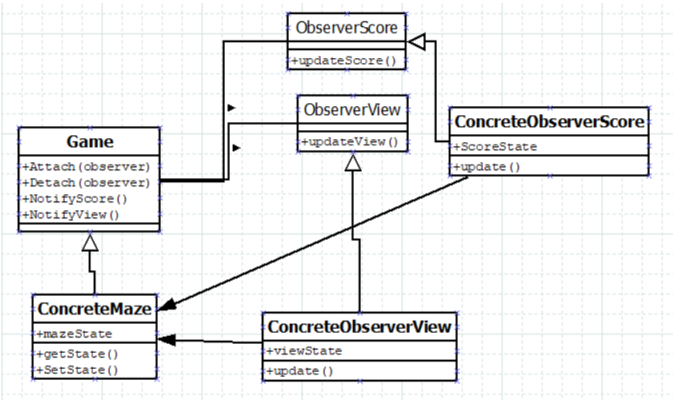

Figure 8. How we will update the GUI

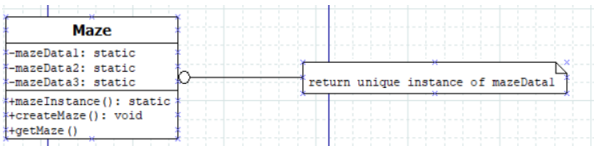

Figure 9. How we will handle the three instances of the maze

4.2. Design Patterns Used

1) Singleton
2) Factory

5. Results

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

Where are you going next with your project? For early deliverables, what are your next steps? (HINT: you will typically want to look back at your timeline and evaluate: did you meet your expected goals? Are you ahead of schedule? Did you decide to shift gears and implement a new feature?) By the end, what do you plan on doing with this project? Will you try to sell it? Set it on fire? Link to it on your resume and forget it exists?

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).