COMPUTER-ASSISTED MAZE DESIGN PROJECT

CAB302 Major Project

Final Report

The development of any application from the ground-up is a grinding task. Here, the details of our development processes will be explained as well as the functionality of our software tools and the roadblocks we faced.

Phillipe Sebastiao, Louis Yanagisawa, Ethan Mornement and Steven Jalop

Group Members

Steven	N11005980
Louis	N10221221
Phillipe	N11029935
Ethan	N10482491

Table of Contents

Introduction

Designing our software tool required constant planning throughout the process and constant reiterations. Our employer requires that it is able to allow generating, editing, storage and exporting maze designs to be used for their commercial use.

Our contractor gave us a set of requirements that the tool should be able to perform, which we have listed in below. We are happy to declare that the software tool has adequately met the requirements given to us in the Specification document, the degree of which we have specified below.

Further detailed are the steps to making the tool operational on any device (preferably PC) and the path of development that went to creating it.

Design Justification

Approach Taken

While many of the user requirements indicated specific types of mazes such as a traditional maze with arrows pointing inward and outward of the maze or a simpler maze with images doing such, the approach taken allows all these requirements to be satisfied in one location. The edit maze screen allows for images to be uploaded to change the style of maze, maze sizes of a minimum of 5x5 and maximum 100x100 allows the user to adjust the potential difficulty easily, and a cell size slider allows the user to change to cell size accordingly. This design provides alot of freedom to drawing mazes.

As requested from our client, they had given us a set of requirements in the form of user stories which we had distilled into the following user requirements:

- Automating parts of designing a maze
- Draw mazes starting from scratch or randomly generated using the GUI
- Allow Images to be Uploaded onto a maze's design
- Verify a maze can be completed with a toggleable solution
- Be able to save/load mazes from an external database
- Export maze designs from a database as an image file with/without a solution

Automating parts of designing a maze

The automation of designing the maze came from one of our members after researching online. It automatically generates a maze given dimensions of a grid as the number of cells along its length and width. The design of a generated maze prioritised the following set of requirements:

- One 'start' cell and one 'exit' cell which where always generated at opposing ends of the maze ('start' at the top left and 'end' at the bottom right) and are not walled off from the going into the maze.
- All cells that made the perimeter of the maze had a wall(s) that defined the perimeter
- There must be a start and end cell present on the grid to attempt to find a solution.
- No logo is present upon initial random generation unless an image is specified before generation of the maze design, this allows for users to decide whether to generate with or without a logo.
- A minimum grid size of 5x5 and maximum of 100x100
- The cell size can be changed to the users liking, this allows users to use this function as a 'zoom' feature and to change the size of the image exported to their liking.

The start, end and 'logo' of the maze are designated as image files, able to work with png and jpeg images. 'start' and 'end' have a default down arrow as their default image taking up a singular cell each. This gives a clear representation of a typical maze design with a start and end point along the perimeter. All these cells can be changed to an image selected from the device, allowing users to have a choice of either a standard maze design with arrows as start and end indications or one with images. This design allows us to give a 'bareback' design to making a maze as not limiting the functionality or options of a user allows for freedom and flexibility to design any maze all in one simple edit screen.

Draw Mazes starting from scratch or randomly generated using the GUI

The tool allows two methods of editing a maze; editing a generated one or making one from scratch. When making a maze from scratch, it would have no existing structure on the grid, allowing the user to have complete control of the maze's design. Given any maze design, the user has the option to place the start and end anywhere on the grid, while a logo image has to uploaded in order to be placed into the maze. This allows for more creative designs so that the tool's products are not designed monotonously. A generated maze will place the start and end cells automatically, while the logo must be uploaded before generation to be placed.

Allow Images to be Uploaded onto a Maze's Design

A user can upload their own images to change the look of the start, end and logo cells. These images are also saved to the database server when saving the maze for later use.

Toggleable Solution

Given any grid with a start and end cell present, the user can toggle the solution to indicate at any point if the maze is solvable. The maze solver finds the best solution to the maze and shows the path in red. The maze can be exported as an image with or without the solution highlighted. The solution for a maze design is toggled in the 'Edit Maze' panel and shows a path in red on the maze leading from the start cell to the end cell.

Able to save/load mazes from an external database

The user has the option to save the maze design to the database sever with a title and an author, which can be viewed in the 'Browse Mazes' section of the application, allowing for users to either continue editing the maze, or allow other users to export it. By having a readily available save/load interface, users do not lose their work between opening and closing the tool. The user can sort mazes in the browse mazes section by title, author, date created and date last edited. The database server used is MaraDB and an editable file 'db.props' is included with the database configured to this file on start-up.

Export maze designs from a database as an image file

From the editing window or the browse mazes section of the application, the user is able to export the immediate design as a 'png' image to their computer. The image can be exported with or without a solution highlighted to it. Using a 'png' image file allows for more useability for the maze as they can be given a transparent background, allowing different designs.

Additional Functionality

Given a maze design, there is an indicator displaying 2 aspects of the maze in percentages; the percentage of cells needed to travel to get from the start to the end of the maze and the number of dead ends in the maze. These numbers can give the user an understanding as to the difficulty of the maze.

Design

The design of the tool was designed to start from a 'home' page which the user can then decide as to what they want to do. Upon start-up of the tool, a small panel containing two buttons appear: 'New Maze', 'Browse Maze'. 'New Maze' would lead to the editing window of the application containing a default 10x10 maze where the user can edit, save or export the displayed maze as an image. The panels for editing/viewing the maze lines the left of the application while the design of the maze is displayed on the right. The panels are separated into 4 parts; grid dimensions and cell size, cell editing, auto-generation of the maze and finally saving/exporting/exit the maze editing window. The format of the maze is generated through a table of cells, the size defined by two integers defining the height and width of the maze. 'Browse Maze' allows users to go through a list of mazes saved to the database where maze designs and images would automatically be saved to. When opening a maze from the list in browse maze a different edit maze window will be constructed that does not allow for maze generation and changing the grid. While this could be left as the original edit maze, the purpose of viewing a maze from the browse section is to view an already created maze, not create an entirely new one.

Class Functionality and Description

Full Class and Method descriptions are available in the javadoc found in the 'javadoc' folder.

Database

Ensures there is a connection to the database using the 'db.props' files contents and connects to the database.

MazeDataSource

Interfance to provide functionality to the data source for the applicaton

MazeDataJDBC

Uses prepared SQL statements to retrieve and store maze details in the database. Extends *MazeDataSource*

BrowseMazeData

Uses *MazeDataSource*'s methods to create and instance of the data in the current running application and provide functionality for getting and setting data to the database through the GUI application.

Cell

This class is responsible for the cells that make up the maze's design. Provides functionality to drawing cells onto the grid and setting the grids cells details. The properties of these cells are manipulated in the 'EditMaze' panel through the Grid.

Grid

This class represents the design of the maze, represented by an array of Cells along its height and width. Upon detecting the properties of the cells, making the design of the maze and allows complete control to the design of the maze.

MazeFile

Class for the functionality of saving mazes and opening them. Uses a method of writing a string of chars to save the maze and reading these chars to execute methods to draw the maze based on the char in the String.

MazeGenerator

Class for generating mazes and the mazes functionalities

MazeSolver

Class for finding a path from the start to the end of the grid, uses a DFS Algorithm to do such. Used to calculate the difficulty parameters of the maze.

GUI: Browse Maze

GUI for viewing the database of all designs created

GUI: EditMaze

This GUI allows the user to fully customise the design of the maze. From here they can generate random maze designs, find the solution for a maze, change the dimensions, view their difficulty as well as save and export designs.

GUI: HomePage

The start-up panel for the tool to guide them on what they can do. Has 2 Buttons: “New Maze” which allows for designing a new default maze and “Browse Maze” for viewing the tool’s database for maze designs.

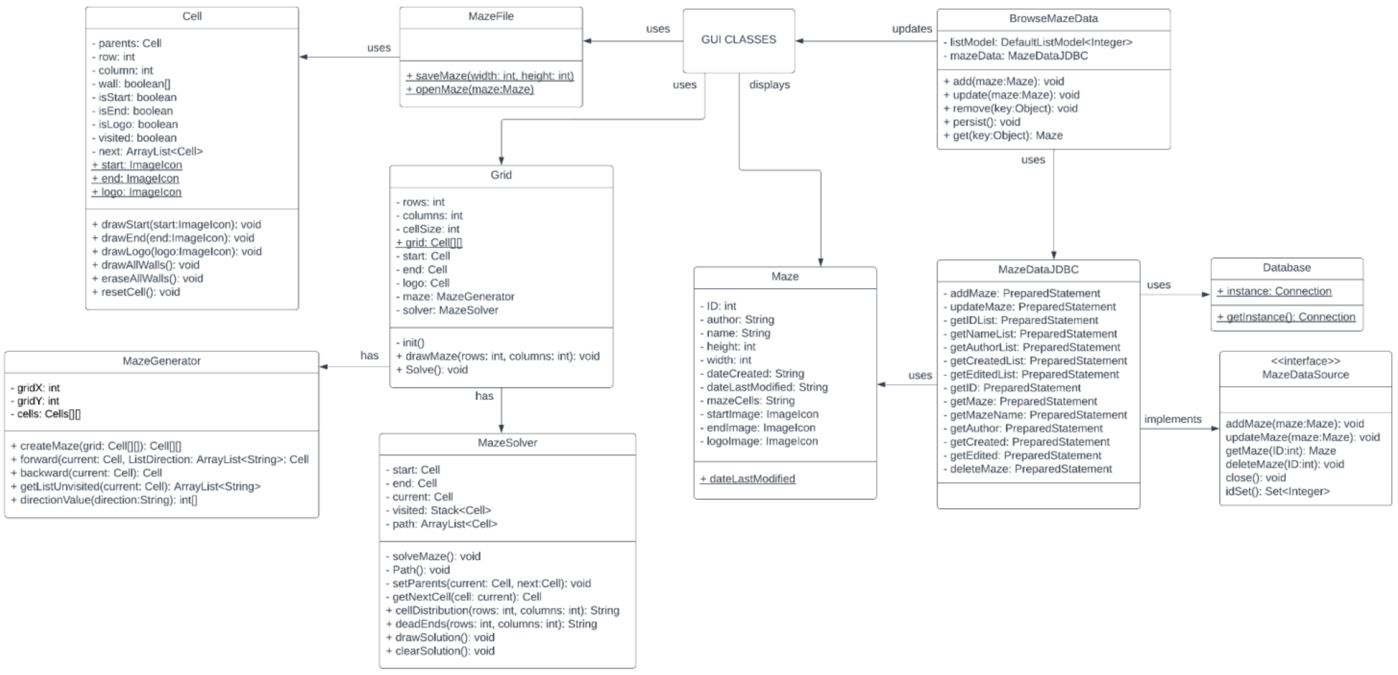
Images

Allows for maze designs to be exported as images and image manipulation.

Main

Class used to deploy the whole tool. Calls the ‘HomePage” GUI.

UML Class Diagram



(Class diagram included in assignment folder as .png)

AGILE Development

Using Agile as a basis for developing the product certainly helped organise our team but does not account for extremely tight schedules, thus lead to significant work being done over short and stressful periods compared to a more relaxed schedule as initially planned. The tool has maintained the initial structure of classes, its implementation has changed. Thankfully this was accounted for given it was the prototype’s basic and broad design.

We continually tested the tool throughout development, each main component initially able to create a GUI to perform its tasks, which we eventually linked to a ‘home page’ which encompassed the whole tool. Although initially designed to use the ‘editMaze’ GUI as the home page, the design was changed to allow simplicity, compartmentalising each section to be more manageable. The tool now segregates its functions through multiple windows; Editing/generation of designs, loading designs and attached attributes (name, author) from a database and Exporting designs as images. The selection of which was decided to go through a home page, the HomePage GUI.

The main iteration of development was changing the sequence of available actions to prevent problems due to bad design. If a user wants to export a maze design as an image, they will need to first select a maze design, thus we hid that option in the ‘EditMaze’ GUI or the database menu for selecting designs so that the immediately displayed maze would be the one exported. Upon saving a design to the database, it would also request a label and the Author’s name for the maze’s design so a user can remember their previous work if they’re doing it over many days.

Another iteration of development was creating the ‘MazeSolver’ Class. Initially this was inbuilt with the MazeGenerator and provided a solution alongside the generated maze. However, with editing of the maze, the solution given would not be correct. To correct this, the solver was given its own class 'MazeSolver' which could be called upon given any maze design, allowing the user constant notice as to the difficulty and solution of their maze.

Overall, development of the tool was not only to iron out bugs but to also streamline the processes of the tool so that the user would not be confused as to what they could do at any moment's notice. By hiding the functions of the software behind GUIs we could guide the user as to how they should use the product without resulting in errors.

Test Driven Development and Unit Testing

Full Class and Method descriptions are available in the javadoc found in the 'javadoc' folder.

For each component of the tool, unit testing assists in finding errors in the code so they can be quickly solved and fixed. Due to the complexity of designing an interface that allows for editing a maze, we focus on putting Unit Tests for aspects of the maze design rather than for all classes as other errors were easier to identify. We performed these Mock-tests on the cells that make up the grid, the grid itself that it contains all the components needed for a maze and testing the maze itself that it had solution or could generate mazes given any grid.

Class; 'CellTest'

Ensures that cells can form walls along their sides in a grid.

Class; 'GridTest'

Tests that a given maze design contains logos, start, end, cells with walls and can be edited by the user.

Class; 'MazeTest'

Tests that a maze design has appropriately sized cells, dead ends can be identified, a solution for the maze can be found and the size of the maze can be handled by the tool.

Continuous Integration

Detailed in the GitHub link in "Actions".

A vital part of AGILE development is that it is always in a constant state of deployment, such that parts of the tool are always available for operation. We have integrated Maven for Continuous Integration and is implemented with each push to main. However we do not have test cases for it so we cannot verify if test cases work, even if the build operates as it should.

AGILE Development

Using Agile as a basis for developing the product certainly helped organise our team but does not account for extremely tight schedules, thus leading to significant work being done over short and stressful periods compared to a more relaxed schedule as initially planned. The tool has maintained the initial structure of classes, its implementation has changed. Thankfully this was accounted for given it was the prototype's basic and broad design.

We continually tested the tool throughout development, each main component initially able to create a GUI to perform its tasks, which we eventually linked to a 'home page' which encompassed the whole tool. Although initially designed to use the 'editMaze' GUI as the home page, the design was changed to allow simplicity, compartmentalising each section to be more manageable. The tool now segregates its functions through multiple windows; Editing/generation of designs, loading designs and attached attributes (name, author) from a database and Exporting designs as images. The selection of which was decided to go through a home page, the HomePage GUI.

The main iteration of development was changing the sequence of available actions to prevent problems due to bad design. If a user wants to export a maze design as an image, they will need to first select a maze design, thus we hid that option in the 'EditMaze' GUI or the database menu for selecting designs so that the immediately displayed maze would be the one exported. Upon saving a design to the database, it would also request a label and the Author's name for the maze's design so a user can remember their previous work if they're doing it over many days.

Another iteration was the ‘mazeSolver

Overall, development of the tool was not only to iron out bugs but to also streamline the processes of the tool so that the user would not be confused as to what they should do at any moment’s notice.

Deployment

Setting up the Database

Before starting up the tool, the user needs to download and configure HeidiSQL and MariaDB for saving and importing designs into a database. Before Initiating the ‘main’ class, MariaDB needs to be configured and installed, keeping in mind the password you set it as. After this HeidiSQL needs to be installed. Open a new session using HeidiSQL and open ‘Manage User Authentication and Priviledges’ and create a new user account with the credentials from the ‘db.props file’ and give this user Global Privileges. Finally, create a new database with that of the name of the schema in the ‘db.props’ file, in this case ‘mazeco’. Ensure that the Project Structure’s SDK is set to Amazon 17 as well to prevent further errors, and add the ‘maradb-java-client.jar’ file to the project library. Afterwards the application can be run and will configure itself.

GUI Navigation

Upon opening the software, you will have the HomePage with 2 buttons; New Maze, which will allow for creating new designs, and Browse Mazes, which will allow the user to view previous designs that have been created and will be saved over opening and closing of the tool.

Opening ‘New Maze’ will open the ‘EditMaze’ GUI allowing the user to edit or generate new maze designs with its functions listed along the left.

The bottom-left buttons allow for the generated maze to be saved to the database or exported as an image to the user’s computer as a ‘png’ file. Upon pressing ‘save’ the user will be requested to give a label and the Author’s name for the maze’s design. After this, the user can press ‘Exit’ to return to ‘HomePage’, reminding them to save their work.

If the user presses ‘Browse Mazes’ the user will be able to view all saved designs in the database, along with their Title, Author, Date Created and Last Edited.

Here the user has the option to keep editing a file and open it in the ‘EditMaze’ GUI, delete the file or export the design as an image. Reviewing the files can be sorted by title, author, creation date or edit date.

Statement of Completeness

Requirements	Done
Start and end of a maze have indications (arrows) and are not walled off	Yes
Have a type of maze where the start and ends are replaced by images	Yes
Variable sized mazes with 100x100 being a maximum	Yes
Different sized cells for mazes	Yes
Create a realistic maze from scratch	Yes
Create a realistic maze automatically generated from maze algorithm	Yes
Can Edit maze	Yes
Logos can be manually placed	Yes
Logos can be automatically placed when randomly generating maze	Yes
Start and End images can be automatically generated and edited manually	Yes
Support for logos and start/end images	Yes
Allow for different sized logos that can act as walls in maze	No
Be able to upload logos as images that also get exported when exporting a maze as an image file	Yes
Indication at any point when editing that a maze is currently solvable	Yes
Have a togglable solution to a maze with an indication that is a different colour from the rest of the maze	Yes
When editing a maze, show the percentage of cells that are require to travel and complete the maze	Yes
Difficulty indicators	Yes
Mazes have names, author names, data and of creation and last edited	Yes
Database can be sorted by data	Yes
Mazes have time created and edited	Yes
Database uses MariaDB	Yes
Has editable properties file called 'db.props'	Yes
Database configured with the user putting a password into the db.props file	Yes
Tables created on start-up if not already created	Yes
Be able to export current maze to image to specified file with/without solution	Yes
Be able to select any number of mazes from a list to be exported as an image file to a specified folder with the option of including the solution * only one maze can be exported at a time *	*Yes
Save maze to database and open mazes from a list of mazes	Yes

GitHub Link

https://github.com/ethiom101/CAB302_2022.git