ACIT2515 - Maze Game Project Report

Ivan Chen
Bradley Gee
Arshia Aryanfar
Edward To

Team Member Contributions

Ivan Chen - I was tasked with writing the game logic and GUI. I wrote the logic behind the creation of the maze and the movement of the player. This includes the code for reading the maze from a text file, randomizing item locations, picking up items, and ending the game. I also wrote the code for the GUI, including the maze, game over and high scores screens. I wrote in-code documentation for some of the more complex code that may not be self-explanatory at a glance. I gathered, resized, and cleaned up some of the assets using an image editor to remove the white backgrounds on the item and player assets, as well as edited the door asset to make it look wider.

Edward To - I gathered the visual assets and programmed the Flask web API. This includes writing the code for the main Flask app, and designing/writing the code for a basic web page to display the scores. I also coded in the data handling for the game sending data to the server, as well as the game receiving data from the server.

Bradley Gee - I was responsible for documenting all of the classes, creating and updating the UML and writing the pytest for each class. I also helped in giving my input for the code and helped debug errors.

Arshia Aryanfar - I did the basic structure of the maze game and GUI and developed at advanced level with the help of Ivan.

UML

Refer to the documentation folder, image size too large.

Code Structure

The project is separated into the game itself and a web API. The game itself is stored in the maze directory. Within the maze directory is the file the game will be started from, main.py, and four more subdirectories. These subdirectories are the controllers, models, views, and tests directories. The controllers control the majority of the logic behind the game, such as reacting to player inputs. The views provide the GUI for the game. The models represent the virtual representations of the maze and player. Separating these aspects of the game into individual parts allows the majority of the game to remain unaffected if one part is changed.

Score Calculation
The score for the maze game is based off of how many keypresses the player used to win the game. The player's total number of keypresses is tracked and 33 (the minimum number of *single* keypresses to win) is subtracted from that total number. This difference is then subtracted from 100 to give a score.

Web API
The maze game logs the player's score as an object and then serializes the object into a JSON string. The JSON string is then sent to the Flask web server via a put request and is written to the scores.json file. This allows for the score data to be saved on the server side instead of the client side and can still be accessed by the client through a get request.

Bonus Features
Timer - We added a 20 second timer for you to collect all of the gems, and to find the end of the maze. If you do not exit the maze in 20 seconds you will lose the game.

Backpack - When you pick up an item, there is an item counter in the bottom left that shows the number of items you have picked up. Additionally, each item you pick up will have its image appear on the bottom bar as well.

Player name - After finishing the game, if you win you will be prompted for your name through a GUI. The name is capped at 5 characters with no minimum. This name and the score will then be saved to a json with all other recorded scores.