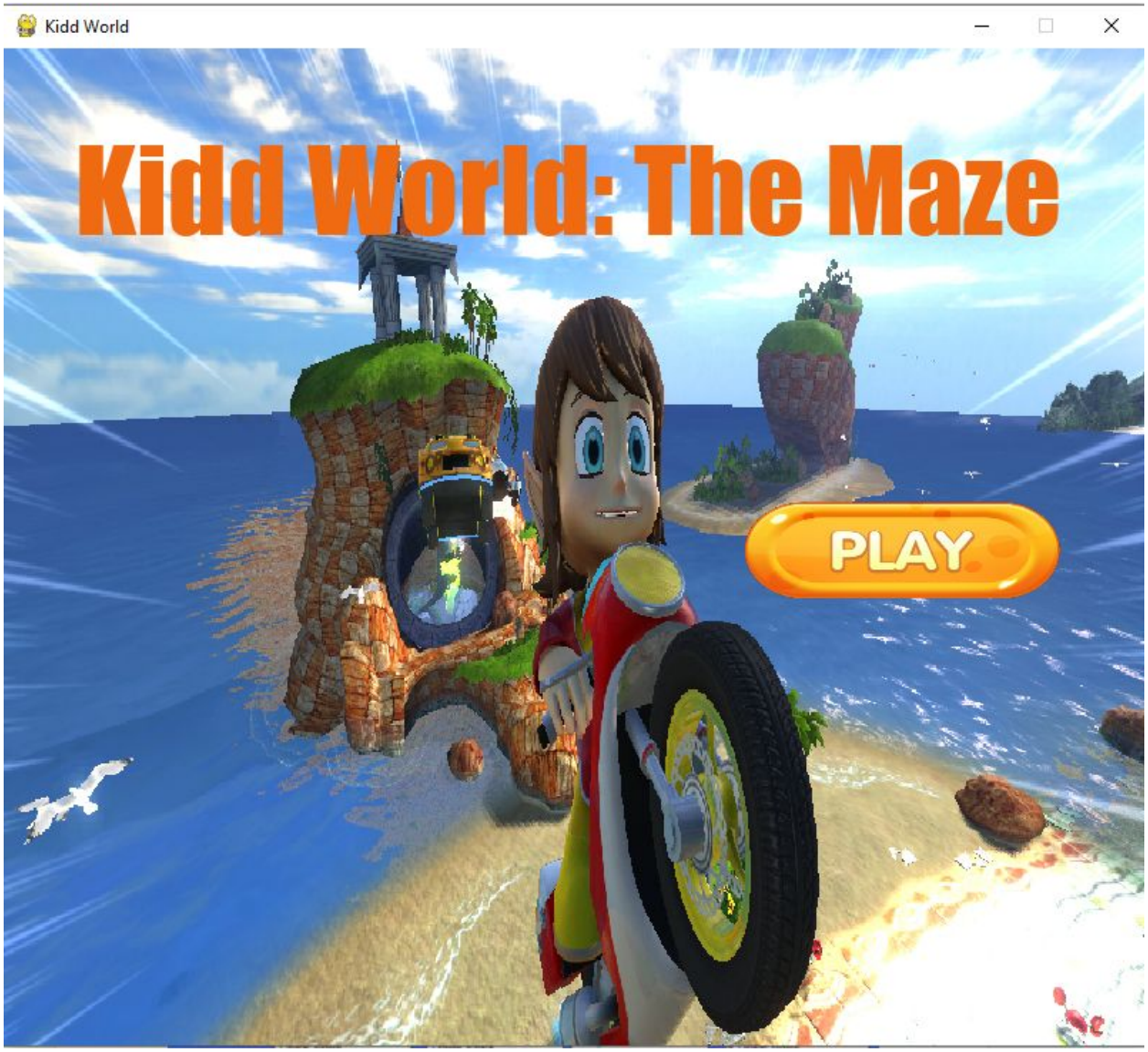
Ijaz Hussain
Taymoor Khan
Luke Salmon

Kidd World: The Maze - Report



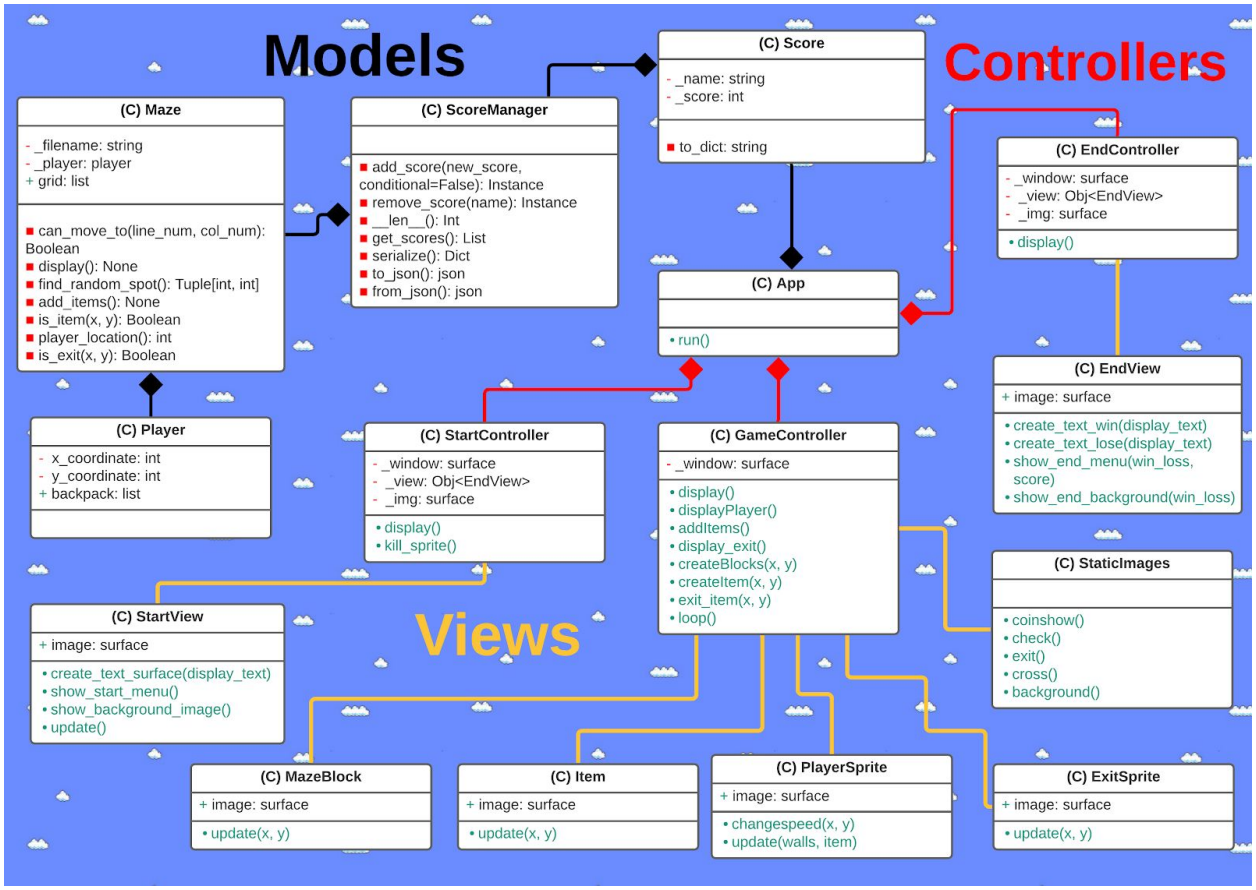
Member Contributions:

Taymoor Khan: Contributed by creating initial Player and Maze classes, implementing MVC, developed algorithms for player movement and backpack items. Contributed to pygame implementation as well as conversion from a dictionary to JSON file. Helped format and style webpage.

Ijaz Hussain: Contribution included development of the start and end views. Also, designed the UML diagram as well as contributed towards the documentation of the project. Assisted in the formatting, style and functionality of the game window as well as the web-page. Also, helped with creating the project's final report.

Luke Salmon: Contribution included testing of various packages.Also contributed to various parts of pygame implementation that included but not limited to player movement and sprite group implementation. Created Web Page and implemented route to view page.

UML Diagram



Link to full resolution PDF:

https://drive.google.com/file/d/1Lw9MKI4Qy01mD22AuFur_zRUeroC6oIz/view?usp=sharing

Code Structure

We have implemented the MVC pattern into our code structure. There are four models in our game structure. The Player class represents the player in the game. The Player has an attribute named backpack which stores the items that are picked up by the player. The Maze class represents the data about the maze. The data is loaded from a text file, which is grid_02.txt. The ScoreManager manages and serializes the scores, and the Score represents the class instances. There are three controllers as well as three views implemented in our file structure. The StartView is controlled by the StartController, the GameView is controlled by the GameController, and the EndView is controlled by the EndController. These three controllers are manipulated by the App controller. The templates folder is used to store the index.html page that our Flask application uses to display high score data. The stylesheet and background images for our Flask web page is located in the static folder. The app.py in the root directory of the game controls the Flask application. The main.py in the root directory is used to launch the App controller, which runs the game. The Pytests are run directly from the root of the maze directory. We were unable to run the Pytest files from a dedicated test folder due to us getting ModuleNotFound errors. The test runs from the maze folder using the ‘python -m pytest test_name.py’ command.

Score Calculation

The remaining time and number of coins collected will contribute to the player’s final score. We used clock.tick() with a multiple of fps to calculate the player’s score. If the player collects the 5th coin, it would be an additional 500 points. If they collect the 6th coin, it is another 500 points. These are both bonus coins in addition to the minimum of 4 coins to win.

Web API Data Storage

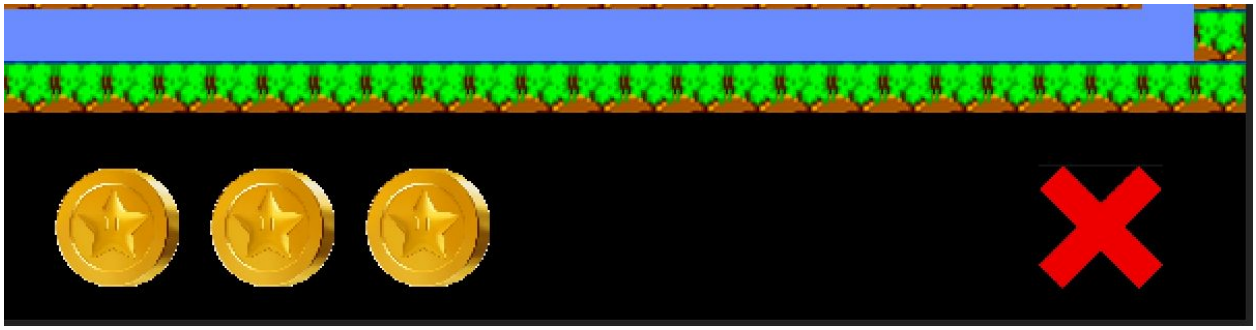
Our Flask web API writes and reads data to and from a JSON file. We chose to use JSON due to the ease of being able to serialize the player name and score. Once the data was serialized we were able to store that data in our score.json object. We were then able to use Flask and output the data from our JSON file and output it on our local web-page.

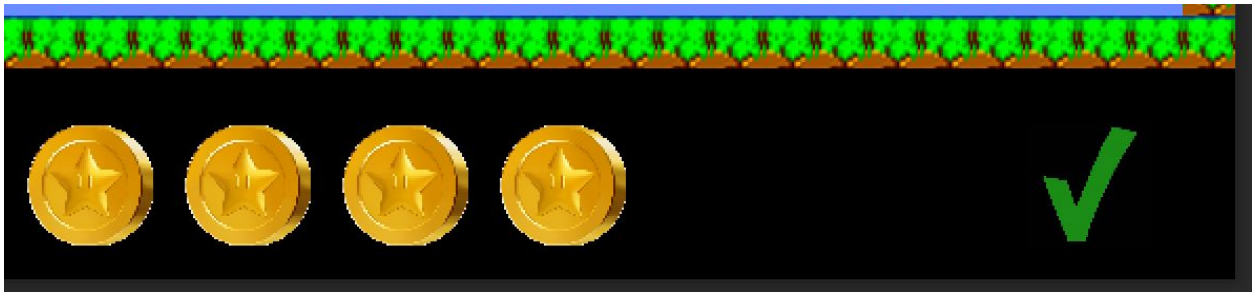
Bonus Features:

Timer

We have implemented a timer that starts off at 60 seconds. The player has 60 seconds to pick up at least 4 coins to put into their backpack or else they lose the game and will be taken to a “You Lose. Game Over.” screen. The timer starts off in green font, turns to yellow once there is less than 30 seconds left, and then red under 10 seconds to indicate time is running out.

Backpack





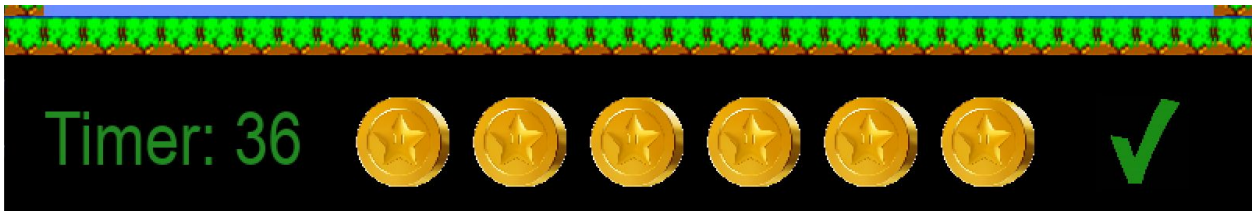
The player’s backup is displayed at the bottom of the window, next to the timer. Each time the player collects a coin, it gets added to the player’s backpack. If the player has less than 4 coin’s in their backpack, a red “x” is displayed to tell the player they require

more. Once the player has 4 coins or more, a green check mark is displayed instead. This lets the player know they have collected enough coins to win the game. The player's backpack can hold a maximum of 6 coins.

Enter Player Name through Terminal

If the player is successfully able to collect 4 or more coins and exit the maze in under 60 seconds the game will ask for the player's name. A window will appear congratulating the player then prompting them to enter their name into the console. Their name as well as score is then serialized into the scores.json file.

Bonus points if at least 5 coins collected



The player only needs 4 coins to be able to win the game. If 5 or 6 coins are collected, the player is rewarded with bonus points added onto their final score. The 5th coin is an added 500 points and the 6th coin gives an additional 500 points on top.