Project Report

Oliver Yan --- Code master

- Write the code accurately according to programming logic
- Calculation for player score
- Use of Json file (get data and put data etc…)
- Use of pygame(picture, format, showing text etc…)
- Accurate use of timer in pygame
- Accurate use of Player backpack

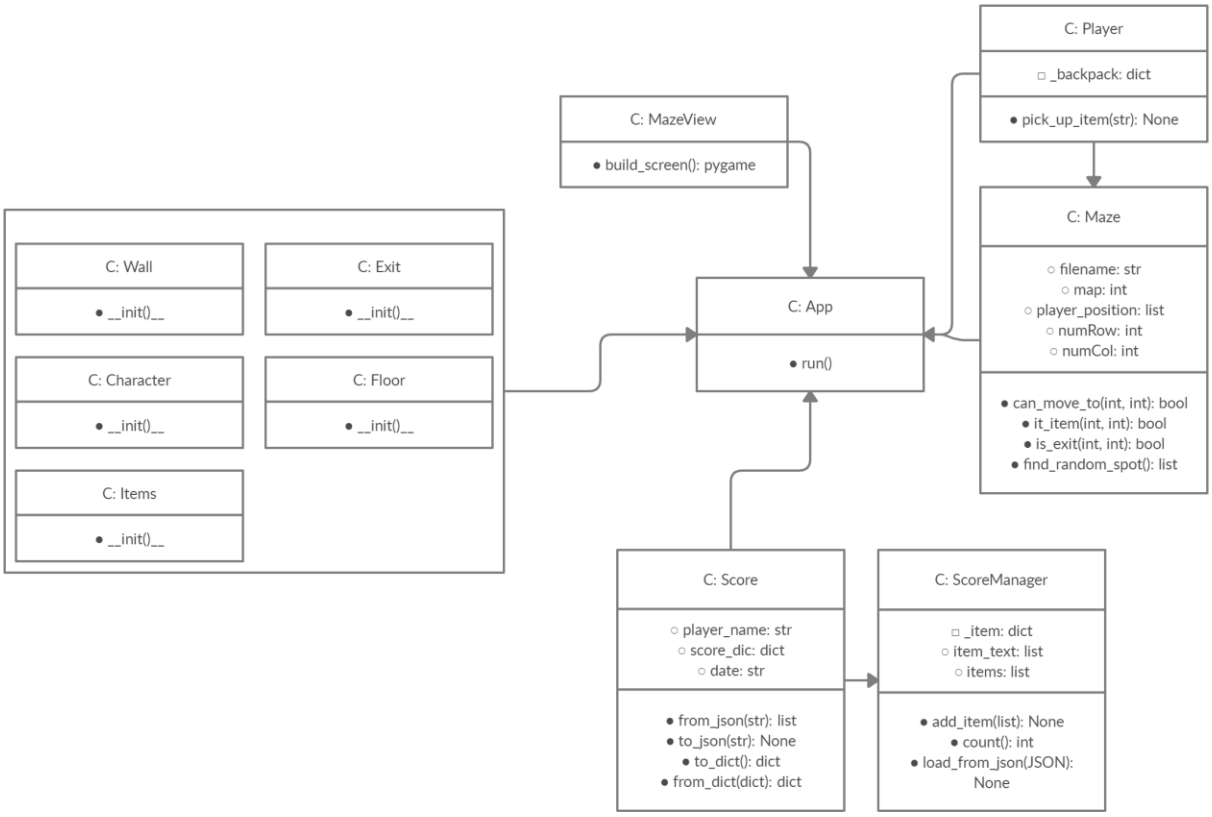
Oliver Si --- Cheer leader

- Give the programming logic of the whole project
- Use excellent coding skills to contribute
- Create the maze map, random item, random player position, player control etc
- Use of pygame(picture, format, showing text etc…)
- Bug fixing (Logic of the code, Flask API showing condition, use of html etc…)

JianFei Liu --- Code master

- Use excellent coding skills to contribute
- Web API including Flask API, persist user data etc…
- Unit Test for the properties such as maze, player, score, score_mamanegr
- Bonus feature for the pygame
- Use of pygame(picture, format, showing text etc…)

UML



Reason behind code structure

At the beginning of the project, we created a terminal type of the maze game which is working well in the terminal. As we do the group assignment part gradually, we found out that we didn't use the MVC pattern implementation for this project. After we got the feedback from demo, we tried to manage our files in MVC pattern implementation. However, we literally need to write the whole project again to achieve the MVC . Therefore, we decided to try use MVC pattern as much as we can. Basically, we use main.py to run the whole project. App.py is the main part of our project. We use app.py to build the pygame windows and interface by importing the models in the models folder including the maze map, picture displaying, and score displaying. We use web_api.py to display the Flask API and the web page based on the templates base html and

player_rank.html. Tests are all allocated in the tests folder which are test_maze, test_player, test_score and test_score_manager.

How to calculate the score when a player wins

Score = how many second you left * 1 + the number of items you pick up*10

For example, you spend 15 seconds to pick up all the item, you have 45 seconds left. The score will be 45*1+4*10 = 85.

How web API stores data and why we choose that

API will take our json file every time you run the web_api.py. We will save the game result into our json file including the name, the score, the date. And our Web api will request our json file, take the information in it and display the information on the web page. We choose JSON because we learned this approach during this semester and last semester, which is more familiar than other two approaches (CSV, SQLLite).

Bonus features

- We changed the pygame windows title from “pygame” to “maze game”. We use pygame.display.set_caption() to change the title which is coded in app.py.
- We added a background music for this game. When you start the game, you will hear an amazing background music until you quit the game. We use pygame.mixer.sound() to add this background music which is coded in app.py.