1. **Game**

In the rural farm of Stein Ville, a farmer raises havoc with the creation of violent pigs and must abandon his land by escaping the burning crops and vicious pigs. Stein must collect coins and cash to restart his farming empire in another life. Fires are made to deduct Steins points and Coins and Cash will increase Stein's points, any unfortunate encounter with a wild pig will result in instant death. Our initial UML diagram was overly simplified as none of the group members had any experience coding in Java so the final code structure was more complicated than expected. We removed the initial plan for login feature as that would involve the additional complication of data storage that we may not have time for. We did learn that testing is a great way to reveal unused or duplicate code as our test suites are focused on our main requirements and most of the lines not covered are usually not essential and can be deleted to improve overall code structure and readability.

In terms of the new structural design, we created an 'Entity' class as the parent class for the Player class that creates an instance of Stein. This parent class will allow further addition of classes with movement and keyboard control. It also tracks the collision with other objects and supports the interaction with other map components. The 'GamePanel' class acts as a stage where all the components, interactions and movements are painted onto a panel. It works with 'KeyCatcher' which reads keyboard inputs and 'CollisionChecker' to keep track of the collisions between objects.

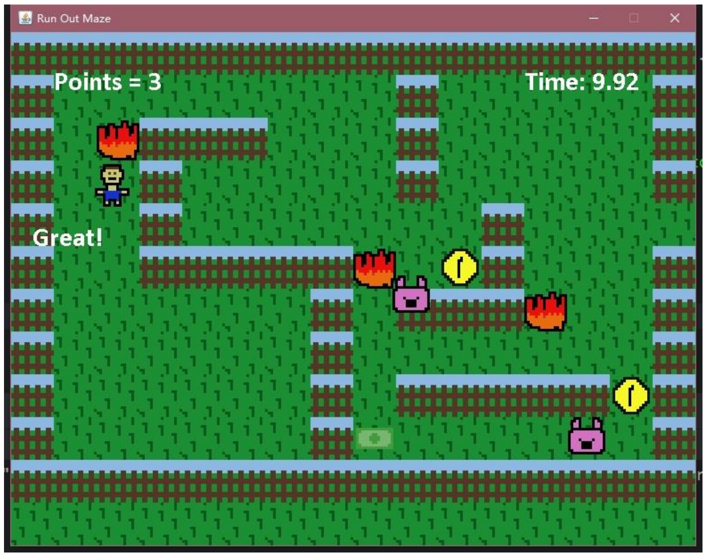
In terms of implementation, classes were made for the UI, the map components such as (player, rewards, enemies), and the interaction between the components . We extended JPanel class from the Swing package in GPanel to create the visual for the various components of the game. We also implemented the KeyListener interface from java.awt.event package to receive keyboard inputs. For testing, we have unit tests to check that the map components and the UI are properly created at the location we specify. We also have integration tests for the KeyListener implementation to ensure the correct variables are updated.

The biggest challenges were figuring which libraries and packages to use and when to create a new class versus a new method. Another challenge was writing test cases as we wrote most of our tests based on our existing codes.

For project and team management Janet worked on the Cells and UI classes and updated the other classes to use these functions along with the reports for phase 2, phase 3, and phase 4. Haoxin implemented the initial GamePanel, Main, Player, Collision and Object related classes in phase 2 and conducted structural and integration tests in phase 3 along with refectoring for phase 4. Lin wrote the javadoc and comments for all the codes in phase 2 and unit tests for GamePanel, Cells, Player, UI in phase 3. Yizhou researched information for the libraries we imported and used in the codes and did an outline for the report in phase 2 and updated the unit test and worked on phase 3 report.

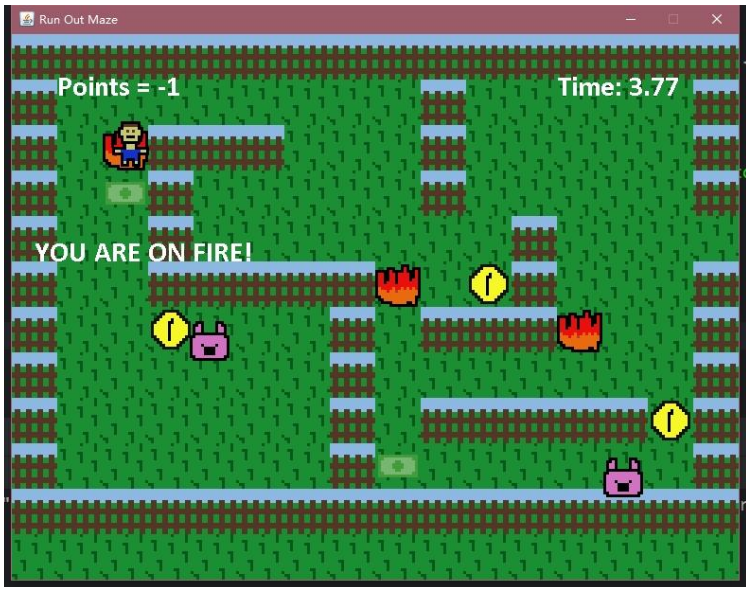
2. <u>Tutorial</u>

There are 4 main scenarios in our game such as collecting coins and cash, running into a fire, running into a pig and losing the game and winning the game by reaching the exit with a positive score. The Stein character is controlled with the WASD keys with W corresponding to moving up and ASD corresponding to left, down and right respectively. The coins add 1 point and cash add 2 points and they are stationary objects. The pigs move in random unpredictable directions so players should avoid them at all cost in order to continue the game. Fires are stationary but will deduct the overall player points. If the point reach below 0, the player will also lose the game so Fires should be avoided unless the Stein has already collected Coins and Cash.

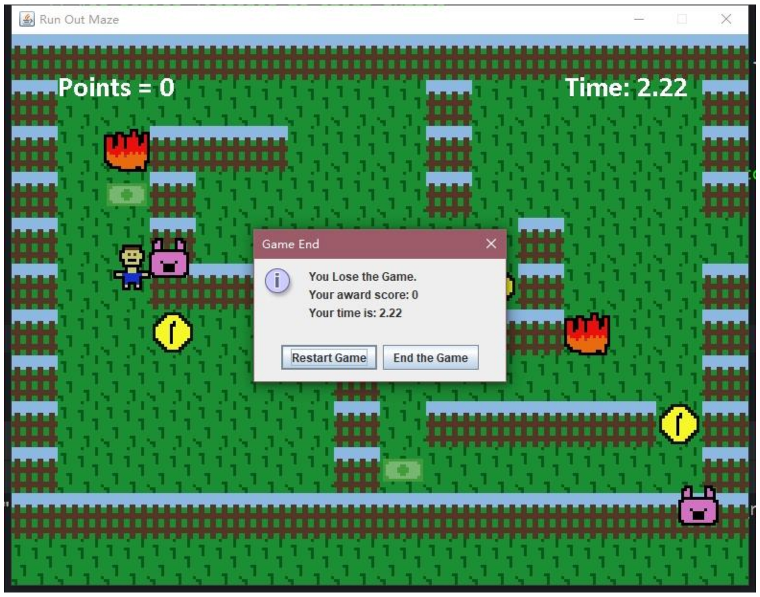


.

Scenario 1: when the player collects cash or coin and gains points. The “Great!” messaged is displayed. Coins add 1 point and Cash adds 2 points.

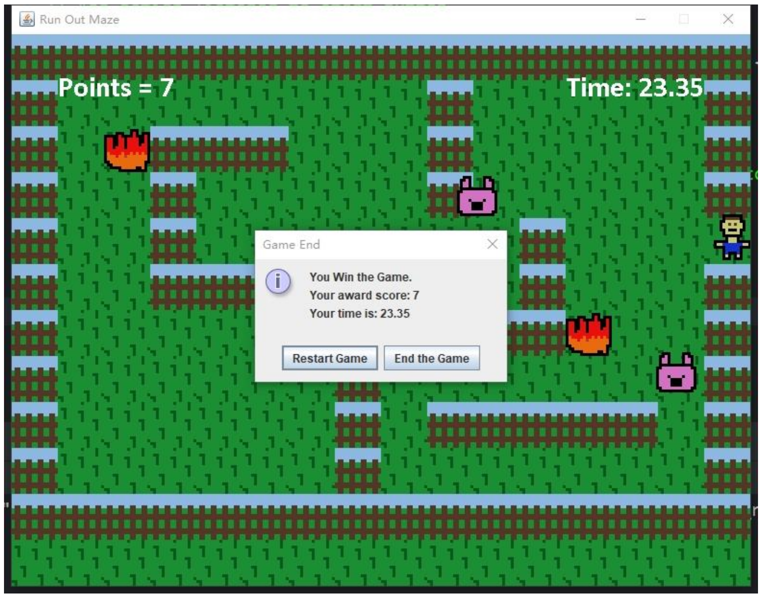


Scenario 2: When the player runs into a fire, a point is deducted and the message “YOU ARE ON FIRE!” is displayed. The game will end when the player points are negative.



You Lose the Game.
Your award score: 0
Your time is: 2.22

Scenario 3: When the player run into a pig, they lose the game instantly.



You Win the Game.
Your award score: 7
Your time is: 23.35

Scenario 4: When the player collects the rewards and reaches the exit, the game is won.