The Game

The theme of our game, CheeseRun, is built upon the classic enmity that exists between cats and mice where the cats seek to capture the mice and the mice are looking for food while trying to avoid the cats. Our main character is named Sir Mouse who happens to be an expert on all types of cheese. He has tried over 500 types but there is one type of cheese that has always eluded him: the rare Organic Cheese. No other mouse has tried this rare form of cheese because it is only found in one place, The Maze.

The Maze is a labyrinth inhabited by the ferocious cats Colonel Furball and Madame Whiskers. The Maze is meant to lure mice in search of the rare cheese and aid in their capture. Even if a mouse is able to evade the two cats, they have to watch out because Colonel Furball has planted mouse traps all over the maze to ensnare unsuspecting mice. Despite all of the risks, Sir Mouse has made the decision that he will venture into the maze and will not come back empty handed.

Implementation Updates

Since phase one when we first created our class diagram, we have used the same class structure and relationships to build our game. For example, we implemented the specialization relationship where the cheese and the organic cheese classes are both specialization of the rewards class.

One slight change we made was regarding naming. We renamed the MazeBarrier class to be called Tile and our Maze class is now called LevelOne. Aside from this, the only real changes we made were with respect to graphics: during phase one none of us had used Java Swing so we didn’t know what functions or attributes we would need in each class to display each component. We originally added generic “draw” functions to each class in our class diagram but after we learned about swing, we created one function in each class which would create the Jlabel associated with the object and added attributes to store that Jlabel and the image we wanted the jlabel to display.

In addition to the class structure we have preserved the aesthetic we had developed for our game when we were creating UI mockups during phase one. We used the same color scheme and maze design with the addition of more detailed textures for things like walls. We also created a game logo and put it at the top right of the game map.

Lessons Learned

Some of the most important lessons we have learned through building this project are in the context of refactoring, git, and testing.

The importance of refactoring became apparent to us when we were writing the MoveMouse function in the Game.java class. For our project we followed an incremental approach for development where we would implement one feature and manually test it before moving on to the next. With this approach we first made the mouse move without any checks, then we added a check for barriers, then we added a check for cheese and incrementally added checks for all objects that the mouse could encounter.

At this point we had not learned about refactoring, so with each increment we added five to ten lines of code and this code was repeated for each direction. By the time we had implemented all of the checks for the game, the moveMouse function was close to 500 lines long. Hence, testing this function was difficult since there was duplicated logic.

To improve the design of our function, and make it easier to follow, we found all occurrences of duplicated logic and extracted them to new methods. By the time we finished refactoring we had extracted 7 new methods and cut the length of function down to half. Extracting these methods really helped us find if there is a bug with that functionality, we only need to apply the fix in one place instead of several.

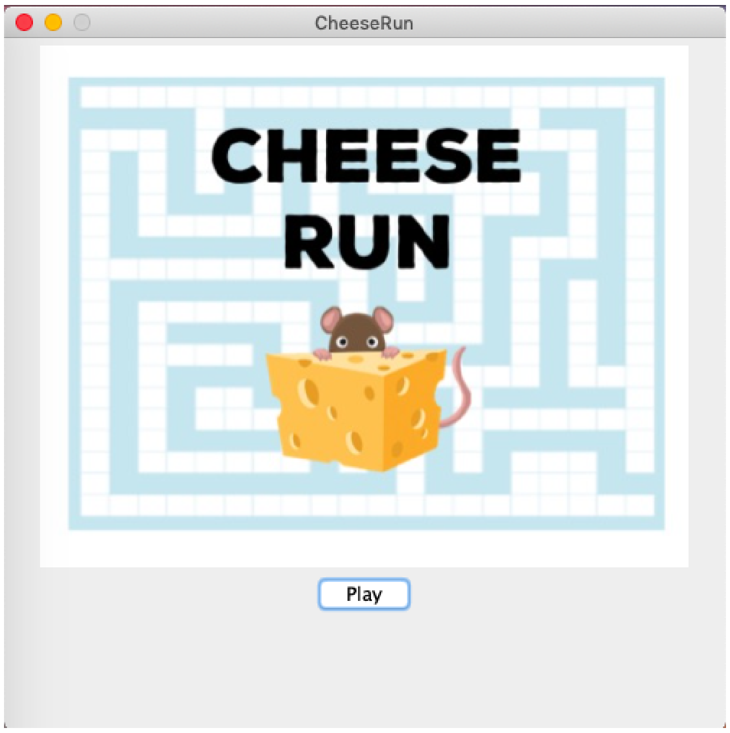
The second significant lesson we learned was with regards to the git workflow and how to coordinate our work on a shared remote repository. Prior to taking CMPT 276, we had only used git in scenarios where we were the sole contributors to a remote repo and we only needed to use the git add, git commit, git push functionality. While working on our project, we realized that working in a shared repository is quite different especially when the scale of the project is large. We learned that it is best practice, when adding a new feature, to create a branch, and implement the functionality on the local branch. Then you can merge that branch to the local master branch to resolve merge conflicts and test to make sure your code didn’t break someone else’s.

Lastly, most of us did not have any experiences with unit and integration testings prior to phase three. By learning about JUnit and creating JUnit test cases, we had the opportunity to not only further understand how each component of our game interacts with one another, but also giving us a glimpse of how software is tested in the real world.

Tutorial

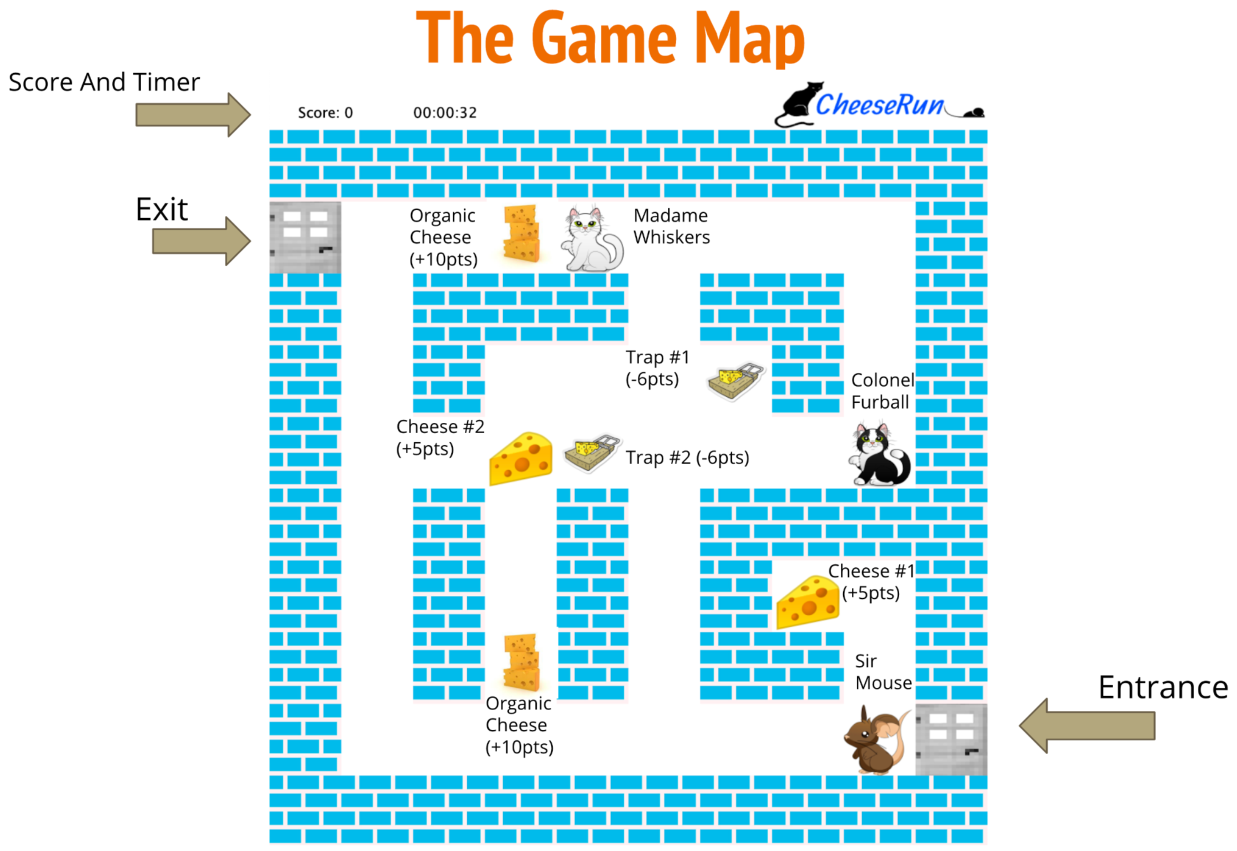
1. Getting Started

When the player runs the game, a welcome screen (below) with a “Play" button will appear. To be taken to the game page, the player simply has to click on the play button.



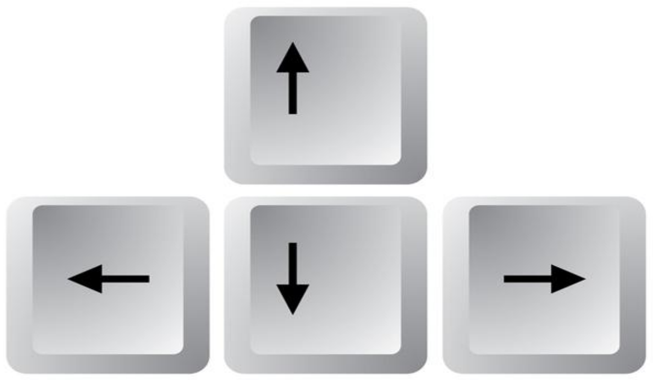
2. The Game Page

The game page consists of a score label and a timer label which are both located at the top left of the screen. It also displays the positions of the relevant objects such as the mouse, cats, cheeses and mousetraps.



3. How to control the mouse:

To navigate the maze, the player can use the four arrow keys: “Up", “Down", “Left" and “Right". Each key will move the mouse one cell in the respective direction if the movement is valid. The movement is only valid if the adjacent cell towards the inputted direction is not a barrier. If it is a barrier, the mouse’s position will not be updated.

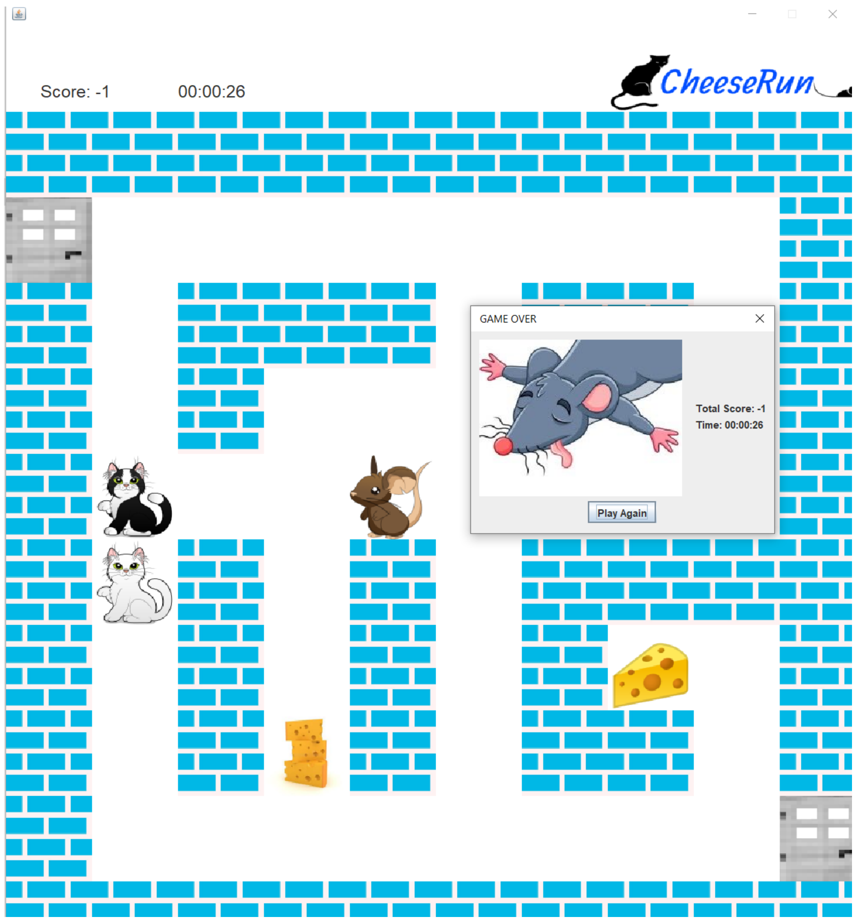


4. Things to avoid:

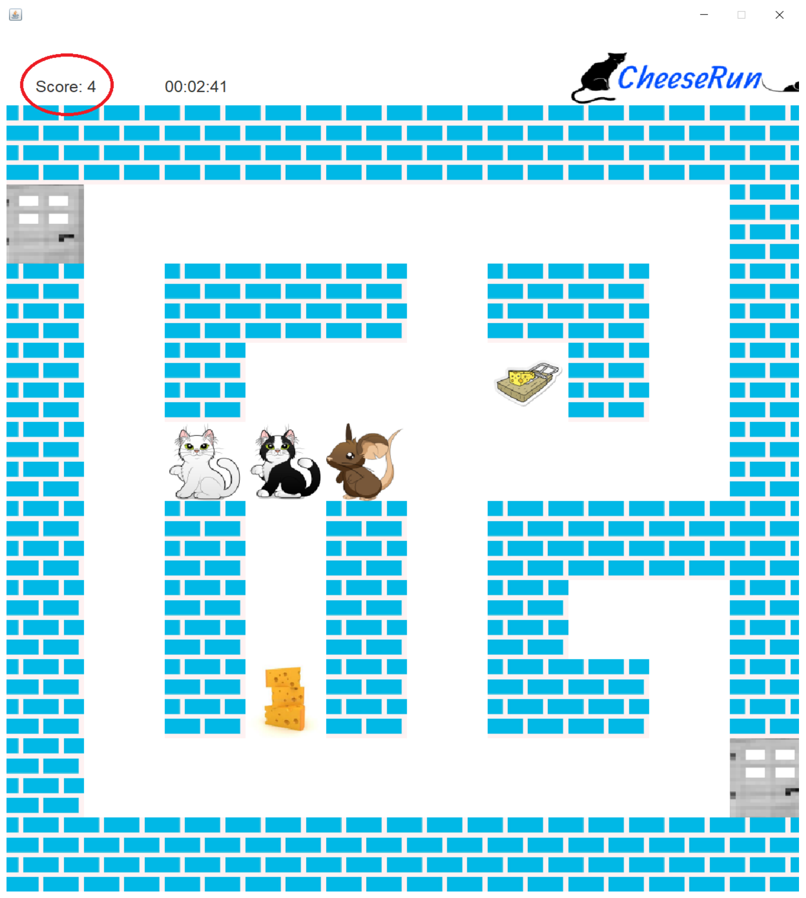
Mousetraps - Punishment

When the player (mouse) steps on a mousetrap, 6 points will be deducted from the total score. From here, two scenarios could happen:

A. If the player's score is negative (Total Score < 0): The player loses, and a "Game Over" screen will be displayed to show the player's final score and time elapsed. If the player wants to try again, they can click the "Play Again" button to reset the game.

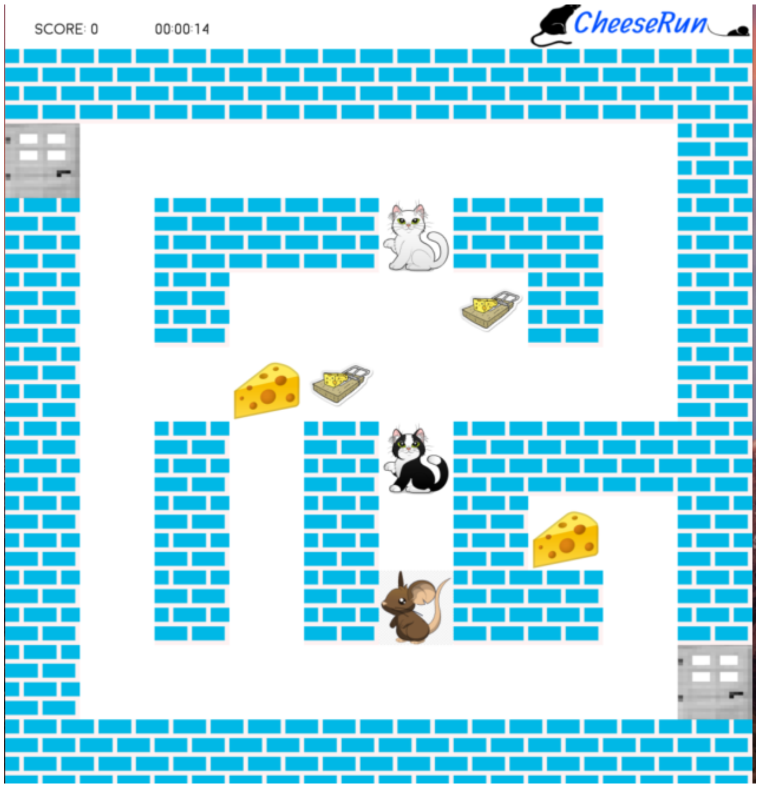


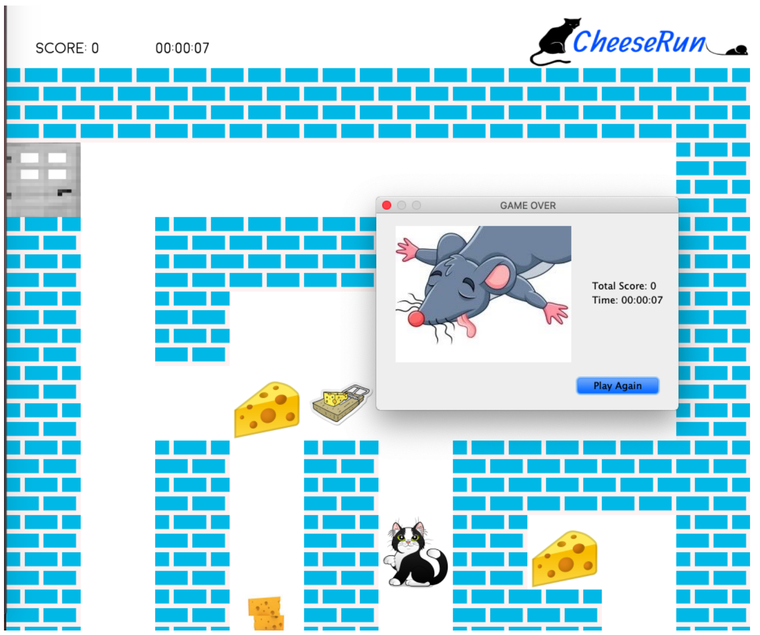
B. If the player’s score is zero or higher (Total Score >= 0): The player can continue to play and try to win the game by collecting more cheese, avoiding the cats and traps, and most importantly, reaching the exit.



Cats - Moving Enemies

In response to every movement input made by the player, the cats will try to move to an adjacent cell that will bring them closer to the mouse. Even if the mouse's movement is invalid, the cell has a barrier, the cats will move towards the mouse. If the mouse collides with either of the cats, it is game over.



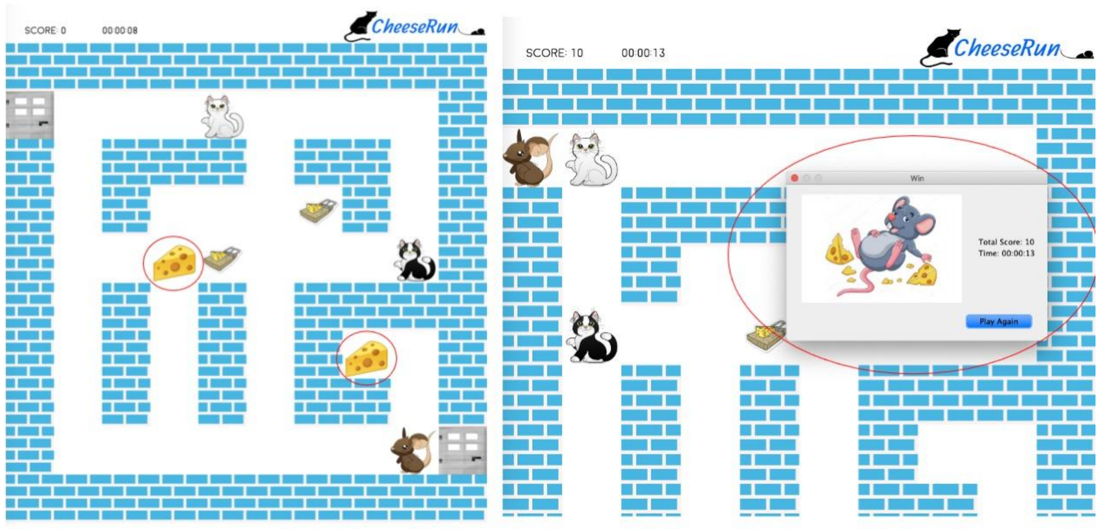


Total Score: 0
Time: 00:00:07

5. How to win:

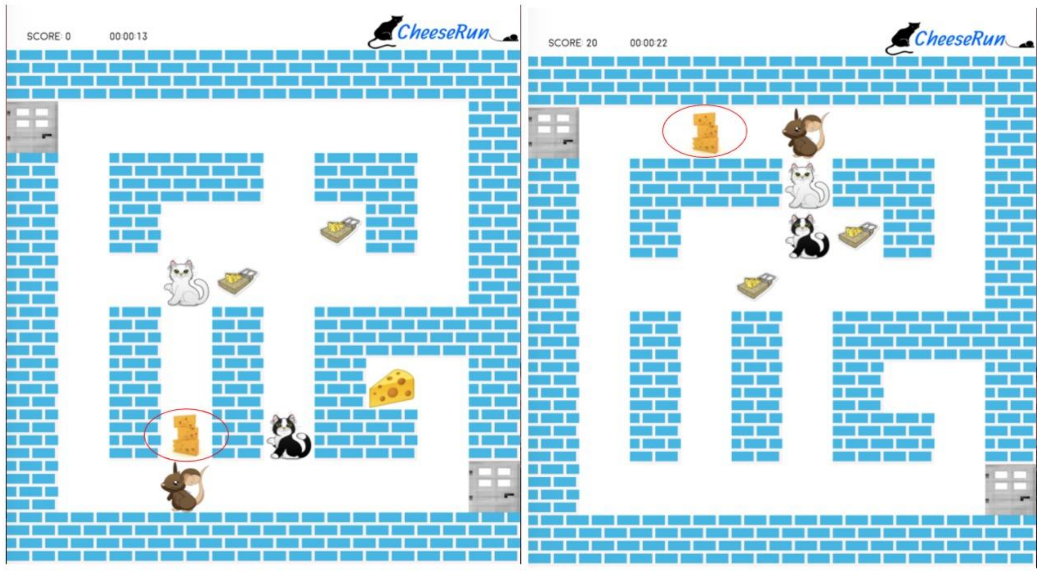
In order for the player to win, they must get the mouse to collect all of the Regular Cheese (Labels with a single cheese) before heading to the exit. If the player does not collect all of the Regular Cheese as required, they will not be able to exit the Maze to win the game.

After collecting all the Regular Cheese and upon reaching the exit, a Win screen will pop up, displaying the time elapsed and their score:



Trying to achieve the highest score:

In addition to collecting the required Regular Cheese,The player can pursue the two Organic Cheeses (Labels with a stack of cheese) which randomly appear and disappear throughout the game. Each of these rewards will add 10 points to the score.



The position of two Organic Cheeses