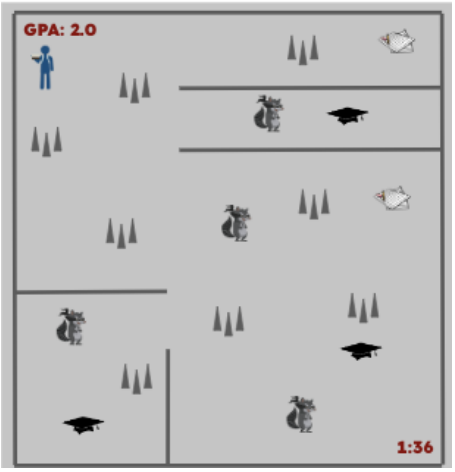
THE GAME

Our game was originally inspired by SFU, where we have the main character as a student, the enemy as a raccoon, regular rewards as a graduation cap, bonus rewards as a cheat sheet, and traps as spikes. This was consistent with what we had wanted to do since the very beginning of Phase 1. Our Phase 1 mock-up of the map stayed basically the exact same as the map we use now for the game, with some extra walls.



Originally, we wanted to add different difficulty levels for the user to choose from, but we ended up not having enough time to implement this. We also thought we would have more enemies for the user to avoid, but we found having 1 enemy was already difficult enough as our path finding algorithm worked pretty well. We wanted the user to be able to choose their main character, but again we ran out of time to implement this. The name of our game also changed from “The GPA Challenge” to “Grade Quest”, we thought the new name fit the aesthetic of our game better. We also wanted the score to start off at 2.0 since it represented the user’s GPA but this made our game too easy so we changed it to start at 0.

Features we added that were not originally planned:

- Sound (music and sound effects)
- Players can see their previous score and time when they beat the game

Features we did not implement:

- Difficulty selection
- Settings option in the menu

Lessons We Learned

Refactoring

We learned how important it is to refactor throughout the entire project life cycle. For example, our player class had a scoreObject variable, which was not the same scoreObject that the playing class uses. This meant changes in one scoreObject wouldn't reflect on the other. Instead of fixing this immediately we created a getter method to get the same scoreObject the playing class uses. This was terrible practice as it was confusing to the group since now there was an unused variable which our group members still tried to use/update when writing test cases. This ended up wasting a lot of time as we had to figure out why the score wasn't being updated correctly during testing.

Automating Tests

We learned the value of having some automated test cases, as they ensured that new features and modifications did not break any existing functionality. For example, there were times when we unknowingly introduced a bug, believing the change was inconsequential, so we didn't manually test it. We then built upon the bug, which made it harder to isolate and rectify. This could have been prevented if we created automated test cases prior to adding new features.

Using Git

We had some issues with Git at the beginning of our implementation phase. This was due to the team members not being comfortable with using Git and collaborating with other people. We learned that continuously committing and rebasing was super important for us to avoid merge conflicts and know what is already done.

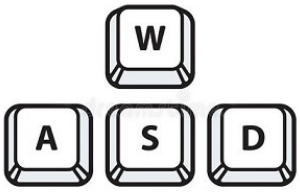
Documentation

We also struggled with documentation during our beginning stages of implementation. We found that we constantly forgot to write Javadocs for our classes and methods after implementing them. This was confusing for other members as we would have no clue what the classes/methods were intended to do, which caused us to take more time to understand the code. We learned that taking the extra time to write clear Javadocs/documentation ultimately would have saved us more time.

TUTORIAL

Instructions for our game:

- Control the movement of the Main Character using the keys:

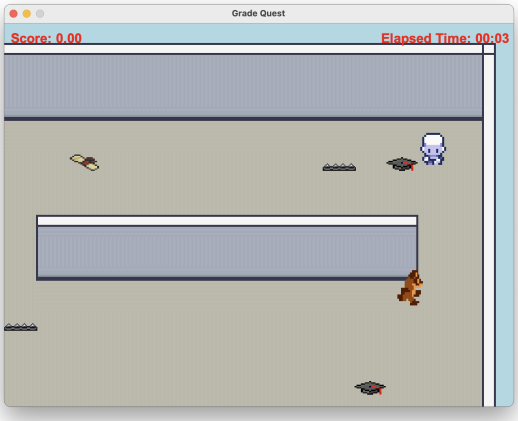


- Avoid the enemy or if he catches you, you will lose the game instantly
- Avoid the traps as it decreases your score by 1
- If your score goes below 0 you lose the game
- Collect cheat sheets to increase your score by 1
- Graduation caps increase your score by 0.5
- Collect all Graduation caps to open up the door, navigate your way to the door and you will win!

The image shown below (Figure 1) is the start screen when the game is loaded. The user can navigate the options by pressing the ‘W’, ‘S’, up arrow, or down arrow keys. Pressing ‘enter’ will select the option the ‘>’ key is currently on.

Figure 1: Start Screen

Figure 2: Gameplay



When the player selects ‘START,’ the user has entered the game and the image above (Figure 2) is presented. When the game starts, the player is randomly spawned in one of three spawn locations on the map: the top left, top right, or bottom left; in this case, the player has spawned in the top right of the map. On the top left of the game screen is the score of the user, initially beginning at 0. The top right of the screen displays the amount of time passed. Pressing the ‘esc’ key will pause the game and change the screen back to the game menu shown in the previous image. The player is the sprite on the top right in white and light blue, representing the student. The enemy is the sprite in brown, representing the raccoon. To the left of the student is

a graduation cap; to the left of the graduation cap is a trap; on the far left of the trap is a cheat sheet, representing a bonus reward that spawns randomly on the map.

Figure 3: Lose Screen

The image above (Figure 3) is the game over screen. Pressing the ‘R’ key will restart the game and pressing ‘Q’ will quit the game.

The image below (Figure 4) is the situation where the player has collected all three graduation caps on the map and the door is now open. Upon walking into the door when the door is open, the user is greeted with the win screen, shown in the image below (Figure 5). The statistics are displayed to the user including the user’s score, their time, their previous score and time, and the option to restart and quit the game.

Figure 4: Items Collected

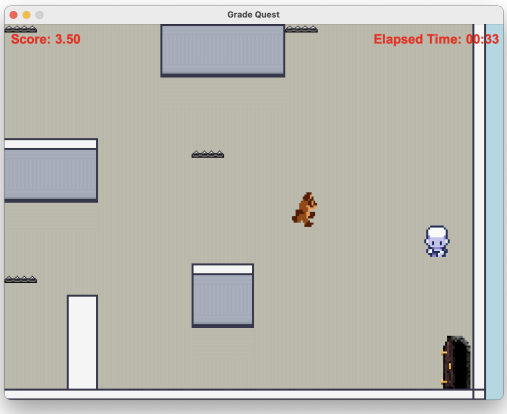


Figure 5: Win Screen