CMPT 276
Project Phase 1
Group 17

“Koi Kapers vs. Raccoon Raiders”

Colton Blackwell, Jonathan Bryan, Ridham Sharma,
Sophia Don Tranho

Statement: Utilising GitHub and JDK, we decided to develop a game centred around the SFU Koi Pond. The concept for this project was inspired by the various chairs and cones that were tossed into the frozen pond last winter.

Main Character:

- The white fish will serve as the player's avatar, initiating the adventure from the starting point
- The starting point is a cell on the boundary board of the ice wall

Enemies:

Moving Enemy: Swimming Racoons

- Punishment: Interaction with moving enemies results in points being subtracted from the player's score.
- If the player points go to 0, then game ends

Non-Animated Enemies (Penalty): Chairs, caution wet sign, cones

Rewards:

Regular Rewards: Alternate versions of koi fish (could inherit similar looking characteristics from the main character but cannot be too identical)

- When a koi fish reward is collected, they will follow behind the main character and imitate their path

Bonus Reward: Black koi fish (Only one instance of this occurring on a map)

Barriers: *Ice Blocks* obstruct movement and interaction with any animated object

- After the white fish has exited the starting point, that cell will 'freeze over' and become an ice block in order to differentiate the exit and start point

Board:

Points System:

- (+10 points) for regular rewards
- Player must collect all regular rewards in order to win the game
  - If players head to the exit before collecting all the regular rewards then the game will not end
  - It will also show a pop up message reminding the player that they need to collect all the regular rewards
- (+20 points | double the points of regular reward) for bonus rewards
- (-15 points) if in contact with stationary penalty
- (-20 points) if in contact with moving penalty
- If the total score of the player drops below 0, then the player loses
- Bonus points are received if the player completes a level under a certain time

Pop-up Rules/legend: A pop-up rule screen can provide the user with the fundamentals of the game

Time: Time that has elapsed since the start of the game

Cover Mechanics: Lily pads that act as a shelter for the main character to hide from the moving enemies

- The main character will be able to move under the lily pads and while in this state, the moving enemies will not move towards them
- If there are multiple fish following behind the main character and the main character hides under a lily pad, the tailing fish will swim to the lily pad so all of the fish will be hidden underneath the lily pad cell

Use Case: Start Game
Primary Actor: Player
Goal in context: To start and play the game
Preconditions: The game is launched and will first display the main menu
Scenario:
Basic flow

1. The user launches the game
2. The user is presented with the main menu for the game, it has a button labelled "Start"
3. The user clicks the button and the game environment and level is loaded

Priority: Moderate priority, to be implemented after basic functions
Frequency of Use: Frequent

Use Case: Checking the rules of the game
Primary Actor: Player
Goal in context: To view the rules for the game from any state of the game
Precondition: No condition other than the game being running
Scenario :
Basic flow

1. The user launches the game.
2. The user is presented with the main menu for the game, it has a button labelled "Rules"
3. The user clicks the button and is presented with the "Rules" pop-up screen which provides an overview of the game's objectives, characters, enemies, and rewards
4. After reviewing the game rules, the user clicks the "back" button to return to the main menu

Alternative flow

1. The user has already launched the game and is in the middle of playing
2. The user clicks on the pause button
3. The user clicks on the "Rules" button located on the paused game interface
4. The "Rules" pop-up screen appears
5. The user reviews the rules to get a better understanding of the game
6. The user closes the pop-up "Rules"' screen
7. The user presses the "Un-pause" button
8. The user then gets returned to the gameplay

Priority: High priority, but can be implemented after basic functions
Frequency of Use: Frequent

Use Case: Collecting the rewards (Including bonus)
Primary Actor: User
Goal in context: To collect rewards to gain more points to get a higher score.
Preconditions: The user must be actively playing a level and understand which objects are rewards and which objects are enemies.
Scenarios:

1. The user presses the 'Play' button and loads up a level
2. The user recognizes the rewards objects and the penalty objects

3. When the user goes over the same space as the rewards object, the game initiates the process of collecting the reward.
4. The reward object disappears from the level and points are added to the player's score.
5. A bonus reward object may appear at a random time and random location in the level.
6. The user can choose whether to collect the bonus reward or not. Collecting the bonus reward adds more points to the player's score and thus giving them a higher score.
7. The level requires the user to collect all the rewards before they can complete the level.

Priority: High priority
Frequency of Use: Frequent

Use Case: Consequence due to interacting with enemy objects
Primary Actor: User
Goal in context: Consequence of interacting with the enemy objects and effect on players score'
Precondition: The user must be actively playing a level and understand which objects are rewards and which objects are enemies.
Scenarios:

1. The user presses the 'Play' button and loads up a level
2. The user recognizes the rewards objects and the penalty objects
3. The user encounters moving enemies(Racoons) and stationary enemies (Chairs, caution wet sign, cones).
4. When the user passes through a space that contains a stationary enemy or if a moving enemy passes through the player space, a penalty is applied to the player's score.
5. After interaction with an enemy object, the enemy object is removed from the level.
6. If the user score goes to 0 then the level is ended and the user loses the game.
7. The user may encounter multiple instances of enemy objects in one level.

Priority: High priority
Frequency of Use: Frequent

Use Case: Scoring and Level completion
Primary Actor: User
Goal in context: Define how the user earns points and can successfully complete a level.
Preconditions: The user must be actively playing a level and understand which objects are rewards and which objects are enemies.
Scenarios:

1. The user presses the 'Play' button and loads up a level
2. The game environment is set up, including the SFU koi pond, characters, enemies and rewards.
3. The user recognizes the rewards objects and the penalty objects
4. The user can use the player score tracker to gauge their performance and progress in the level
5. The tracker is displayed in the top-right portion of the screen while the time elapsed in game is displayed on the top-left of the screen

6. The user can gain points by either collecting a regular reward or by collecting the bonus rewards
7. The user loses points when they come in contact with a enemy object, a set penalty is subtracted from the player's score depending on what kind of enemy object the user came in contact with
8. The user's goal is to accumulate as many points as possible in as less time as possible and escape the level.
9. When the user has collected all the rewards they can head toward the exit for the level, if they completed the said level under a certain time a bonus is applied to their points score.
10. After successfully completing a level the user is presented with their time elapsed in game and total points earned during that level.
11. The user can choose to continue and move on to the next level or play that level again to try and get a higher score.

Priority: High priority, but can be implemented after basic function
Frequency of Use: Frequent

Use Case: End Game
Primary Actor: user
Goal in context: To complete the game based on a win or lose scenario
Preconditions: The user must be actively playing a level and has either reached the end cell or has accumulated a negative score
Scenario:
Basic flow

1. The player has reached the end cell
2. The game confirms that all the regular rewards has been collected
3. The game displays a victory screen containing the players stats such as total score and time elapsed

Alternative flow

1. The player has reached the end cell but the game confirms that the player did not collect all the regular rewards
    a. A pop-up message will appear reminding the player that they need to collect all the regular rewards to be able to finish the level
2. The player has accumulated a negative score
    a. The game will display a game over screen with a button labelled "Restart"
    b. The player will click the restart button and the level will reset itself

Priority: Moderate priority, to be implemented after basic functions
Frequency of Use: Frequent