CMPT 276
Project Phase 2
Group 17

# *"Koi Kapers vs. Raccoon Raiders"*
# Phase 2 - Implementation

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

**Describe your overall approach to implementing the game.**

1. Organize a group meeting to allocate roles, and review the necessary UML diagram for the project.
2. Develop the 'skeleton code'  consisting of an initialized but unimplemented public class based on the UML diagram.
3. Design the game's animation sprites and scenery.
4. Watch and read tutorials about 2D game development in Java to help with enhancing necessary skills.
5. Utilize Java's Swing library, including JFrame, to outline the user interface for the game.
6. Continuous communication to prevent confusion and manage merging conflicts.
7. Integrate essential classes, such as keyHandler and collisionHandler, into the game's mechanics.
8. Write comments using JavaDoc to provide the viewer with clarifications for classes and methods.

**State and justify the adjustments and modifications to the initial design of the project (shown in class diagrams and use cases from Phase 1).**

- We hadn't considered the rendering part of the game during phase 1, we started coding we realized that some of the classes we created were redundant due to the addition of rendering classes.
- Added collision handler class (CollisionHandler):
    - Objective: Provides a way for the entire player sprite box to not be susceptible to collisions (reduce the hitbox).
    - Class is used to check if the player or any other entity is encountering a collision.
- Added key handler class
    - Objective: To enable user input player movement and to navigate through the title screen options.

    - Derived from KeyListner and encompasses methods for both keypress and release events.
- Due to time constraints, removed the fish following feature.

**Explain the management process of this phase and the division of roles and responsibilities.**

Roles were assigned during group meetings, while the majority of communication and planning occurred on a designated discord server.

Division of roles are subject to change:
- Colton: Game, Pop-up
- Jonathan: Entity & Subclasses (KoiFish, Obstacles, Player)
- Sophia: Main & Sprites
- Ridham: Board, Cell

Half-Way Update on the division of roles:
- Raccoon: Ridham
- Rules Screen: Colton
- Losing Screen: Colton
- Bonus Rewards: Sophia
- Lily pads: Johnny
- Animation: Sophia
- Check if collect all fish: Ridham
- Map Levels: Johnny

**List external libraries you used, for instance for the GUI and briefly justify the reason(s) for choosing the libraries.**

**Internal libs:**
- java.util.ArrayList
    - Used to store collected fish
- javax.swing.JFrame
    - Used for GUI programming to display the game
- java.util.Random
    - Used to randomly pick a spawn location for enemies

**Describe the measures you took to enhance the quality of your code.**
- Simplified code such as if (keyHandler.up == true) to if (keyHandler.up)
- Stuck to the UML diagram and tried to only create new classes if mandatory.

- Javadoc for comments?
- Made code that was used in multiple places into methods.

**Discuss the biggest challenges you faced during this phase.**

- Minimizing merging conflicts
- Figuring out which tutorial to follow and learning how to do Java game development for the first time
- Trying to combine the UML diagram classes with the tutorial classes
- Which features to prioritize
- Time constraints
- Better time complexity
    - Restricting the instantiation of unnecessary classes
- Trying to figure out why the images weren't loading with a transparent background
    - Turns out it was, there was just no background cell loaded underneath because we followed the tile tutorial instead of the object placement tutorial
- Adding pathfinding to raccoons so they can follow the player
    - We first tried creating a simple algorithm that makes the raccoon go toward the player's position.
    - There were a lot of problems with that implementation, we got it as close to perfect as we could but it still wasn't good enough to be used in the game
    - We decided to switch to the A* pathfinding algorithm, we followed the tutorial for its implementation and then we added movement to the raccoon based on the best path provided by the algorithm
    - We added some other special case handling like when the player is hidden raccoon moves in random directions.