CMPT276 Phase 2 Report: Group 4

Meeting Summaries

October 24th:

- Discussed the sequential diagram.

October 25th:

- Delved into understanding UML diagrams.
- Worked on task distribution among team members.
- Determined the contents to be completed by the mid-deadline.
- Discussed about whether to use JSON libraries to store player's record.

October 31st:

- Final check before halfway deadline
- Testing on all software packages to ensure seamless integration when combined.
- Identified codes that require repair and tested each objects appearance on game window.
- Determined the specific objectives and priorities for the latter half of Phase 2.
- Solicited feedback from each team member regarding their experiences, challenges faced, and suggestions for future enhancements

Strategy Overview

- We segmented the UML diagram into distinct packages, such as "player", "rewards", and "enemy".
- Each team member has been assigned a specific package for implementation.
- Inter-package communication is facilitated via the recordUsedPlace class, which maintains information about other classes.

Design Adjustments

- Incorporated additional classes for keyboard input and game window functionality.

Midway Deliverables (Nov. 1st)

Chengfeng Xiao:

- Design the Reward class.
    - Features: Collection of rewards like candies with potential for specific bonus effects in future updates. Rewards can appear and disappear during gameplay.

Li-Yu Wu:

- Construct the Enemy class.

- Enemy’s initial position
    - Add a function that generate randomly initial position for ghost and in recordUsedPlace class
    - Types: Stationary enemy (spider) and moving enemy (ghost).
    - Behavior of ghost: Pursue the player if they come within a specified range (4x4 grid).
    - *Standard ghosts cannot traverse walls, but there's a consideration for an advanced type that can. Will implement in future if time permits.
- Provide graphical assets for the player and rewards.

Nick Smart:

- Develop the Tile class that encompasses all space types on the game board.

Tianyi Tang:

- Update and submit the UML diagram incorporating the gamePanel functionalities, pop-up window, and keyboardInput implementations.
- Implement the main character's controls via the keyboard.
    - Incorporate the Keyboard class.
    - Utilize the Observer design structure to update the player or game controls.
    - Ensure character movement adheres to the WASD directions and remains within the designated range.
    - Introduce the Room class with variables determining the maximum and minimum x/y coordinates, and a method playerAvailableTiles() which invokes recordUsedPlace.

Deliverables for Final Deadline (November 12th)

Documentation:

- Javadoc comments for all classes and methods

Game Features:

- Introduce a class for tiles to display in-game images, detailing player interactions with tiles, such as wall collisions and door functionalities.
- Bonus reward, the pumpkin, should be able to appear randomly and stay for a fixed length of time and disappear.
- If time permits, we will add special feature to this bonus reward. When the main character picks up the pumpkin, it will act as a shield, making the character invisible to ghosts. Have to add a image of dog wearing a pumpkin head to illustrate.
- Finalize and polish the User Interface (UI) along with all associated graphical assets.
- Incorporate immersive music and sound effects to enhance player experience.
- Design and implement a fully functional menu, complete with responsive buttons for game navigation.

Optional Enhancements:

- Explore the feasibility of incorporating a game-saving feature, allowing players to resume their progress.

Gameplay Mechanics:

- Store both bonus and regular rewards. At the conclusion of each level, compute the total score based on rewards collected.
- Finalize the intricacies of various game levels. Introduce innovative elements to the game map..
- Implement a "Pause" feature, allowing players to halt gameplay midway. During the pause state, the game background should be grayed out, and a distinct "Paused" UI should be displayed to the player.

External Libraries Used

- Gui…and reason why we choose the libraries
- 1. **java.awt.Graphics**: This is a part of Java's abstract window toolkit (AWT) and is used for drawing shapes, text, and images onto the screen. It is crucial for rendering the game's graphics, such as the game characters, the maze, and the score display.

- 2. **javax.swing.JFrame**: JFrame is part of the Swing library, which provides a set of lightweight components for building graphical user interfaces (GUIs). We use JFrame to create the main game window, set its size, and handle close operations.

- 3. **MouseListener and MouseMotionListener**: These interfaces are used to handle mouse events. While our game is controlled by keyboard, players also use mouse listeners for navigating menus.

- 4. **KeyListener**: This interface is used to respond to keyboard events. It is essential for capturing the player's input to control the character's movement.

How We Enhance Quality of Code

- Good communication, clear class diagram

Challenges We Emcountered

- The first challenge we faced in our Java project was the asynchronous progress of work among team members, each responsible for a different package. Due to the varying work paces, there were instances where the attributes across different packages were not transmitted in a timely manner. This resulted in a delay in integration and required additional time for synchronization and verification.

- The second challenge arose during the integration of different packages, where we encountered numerous bugs attributable to the diverse coding styles of team members. Addressing these issues proved to be time-consuming and required a significant amount of effort to standardize the code and ensure consistency across the project.

- The third challenge pertained to our initial lack of understanding regarding the generation and operation of the game window. While we were proficient in expressing the methods and attributes of different packages, we were initially clueless about how to create and run the game window. It was only after extensive research and consultation of various online resources that we were able to identify the appropriate methods to generate the game, gaining valuable insights in the process.