CMPT 276 PROJECT, PHASE 2
“UNDERWATER ADVENTURES”
GROUP 19

Tommy (Seahoun) Kim - 301400674
Hazelle Lebumfacil - 301391749
Jagdeep Singh -301540139
Tsu-Erh Yu - 301426092

Section 1. Game Implementation

Underwater Adventures is a 2D arcade game implemented entirely in Java. The Java Swing GUI Widget toolkit is used to create JFrame and add a Jpanel (Gameboard) to it. The Gameboard class acts as the main container for all other components in the game. The Timer class is used to make a game loop to update and render changes on Gameboard after a short interval of time. KeyListener and Actionlistener interfaces are implemented and used to interact with the user's input and make changes on the Gameboard accordingly.

We started with an entity class which contains the coordinates and size of each entity. This class is extended by all the moving and stationary components on the Gameboard. We started the project by displaying all the moving and stationary entities on the Gameboard Panel. The Maze class was created to keep track of the position of different entities on Gameboard. Once all the entities were present on the Gameboard, we added moving functionality to the moveable entities. The next milestone for the project was to handle collisions and interactions among different entities. Additionally, changing the score, ending the game, and restarting the game were some of the tasks accomplished in this milestone to implement the functionality of the game.

Section 2. Modifications to Initial Design

Functional Changes:

- The user will press the "Enter" button on their keyboard instead of clicking on the screen to start the game
    - Justification: pressing any key in java is overly complicated and not worth attempting
    - To support this change: the following text "Please press the enter key to start" was added to the start screen
- Maps omitted from final design; keys are the only regular rewards instead
    - Justification: Having the scuba diver and the sharks already made the game difficult for the player

Design Changes:

- Moveable and Stationery Interfaces omitted from the final design
    - Justification: not enough similarity between how the move functions were implemented for entities such as the scuba diver, turtle, and shark. For stationery items, it was more intuitive to display and generate initial position within its own class
- "3 Lives" for the Main Character omitted from the final design
    - Justification: 3 lives would be difficult to implement and keep track of; more intuitive to increment and decrement the score as per the requirements
    - To support this change: all corresponding methods (ex. getNumLives() omitted)
- Logic related to collisions between user and entities implemented (not added on UML)
- Removed Seaweed class, squid punishment sufficed for character punishments, since the game was getting difficult for a player to complete

Section 3. External Libraries Used and Justifications

- AWT (java.awt)

    - **Usage:** KeyEvent, KeyListener, Graphics2D, and image classes
    - **Justifications:**
        - AWL is Java's windowing, graphics, and user-interface widget toolkit, had most of the classes and methods that we would need to implement the panel
        - A keyboard event created when a key is pressed or released, which was necessary to control the main character with user input through the keyboard
- IO (java.io)
    - **Usage:** File, IOException
    - **Justifications:**
        - Provides system input and output through data streams and file system, especially needed the file system for pulling images from the
- Util (java.util)
    - **Usage:** LinkedList and ArrayList
    - **Justifications:**
        - Library includes data structures and methods that are easy to use within our existing code. Lists were created to keep track of the number Scuba, Shark, and Squid objects that currently exist on the board
- Lang (java.lang)
    - **Usage:** Math
    - **Justification:**
        - The Scuba Diver will follow the turtle based on its distance away from the Turtle. Used the Math.abs() function to calculate this distance.
- Time (java.time)
    - **Usage:** Clock class
    - **Justification:**
        - When the player touches a Squid object, the punishment is that a large ink splatter will appear on the JFrame, distorting the player's vision of the game board. To ensure that the ink splatter is only displayed for a couple seconds, the millis() method associated with the clock class was used.
- Jframe, Jpanel (java.swing)
    - **Usage:** App class
    - **Justification**
        - It is the main window for the game. Gameboard (Jpanel) which contains all the other components is added to the Jframe

.**Section 4. Role and Responsibility Distribution**

Hazelle

- Created JPanel and JFrame with Game Background
- Implemented Maze class, MapGrid data structure
- Implemented collision handler between Turtle and Barriers
- Implemented Scuba Diver's "following" functionality
- Implemented Squid class and functionality

Files associated with implementations: *AppTest.java, GameBoard.java, Maze.java,Scubadiver.java, ScubaController.java, Squid.java, SquidController.java*

Jag

- Fixed the issue with the Background image
- Created turtle and entity class
- Implemented keyhandler class for turtle to move
- Created Regular rewards
- Added the ability to collect rewards
- Contributed to get to end screen

Files associated with implementations: *AppTest.java, GameBoard.java , Turtle.java, RegularRewards.java, BonusRewards.java, Maze.java, keyhandler.java,*

Carol (Tsu-Erh)

- Created game start screen
- Fixed the issue with background image
- Created bonus reward
- Fixed the issue with key drawing over the barrier

Files associated with implementations: *AppTest.java, GameBoard.java, BonusReward.java, UI.java,*

Tommy

- Created Shark, SharkController, Scubadiver, and ScubaController class
- Implemented collision handler between Turtle and Shark / Scuba Diver
- Implemented Game Over screen and conditions required for it
- Contributed to get to end screen
- Implemented updating Entity positions on map grid

Files associated with implementations: *AppTest.java, GameBoard.java, Shark.java, SharkController.java, Scubadiver.java, ScubaController, UI.java, KeyHandler.java,*

Section 5. Measures for Enhancing Code Quality

- Maintaining Getters and Setters where necessary
- Methods were reusable and maintainable wherever applicable
- Creating meaningful commit messages
- Follow consistent coding conventions
    - Maintaining similar coding styles and methods/ method names for implementation
    - Names for methods and variables are clear and concise
    - Leaving meaningful comments for other teammates
- Consistent communication between all members
    - Code reviews where necessary

Section 6. Biggest Challenges Faced in Phase

- Lagging in gameplay due to the threads
    - Move implementation for main character was a struggle due to this issue
- In general, updating where an entity's image should be (and for how long it should be displayed)
- Handling shared variables amongst entity class files