SNAKE GAME

User Manual

- Idea Of The game:

  Snake Game is a simple implementation of the traditional snake game. The program is developed using the Java language and the JavaFX library to create a graphical user interface. It allows the user to play with a small snake and increase its length by eating food. Points are recorded and displayed on the screen. The program contains two game modes, and the user controls the movement of the snake through the steering buttons. It checks if the snake hits itself or edges to end the game and displays the final score and player score. The program includes sound effects for various events in the game. The program provides an enjoyable experience of the traditional snake game with a simple graphical interface and sound effects.

- Discerption of the game code:

  The code consists of a main class that extends the Application class, and it contains various methods and event handlers for handling the game logic and user interactions.

  The main class represents the main entry point of the application. It sets up the game window, initializes the game elements, and handles user interactions such as button clicks and key presses. The start method is the entry point of the JavaFX application, where the game window is created and displayed. The game logic is implemented using several helper methods. Here's a brief overview of some of the key methods in the code:

  1. handleStartButton: Handles the event when the start button is clicked. It shows the game mode selection screen.
  2. handleClassicButton and handleModernButton: Handle the event when the classic or modern mode

button is clicked. They start the game in the respective mode.

3. createFood: Creates a new food item at a random position on the game window.
4. createPoison: Creates a new poison item at a random position on the game window.
5. createSnake: Creates the snake object at the center of the game window.
6. addChessBoardBackground: Adds a chessboard-like background to the game window.
7. handlepoison: Handles the event when the poison button is clicked. It creates a new poison item.
8. startGame: Starts the game by setting up the game elements, event handlers, and starting the game loop.
9. move: Moves the snake, checks for collisions with food and poison, and updates the game state accordingly.

10.adjustLocation and adjustLocation2: Adjusts the location of the snake when it goes out of bounds in classic or modern mode, respectively.

11. checkCollision: Checks if the snake has collided with the food item.
12. checkpoison: Checks if the snake has collided with any poison items.

13.checkGameOver and checkGameOver2: Checks if the game is over by checking if the snake has eaten itself or collided with poison in classic or modern mode, respectively.
14.gamaOver: Handles the game over event by showing the game over screen and stopping the game loop.
15.handleRetryButton and handleexitButton: Handle the event when the retry or exit button is clicked, respectively.
16.handlepauseButton: Handles the event when the pause button is clicked, pausing or resuming the game.
17.handleBackButton: Handles the event when the back button is clicked, returning to the game mode selection screen.

The Snake class represents the snake in the game. It contains properties and methods related to the snake's behavior and movement. Let's go through the Snake class in more detail:
Properties:

1. length: An integer representing the length of the snake, indicating how many segments it consists of.

2. tails: An ArrayList of Circle objects representing the snake's tail. Each tail segment is represented by a Circle with a center position and radius.
3. currentDirection: An instance of the Direction enum (which I'll explain shortly) representing the current direction the snake is moving in.

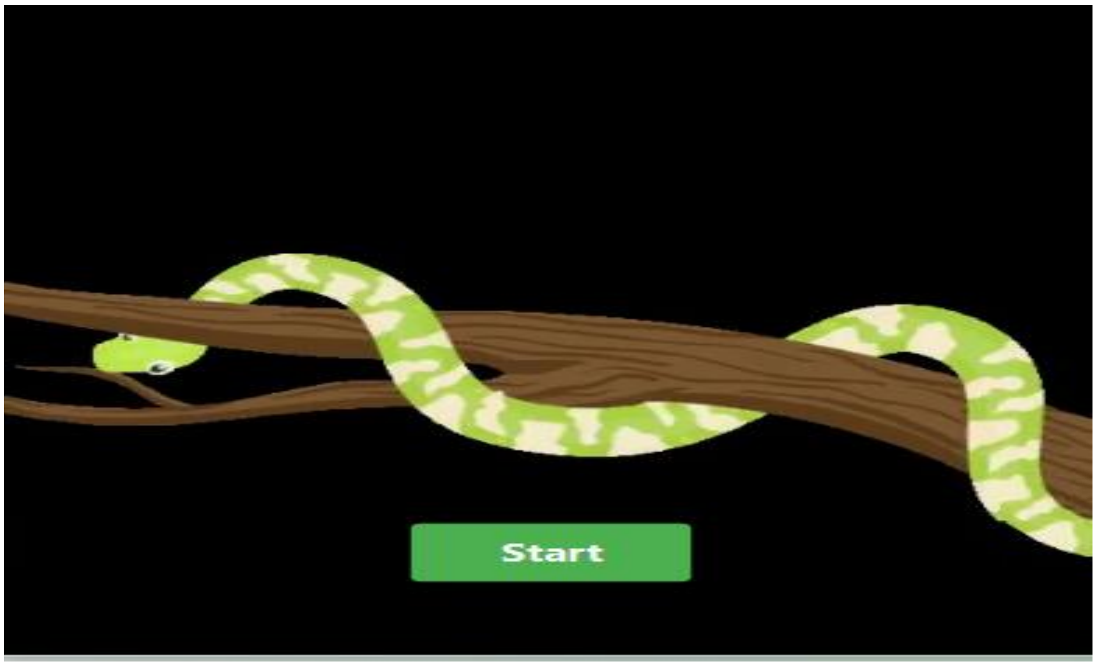
<u>Methods:</u>

1. setCurrentDirection(Direction direction): Sets the current direction of the snake.
2. getCurrentDirection(): Returns the current direction of the snake.
3. getLength(): Returns the length of the snake.
4. step(): Updates the snake's position by moving it one step in the current direction. It updates the head's position and adjusts the positions of the tail segments accordingly.
5. eatSelf(): Checks if the snake's head intersects with any part of its tail, indicating that it has eaten itself and the game should end.
6. eat(Circle food): Adds a new tail segment to the snake when it eats food. It creates a new Circle with the same position and radius as the snake's head and adds it to the tail list, increasing the length.

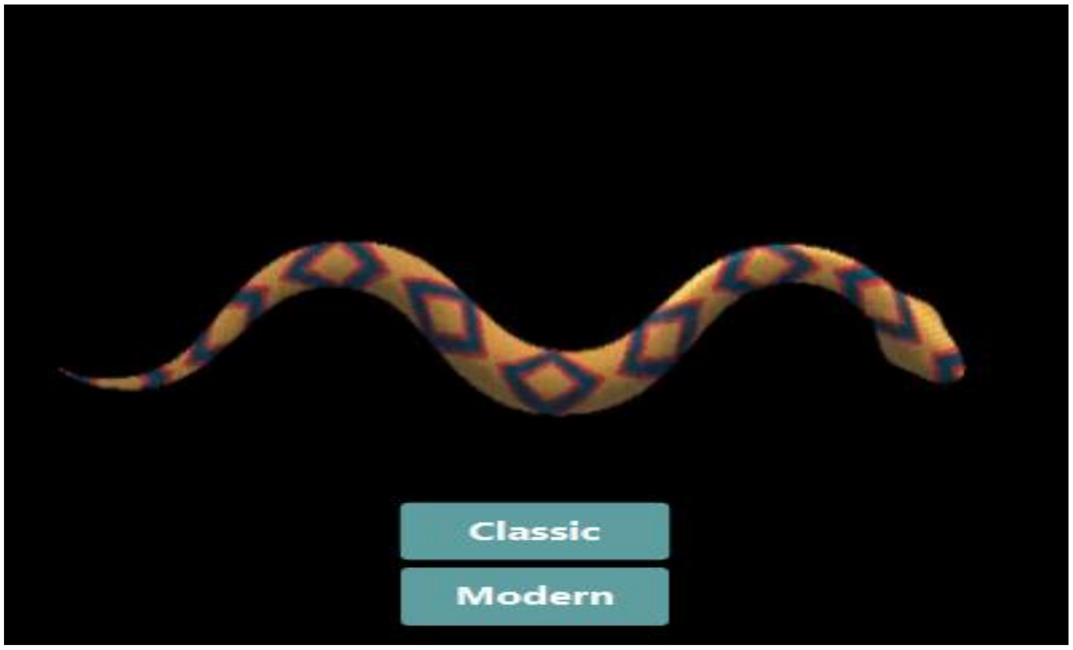
Now, let's talk about the Direction enum:
The Direction enum represents the possible directions the snake can move in. It is typically used to control the snake's movement by setting the current direction. It has four possible values: UP, DOWN, LEFT, and RIGHT. These values represent the directions in which the snake can move.

- Overview of the game shape:

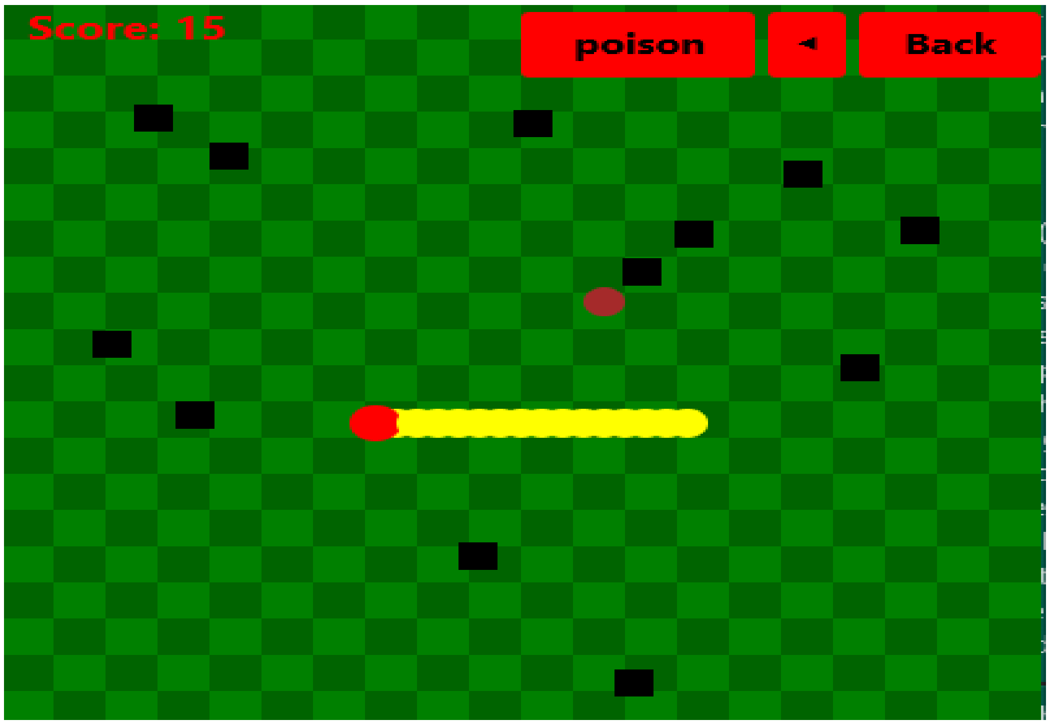
1. *Start game page:*



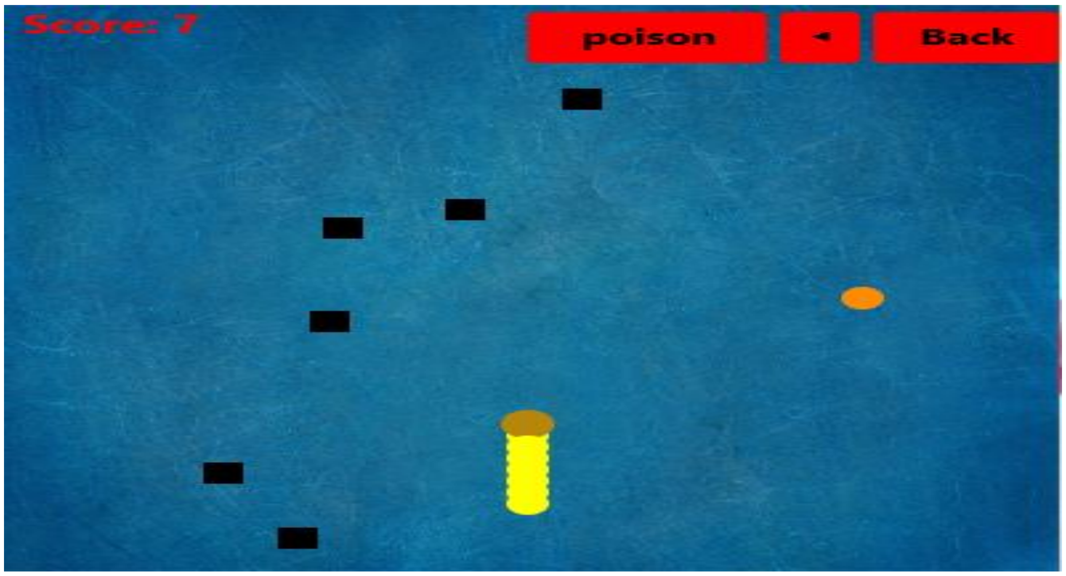
2. Mode game selection page:



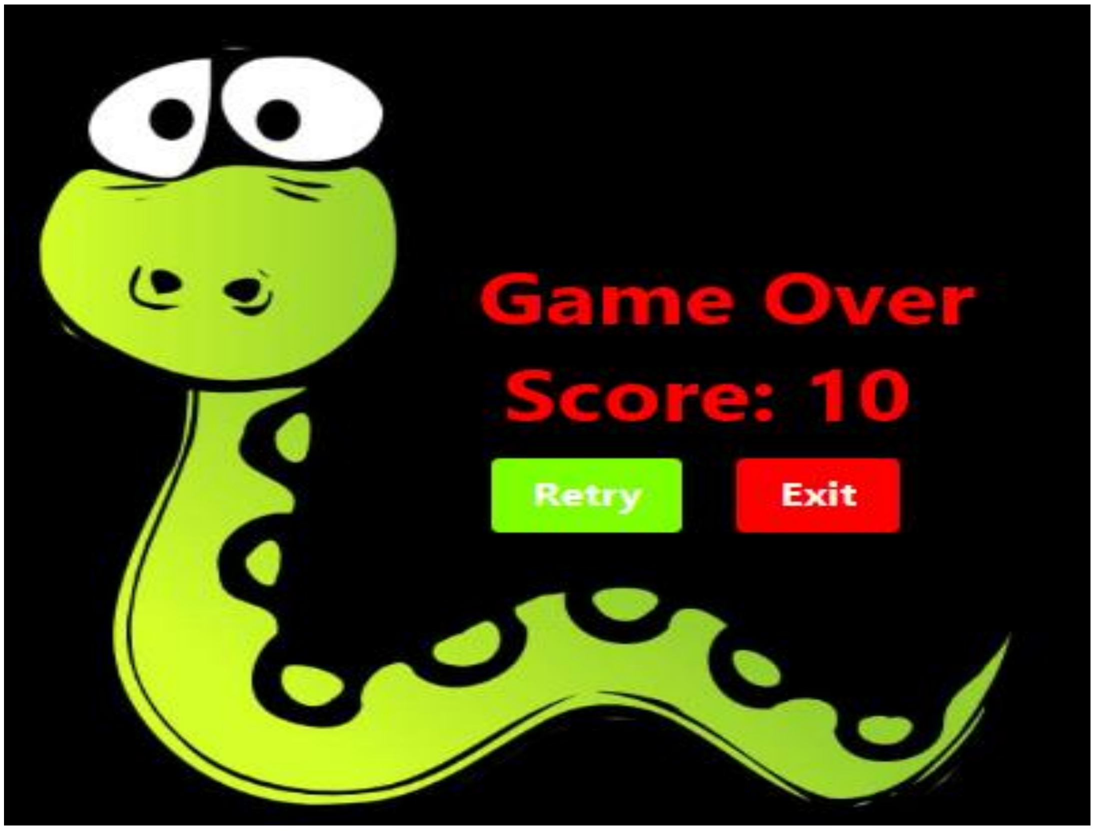
3. Classic mode:



4. Modern mode:



5. Game Over page:



Thank You