Define constants:

- screWidth as 800

- screHt as 600

- cellSize as 20

- ballSize as 10

- trophySize as 20

- Colors for WHITE, BLACK, RED, BLUE, GREEN

- wallProb as 0.3 (Probability of wall generation in the maze)

Class Intro:

Initialize with optional score parameter:

- Set up the welcome screen image

- Define buttons and labels for instructions and navigation

- Define play and quit buttons

Process interactions:

- If play button is clicked, change status to "play" and stop intro

- If quit button is clicked, change status to "quit" and stop intro

Class MazeGame:

Initialize:

- Set screen size

- Define ball and trophy rectangles

- Generate maze

- Place ball and trophy in the maze

Start game:

- Initialize game components (Pygame setup)

- Run game loop until window is closed or player wins

Run game loop:

- Display win screen if player wins

- Otherwise, handle events, update ball position, check for collisions, and redraw game scene

Handle events:

- Handle quit and restart (R key) during game or win screen

Update ball position:

- Move ball towards mouse position, avoiding walls

Draw game scene:

- Draw maze, ball, and trophy

Generate maze:

- Create a grid of cells where each cell can either be empty or a wall based on wallProb

Check collision:

- Check if ball has reached the trophy

Display win screen:

- Display "You Win!" message and restart instruction

Restart game:

- Regenerate maze, reposition ball and trophy

Spawn ball:

- Place ball in a random location that is not a wall

Spawn trophy:

- Place trophy in a random location that is not a wall

Main function:

- Start with the intro scene

- Transition to game if "play" is selected

- Run game

Start program:

- Execute main function

**Main 3 functions broke down further:**

**1. updateBall(self)**

• updates the position of the ball based on the mouse cursor's position.

• calculates the direction vector from the current ball position to the mouse cursor position.

• If the direction vector has a non-zero length (meaning the mouse cursor is not on top of the ball), it normalizes the direction vector to have a length of 1.

• It then calculates the new position of the ball by adding the normalized direction vector multiplied by a speed factor (5 in this case) to the current ball position.

• Next, it creates a new rectangle representing the ball's position after the update.

• It checks if this new position intersects with any maze walls. If not, it updates the ball's position to the new position.

**2. drawGame(self)**

• This method draws the game elements on the screen.

• It first fills the screen with a black color.

• Then, it iterates over each cell in the maze.

• If a cell contains a wall (value of 1), it draws a white rectangle representing the wall.

• Next, it draws the ball as a blue circle using the ball's position and size.

• Finally, it draws the trophy as a red rectangle using the trophy's position and size.

**3. generate\_maze(self)**

• This method generates a random maze.

• It initializes a 2D list representing the maze, where each element is initially set to 0 (0 = no wall).

• It iterates over each cell in the maze.

• For each cell, it randomly decides whether to place a wall based on the wallProb probability.

• It ensures that the maze starts and ends without walls by excluding the first and last cells from having walls.