UNIVERSITY OF ENGINEERING AND TECHNOLOGY, TAXILA

PROJECT REPORT

OOP LAB

SUBMITTED BY:

Ifrah Sarfraz (23-SE-09)

Laiba Ashfaq (23-SE-55)

SUBMITTED TO:

Engr. Sidra Shafi

SOFTWARE ENGINEERING DEPARTMENT

PACMAN GAME IN JAVAFX

The Pacman Maze Game is a desktop application developed in Java using the JavaFX library. The game emulates the classic Pacman experience, featuring player control, interactive ghosts, and a maze with collectible dots.

Key Features:

- **Pacman Movement**: The player navigates the maze using arrow keys.
- **Ghost AI**: Ghosts move randomly to challenge the player.
- **Score System**: Points are earned by collecting dots.
- **Life System**: Player has three lives; collision with ghosts reduces lives.
- **Game Status**: Messages display game state (e.g., win, lose)

Initial Game Setup:

```
73      public void start(Stage primaryStage) {
74          VBox root = new VBox(); // Use VBox to stack game area and info area
75
76          // Create the game pane
77          Pane gamePane = new Pane();
78          Rectangle background = new Rectangle(CELL_SIZE * COLS, CELL_SIZE * ROWS);
79          background.setFill(Color.rgb(0, 0, 51));
80          gamePane.getChildren().add(background);
81
82          drawMaze(gamePane);
83
84          // Calculate the center position for Pacman (one row down)
85          int centerRow = ROWS / 2 + 1;
86          int centerCol = COLS / 2;
87          double centerX = centerCol * CELL_SIZE;
88          double centerY = centerRow * CELL_SIZE;
89
90          // Set up Pacman at the center of the maze
91          pacman = new ImageView(new Image("file:///C:/Users/Cv/OneDrive/Pictures/Saved%20Pictures/pacmanLeft.gif"));
92          pacman.setFitWidth(pacmanSize);
93          pacman.setFitHeight(pacmanSize);
94          pacman.setX(centerX);
95          pacman.setY(centerY);
96          gamePane.getChildren().add(pacman);
97
98          createDots(gamePane);
99          createGhosts(gamePane);
100
101         // Create the info area
102         VBox infoBox = new VBox();
103         infoBox.setStyle("-fx-background-color:rgb(0, 0, 30); -fx-padding: 10;");
104
105         // Create an HBox for score and lives
106         HBox scoreAndLivesBox = new HBox();
107         scoreAndLivesBox.setSpacing(20);
108         scoreAndLivesBox.setAlignment(Pos.CENTER_LEFT); // Align items to the left
109
110         scoreText = new Text("Score: " + score);
111         scoreText.setStyle("-fx-fill: white; -fx-font-size: 16; -fx-font-weight: bold;");
```

```
112         livesText = new Text("Lives: " + lives);
113         livesText.setStyle("-fx-fill: white; -fx-font-size: 16; -fx-font-weight: bold;");
114
115         scoreAndLivesBox.getChildren().addAll(scoreText, livesText);
116
117         gameMessageText = new Text("Press 'S' to Start Game");
118         gameMessageText.setStyle("-fx-font-size: 16; -fx-font-weight: bold; -fx-fill: white;");
119
120         infoBox.getChildren().addAll(scoreAndLivesBox, gameMessageText);
121
122         root.getChildren().addAll(gamePane, infoBox);
123
124         // Set up scene and key press handling
125         int infoBoxHeight = 60;
126         Scene scene = new Scene(root, CELL_SIZE * COLS, CELL_SIZE * ROWS + infoBoxHeight);
127         scene.setOnKeyPressed(this::handleKeyPress);
128
129         // Set up game loop for constant updating
130         Timeline timeline = new Timeline(new KeyFrame(Duration.millis(50), e -> {
131             if (gameStarted && !gameOver) {
132                 updateGame(gamePane);}
133         }));
134         timeline.setCycleCount(Timeline.INDEFINITE);
135         timeline.play();
136
137         primaryStage.setTitle("Pacman Maze Game");
138         primaryStage.setScene(scene);
139         primaryStage.setResizable(false);
140         primaryStage.show();}
```

Maze Rendering: The maze is a grid-based layout where 1 represents walls and 0 represents paths. Walls are visualized using rectangles, and the maze is drawn dynamically.

```
141     private void drawMaze(Pane root) {
142         for (int row = 0; row < ROWS; row++) {
143             for (int col = 0; col < COLS; col++) {
144                 if (maze[row][col] == 1) {
145                     Rectangle wall = new Rectangle(CELL_SIZE * col, CELL_SIZE * row, CELL_SIZE, CELL_SIZE);
146                     wall.setFill(Color.rgb(30, 80, 130));
147                     root.getChildren().add(wall);
148                 }
149             }
150         }
151     }
```

Player Controls: Players move Pacman using arrow keys. Movement checks if the destination is a valid path.

```
153     private void handleKeyPress(KeyEvent event) {
154         if (!gameStarted) {
155             if (event.getCode() == javafx.scene.input.KeyCode.S) {
156                 gameStarted = true;
157                 gameMessageText.setText(""); // Clear the start message
158             }
159             return;
160         }
161         if (gameOver) {
162             if (event.getCode() == javafx.scene.input.KeyCode.R) {
163                 restartGame();
164             }
165             return;  }
166         double newX = pacman.getX();
167         double newY = pacman.getY();
168
```

```
169         switch (event.getCode()) {
170             case UP:
171                 newY -= CELL_SIZE;
172                 pacman.setImage(new Image("file:///C:/Users/Cv/OneDrive/Pictures/Saved%20Pictures/pacmanUp.gif"));
173                 break;
174             case DOWN:
175                 newY += CELL_SIZE;
176                 pacman.setImage(new Image("file:///C:/Users/Cv/OneDrive/Pictures/Saved%20Pictures/pacmanDown.gif"));
177                 break;
178             case LEFT:
179                 newX -= CELL_SIZE;
180                 pacman.setImage(new Image("file:///C:/Users/Cv/OneDrive/Pictures/Saved%20Pictures/pacmanLeft.gif"));
181                 break;
182             case RIGHT:
183                 newX += CELL_SIZE;
184                 pacman.setImage(new Image("file:///C:/Users/Cv/OneDrive/Pictures/Saved%20Pictures/pacmanRight.gif"));
185                 break;
186         }
187         if (canMove(newX, newY)) {
188             pacman.setX(newX);
189             pacman.setY(newY);
190         }
191     }
```

Collision Detection: The game checks collisions between Pacman, dots, and ghosts.

```
280     private void checkCollisions(Pane root) {
281         // Check for dot collisions
282         dots.removeIf(dot -> {
283             if (pacman.getBoundsInParent().intersects(dot.getBoundsInParent())) {
284                 root.getChildren().remove(dot);
285                 score += 10;
286                 scoreText.setText("Score: " + score);
287                 return true;
288             }
289             return false;
290         });
291         // Check if all dots are eaten
292         if (dots.isEmpty()) {
293             gameOver = true;
294             gameMessageText.setText("You Win! Final Score: " + score + ". Press 'R' to Restart.");
295             return;
296         }
297         // Check for ghost collisions
298         for (ImageView ghost : ghosts) {
299             if (pacman.getBoundsInParent().intersects(ghost.getBoundsInParent()) && canCollide) {
300                 lives--;
301                 livesText.setText("Lives: " + lives);
302                 resetGhostPositions();
303                 if (lives == 0) {
304                     gameOver = true;
305                     gameMessageText.setText("Game Over! Final Score: " + score + ". Press 'R' to Restart.");
306                 } else {
307                     canCollide = false;
308                     pacman.setOpacity(0.5);
309                     Timeline timeline = new Timeline(new KeyFrame(Duration.seconds(1), e -> {
310                         canCollide = true;
311                         pacman.setOpacity(1);
312                     }));
313                     timeline.play();
314                 }
315                 break;
316             }
317         }
318     }
```

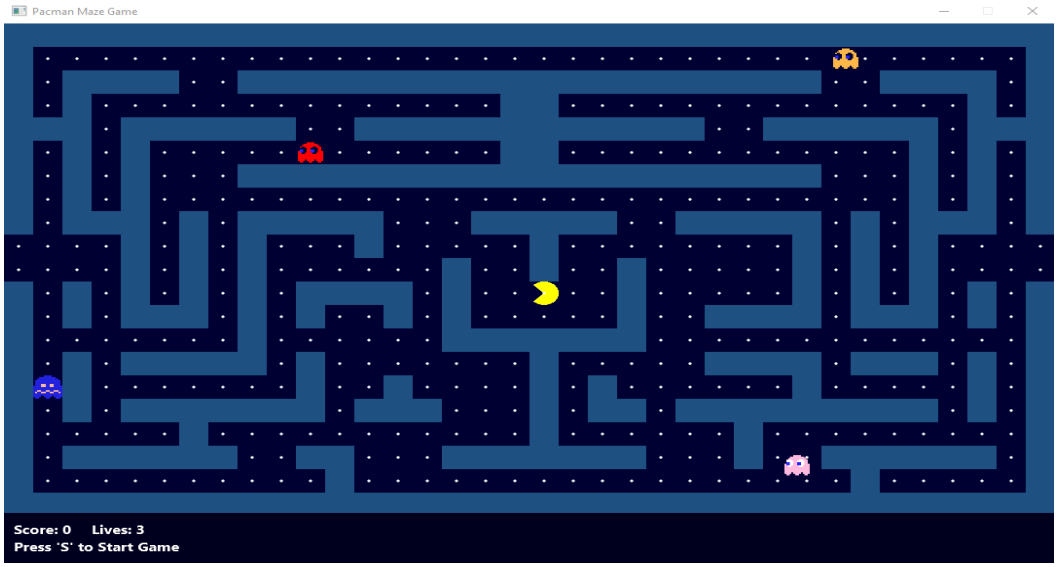
Ghost Movement: Ghosts move randomly, adding challenge to the game. Their positions are updated in the game loop.

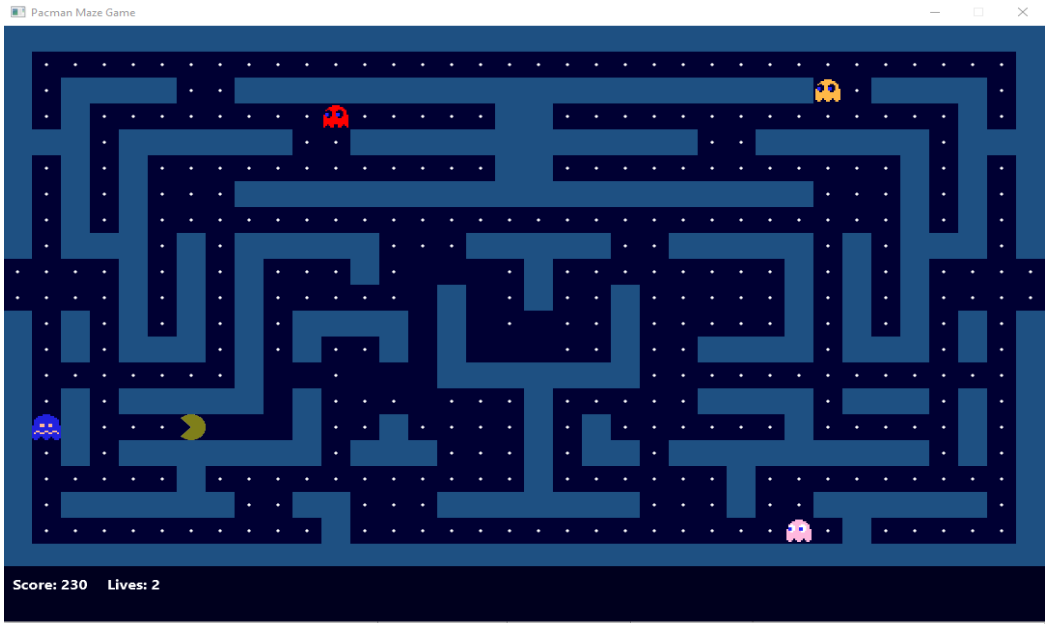
```
241  private void moveGhosts() {
242      ghostMoveCounter++;
243      if (ghostMoveCounter < 4) { // Adjust this value to control the speed
244          return;
245      }
246      ghostMoveCounter = 0;
247      for (int i = 0; i < ghosts.size(); i++) {
248          ImageView ghost = ghosts.get(i);
249          double newX = ghost.getX();
250          double newY = ghost.getY();
251          int direction = random.nextInt(4);
252
253          switch (direction) {
254              case 0: // Up
255                  newY -= CELL_SIZE;
256                  break;
257              case 1: // Down
258                  newY += CELL_SIZE;
259                  break;
260              case 2: // Left
261                  newX -= CELL_SIZE;
262                  break;
263              case 3: // Right
264                  newX += CELL_SIZE;
265                  break;
266          }
```

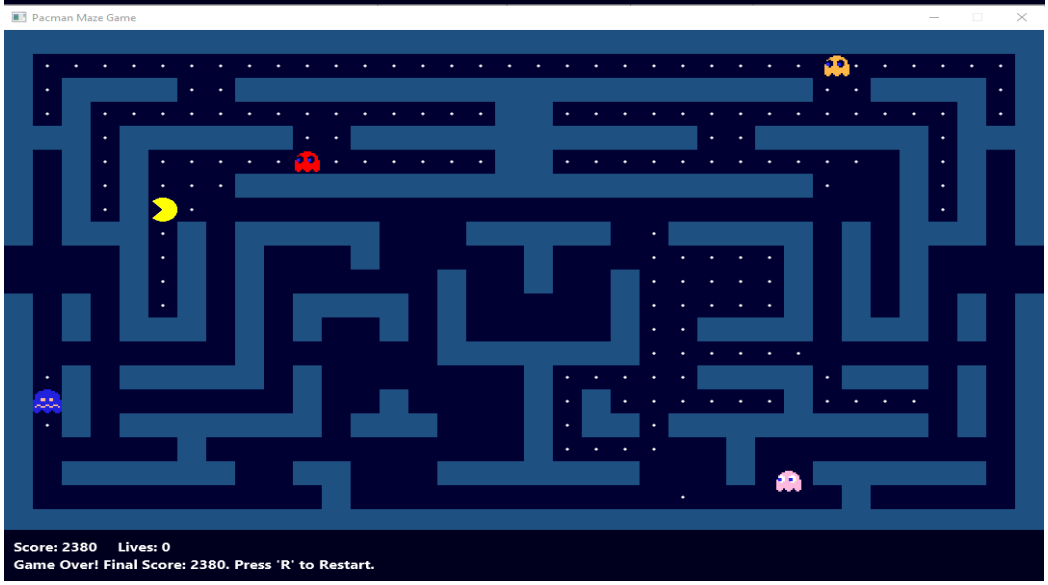
Restart Game: The restart function reinitializes the game state, resets Pacman's position, and repopulates the maze with dots and ghosts.

```
327   private void restartGame() {
328       score = 0;
329       lives = 3;
330       gameOver = false;
331       gameStarted = false;
332       scoreText.setText("Score: " + score);
333       livesText.setText("Lives: " + lives);
334       gameMessageText.setText("Press 'S' to Start Game");
335
336       int centerRow = ROWS / 2 + 1;
337       int centerCol = COLS / 2;
338       double centerX = centerCol * CELL_SIZE;
339       double centerY = centerRow * CELL_SIZE;
340
341       pacman.setX(centerX);
342       pacman.setY(centerY);
343
344       dots.clear(); //Removes collected dots
345       ghosts.clear(); // Resets ghost positions
346       initialGhostPositions.clear();//Clears stored initial positions
347
348       Pane gamePane = (Pane) pacman.getParent();
349       gamePane.getChildren().removeIf(node -> node instanceof ImageView && node != pacman);
350
351       createDots(gamePane);
352       createGhosts(gamePane);
353   }
```

OUTPUT:


Pacman Maze Game
Score: 0 Lives: 3
Press 'S' to Start Game


Pacman Maze Game
Score: 230 Lives: 2


Pacman Maze Game
Score: 2380 Lives: 0
Game Over! Final Score: 2380. Press 'R' to Restart.