TRƯỜNG ĐẠI HỌC ĐÀ LẠT

**KHOA CÔNG NGHỆ THÔNG TIN**



**TIỂU LUẬN**

**XÂY DỰNG TRÒ CHƠI CON RẮN**

Môn Lập trình Java

Giảng viên hướng dẫn: ThS. **Đoàn Minh Khuê**

Người thực hiện: Lê Viết Đăng Quang, 2111875

Nguyễn Nhật Linh, 2115229

**MỤC LỤC**

[I. MỞ ĐẦU 3](#_Toc154424158)

[1.1. Đặt vấn đề 3](#_Toc154424159)

[1.2. Mục tiêu 3](#_Toc154424160)

[1.3. Các chức năng dự kiến 3](#_Toc154424161)

[1.4. Đối tượng hướng đến 4](#_Toc154424162)

[1.5. Phương pháp thực hiện 5](#_Toc154424163)

[II. PHÂN TÍCH THIẾT KẾ 6](#_Toc154424164)

[2.1. Cửa sổ đăng nhập 6](#_Toc154424165)

[2.2. Màn hình chính 7](#_Toc154424166)

[2.3. Màn hình chơi 8](#_Toc154424167)

[2.4. Giao diện cửa hàng đổi thưởng 10](#_Toc154424168)

[III. QUÁ TRÌNH THỰC HIỆN VÀ MÃ NGUỒN 11](#_Toc154424169)

[3.1. Xây dựng dữ liệu 11](#_Toc154424170)

[3.2. Cài đặt và xây dựng giao diện bằng thư viện JavaFX 11](#_Toc154424171)

[3.3. Xây dựng logic trò chơi con rắn 17](#_Toc154424172)

[3.3.1. Xây dựng các class cơ bản của trò chơi 17](#_Toc154424173)

[3.3.2. Chế độ chơi cơ bản 20](#_Toc154424174)

[3.3.3. Các chế độ chơi khác 23](#_Toc154424175)

[IV. KIỂM THỬ VÀ KẾT QUẢ 27](#_Toc154424176)

[4.1. Kết quả đạt được 27](#_Toc154424177)

[4.2. Hạn chế 27](#_Toc154424178)

[V. KẾT LUẬN 29](#_Toc154424179)

[VI. TÀI LIỆU THAM KHẢO 30](#_Toc154424180)

[VII. PHÂN CÔNG CÔNG VIỆC 30](#_Toc154424181)

[VIII. LỜI CẢM ƠN 31](#_Toc154424182)

# I. MỞ ĐẦU

## 1.1. Đặt vấn đề

Trò chơi con rắn (Snake game) là một trong những tựa game điện tử được yêu thích nhất mọi thời đại. Xuất hiện lần đầu tiên trên các thiết bị Nokia vào những năm 1990, đến nay, trò chơi con rắn vẫn là lựa chọn hàng đầu cho những ai muốn giải trí vào những lúc rảnh rỗi. Sự đơn giản trong luật chơi, đồ họa pixel dễ thương và tính chất giải trí cao là những yếu tố quan trọng tạo nên sức hút cho Snake game. Đây sẽ là một dự án thích hợp để tự học ngôn ngữ mới, rèn luyện khả năng lập trình, giải quyết vấn đề và kỹ năng phát triển game, bên cạnh đó đây cũng là cơ hội để thử nghiệm, sáng tạo và cải tiến trò chơi Snake theo nhiều chế độ chơi khác nhau để tăng tính trải nghiệm của game cũng như trao dồi khả năng phân tích logic và kỹ năng lập trình.

## 1.2. Mục tiêu

Như đã đề cập, mục tiêu của đề tài là xây dựng một trò chơi con rắn đơn giản và dễ sử dụng, hoàn thiện các chức năng cơ bản của trò chơi, bên cạnh đó còn phát triển các chế độ chơi khác nhau nhầm tăng độ thú vị cho trò chơi bằng Java; tạo giao diện đồ họa thân thiện với người dùng bằng cách sử dụng thư viện JavaFX, giúp người dùng có trải nghiệm sử dụng thuận tiện; Xây dựng cửa hàng đổi quà, người chơi dùng điểm tích lũy để đổi các phần thưởng.

## 1.3. Các chức năng dự kiến

1. Chức năng cơ bản:

* Điều khiển rắn di chuyển trái/phải, lên/xuống trên màn hình.
* Tự động sinh mồi ngẫu nhiên để rắn ăn.
* Mỗi lần rắn ăn được mồi, thân rắn sẽ dài ra và điểm số tăng lên.
* Phát hiện va chạm giữa rắn và tường hoặc chính cơ thể rắn để kết thúc trò chơi

1. Các chế độ chơi:

* Chế độ chơi cơ bản: Người chơi chỉ việc điều khiển rắn ăn mồi nhiều nhất có thể.
* Chế độ vượt chướng ngại vật: Người chơi mỗi khi ăn mồi thì sẽ xuất hiện những bức tường ngẫu nhiên ở trên bản đồ, người chơi có nhiệm vụ điều khiển rắn né các chướng ngại và ăn mồi nhiều nhất có thể.
* Chế độ đua tốc độ: Người chơi sẽ có một khoảng thời gian nhất định, thời gian sẽ đếm ngược về không, nếu thời gian về không thì trò chơi sẽ kết thúc. Mỗi khi ăn mồi, thời gian sẽ được cộng thêm, Nhiệm vụ của người chơi là cố gắng ăn mồi để sinh tồn lâu nhất có thể.
* Chế độ kết hợp: Chế độ này sẽ kết hợp chế độ chướng ngại vật và đua tốc độ để tạo nên màn chơi kịch tính cho người chơi.

1. Cửa hàng đổi thưởng:

* Sử dựng điểm tích lũy để đổi phần thưởng.

1. Giao diện đồ họa:

* Thiết kế giao diện trực quan, thân thiện bằng JavaFX.
* Giao diện Menu chọn chế độ chơi, cửa hàng đổi thưởng.

## 1.4. Đối tượng hướng đến

* Học sinh, sinh viên: Đây là đối tượng yêu thích cá trò chơi giải trí để thư giản sau giờ học tập căng thẳng. Snake game có tính giải trí, vui nhộn phù hợp cho lứa tuổi này.
* Những người muốn học cách sử dụng thư viện JavaFX trong việc phát triển ứng dụng, trò chơi bằng ngôn ngữ Java.

## 1.5. Phương pháp thực hiện

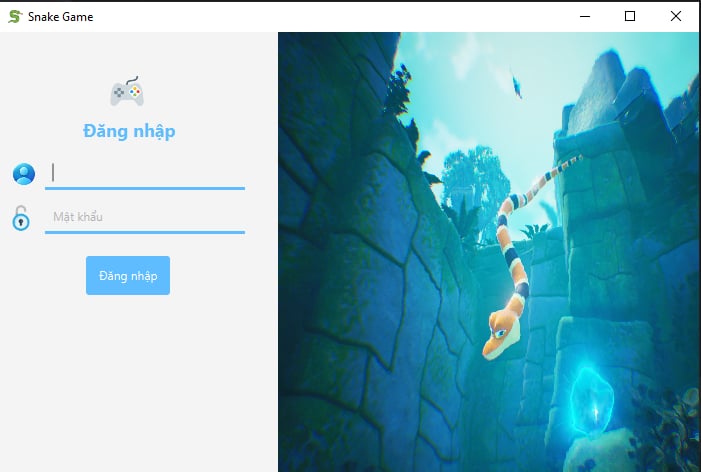
* Sử dụng JavaFx để nâng cấp đồ họa cho cửa sổ đăng nhập, menu game và giao diện của hàng đổi thưởng. Giao diện người dùng sẽ được thiết kế một cách thân thiện và dễ sử dụng, tạo trải nghiệm chơi game mượt mà và thú vị.
* Lập trình xử lý logic cơ bản cho rắn và môi trường chơi. Sau đó phát triển thêm các chế độ chơi như vượt chướng ngại vật, đua tốc độ để làm trò chơi trở nên thú vị hơn.
* Kết nối với cơ sở dữ liệu để lưu trữ thông tin về điểm số và mã đổi thưởng.

# II. PHÂN TÍCH THIẾT KẾ

Giao diện người dùng của trò chơi Snake được thiết kế để mang lại trải nghiệm người chơi mượt mà và thú vị.

## 2.1. Cửa sổ đăng nhập

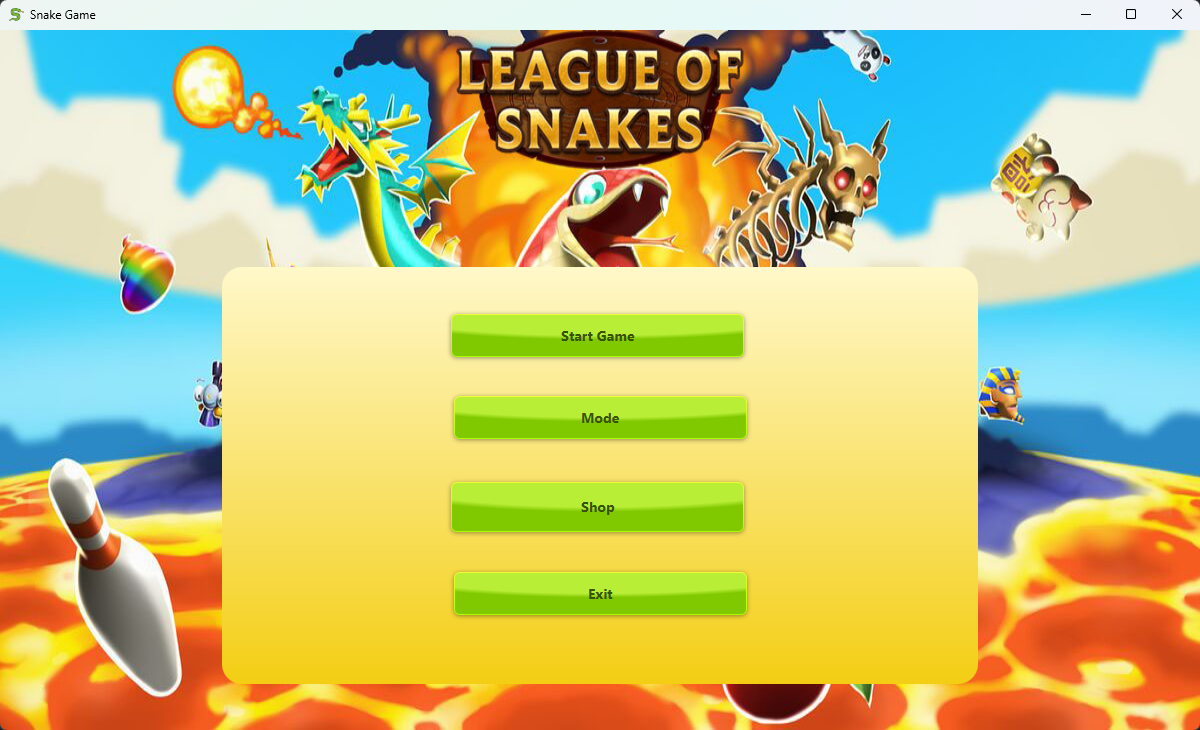
Người chơi phải đăng nhập để vào game, đăng nhập để người chơi lưu điểm số và để đổi mã đổi thưởng từ cửa hang. Giao diện này đơn giản và thân thiện, yêu cầu người chơi đăng nhập đúng tài khoản nhầm lưu trữ điểm số đã tích lũy.



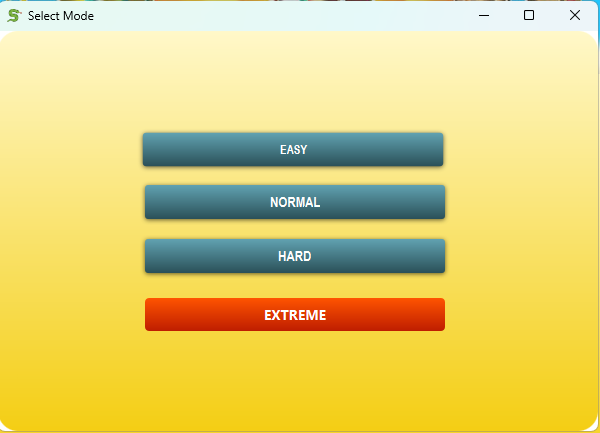
*Hình 2.1. Cửa sổ đăng nhập*

## 2.2. Màn hình chính

Tại đây, người chơi có thể chọn chế độ chơi như chế độ cơ bản, chế độ vượt chướng ngại vật, chế độ đua thời gian và chế độ khó. Bên cạnh đó còn có cửa hàng đổi thưởng. Các nút được kế dễ nhìn, dễ nhận diện, tạo điều kiện thuận lợi cho việc điều hướng



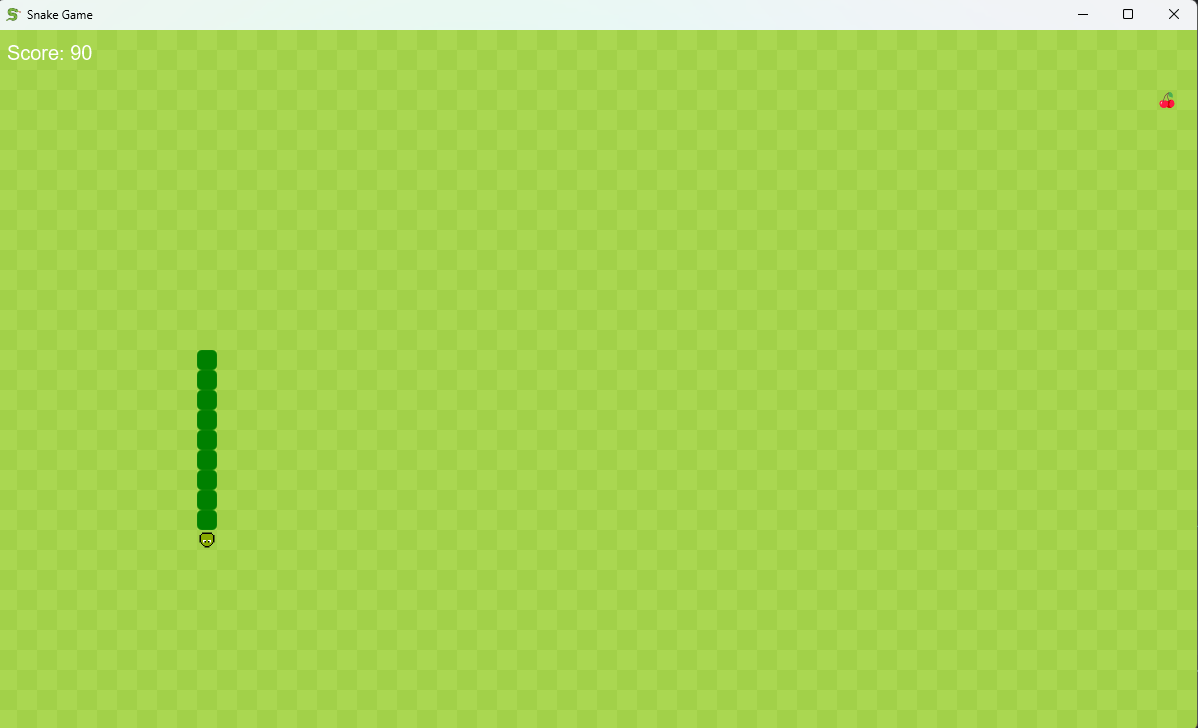
*Hình 2.2. Màn hình chính*

**

*Hình 2.3. Màn hình chọn chế độ chơi*

## 2.3. Màn hình chơi

Trong khi chơi, người chơi sẽ được đưa đến màn hình chơi chính. Giao diện này hiển thị con rắn, thức ăn và điểm số một cách rõ ràng. Các mục tiêu, như đạt được điểm số cao sẽ được hiển thị để khuyến kích người chơi.

**

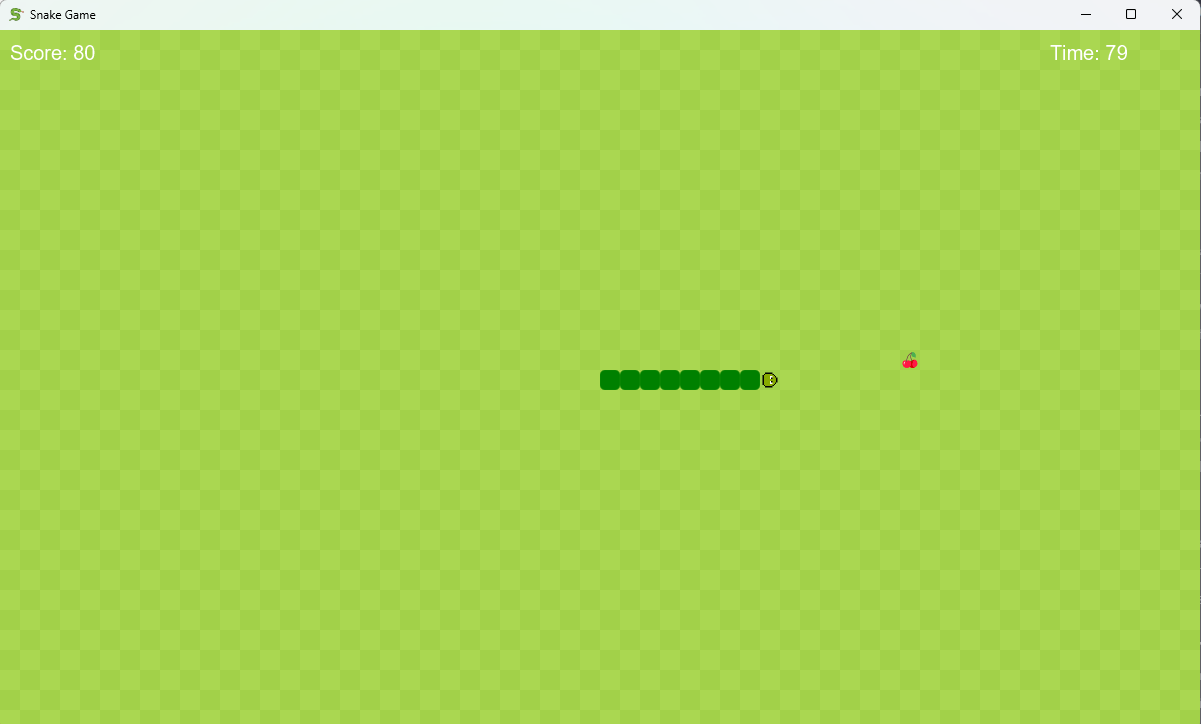
*Hình 2.4. Màn hình chơi chế độ cơ bản*

Chế độ chơi cơ bản: Người chơi sẽ điều khiển rắn ăn thật nhiều mồi và tránh va vào tường hoặc chính cơ thể của rắn. Mỗi lần ăn mồi thì điểm số sẽ tăng, thân rắn dài hơn và tốc độ di chuyển của rắn cũng sẽ tăng.

**

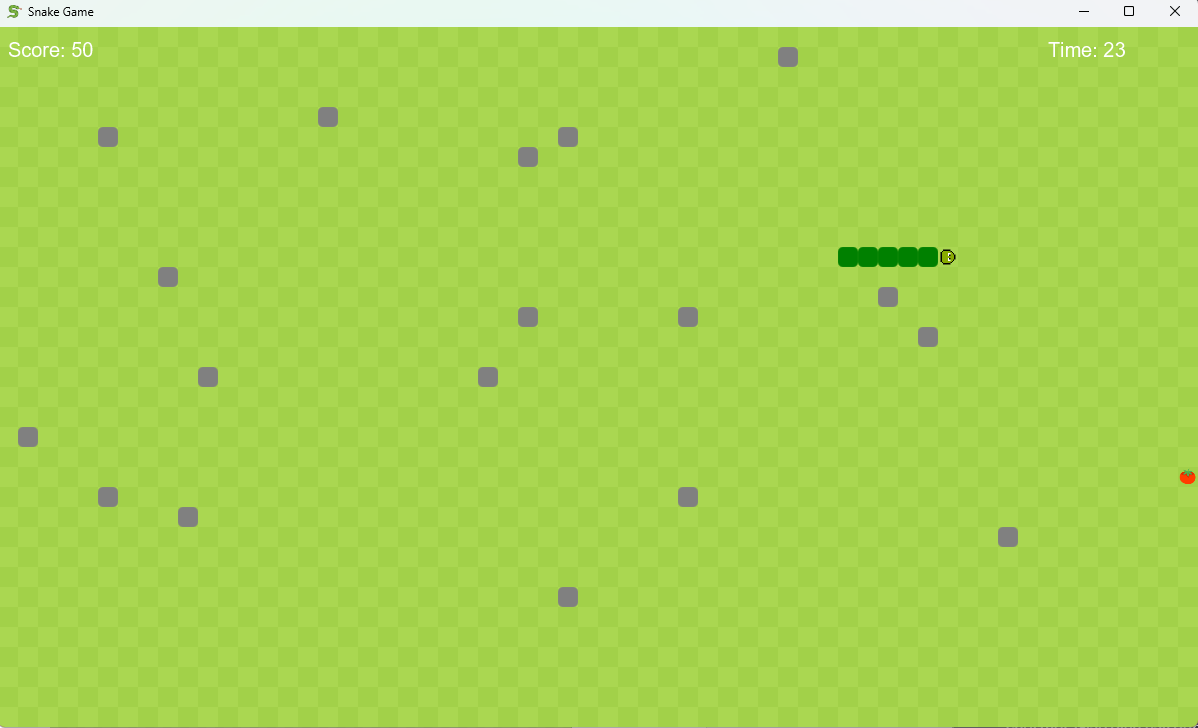
*Hình 2.5. Màn hình chơi chế độ vượt chướng ngại vật*

Chế độ vượt chướng ngại vật: Người chơi điều khiển rắn ăn mồi, và tránh những bức tường hiển thị ngẫu nhiên mỗi khi người chơi ăn mồi.

**

*Hình 2.6. Màn hình chơi chế độ đua thời gian*

Chế độ đua thời gian: Chế độ này sẽ có thời gian được quy định trước và đếm ngược đến 0 thì trò chơi sẽ kết thúc nên người chơi phải điều khiển rắn đến con đường ngắn nhất để ăn mồi, mỗi lần ăn mồi thì thời gian sẽ được tăng.

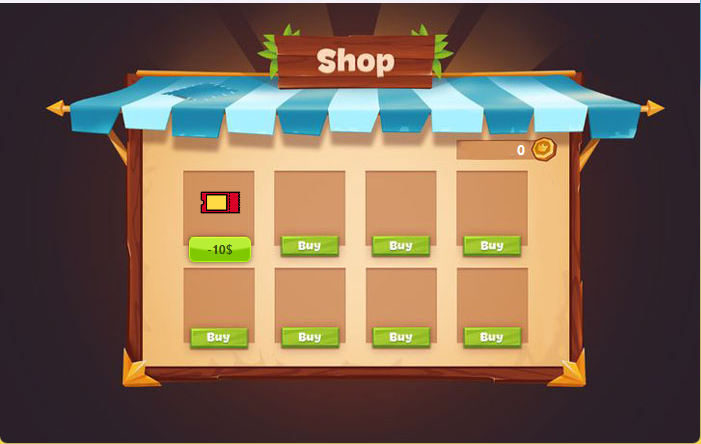
**

*Hình 2.7. Màn hình chế độ khó*

Chế độ khó: Đây là chế độ kết hợp giữa hai chế độ trên để tăng tính thú vị cho trò chơi, người chơi phải điều khiển rắn ăn mồi, né những bức tường xuất hiện bất ngờ và phải tìm con đường ngắn nhất để ăn mồi nhằm tăng thời gian.

## 2.4. Giao diện cửa hàng đổi thưởng

Tại đây, người chơi có thể đổi thưởng từ điểm tích lũy chơi để nhận mã giảm giá có trong cơ sở dữ liệu. Giao diện này được thiết kế cho dễ sử dụng và hấp dẫn với các biểu tượng.

**

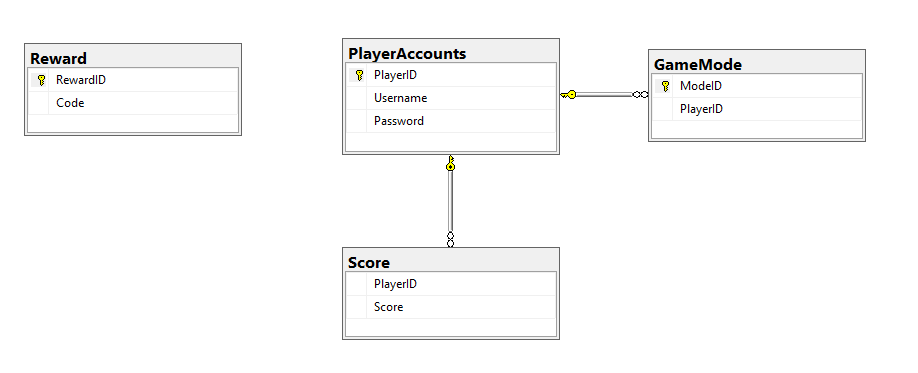
*Hình 2.4. Màn hình cửa hàng đổi thưởng*

# III. QUÁ TRÌNH THỰC HIỆN VÀ MÃ NGUỒN

## 3.1. Xây dựng dữ liệu

Cơ sở dữ liệu gồm 4 bảng:

* Bảng PlayerAccounts: Quản lý thông tin tài khoản
* Bảng GameMode: Quản lý chế độ
* Bảng Score: Lưu thông tin điểm số của tài khoản
* Bảng Reward: Lưu mã đổi thưởng



*Hình 3.1. Lược đồ minh họa dữ liệu*

## 3.2. Cài đặt và xây dựng giao diện bằng thư viện JavaFX

3.2.1. Cài đặt JavaFX

* Đầu tiên, cần cài đặt Java Development Kit (JDK) về máy tính. Truy cập website của Oracle để tải phiên bản JDK phiên bản mới nhất.
* Sau khi cài đặt JDK, bạn cần tải JavaFX SDK. Truy cập trang chủ JavaFX để tải bản JavaFX SDK mới nhất về máy.

3.2.2. Xây dựng giao diện và kết nối cơ sở dữ liệu

Trước tiên thiết kế giao diện SceneBuilder để tạo ra giao diện cần thiết như trang đăng nhập, trang menu, các nút chọn chế độ chơi, cửa hàng đổi thưởng, giao diện trang nhận thưởng.

Sau khi thiết kế giao diện từ SceneBuilder, ta chuyển sang viết các trang Controller như trang LoginController, ModeController, ShopController, RewardController để xử lý sự kiện như việc đăng nhập, chọn chế độ chơi cũng như là vào cửa hàng đổi thưởng để đổi lấy phần quà từ số điểm người chơi đã đạt được.

Đầu tiên với LoginController, khởi tạo chuỗi chứa dữ liệu từ khung nhập bên giao diện

String username = tfUsername.getText();

String password = tfPassword.getText();

Sau đó kết nối cơ sở dữ liệu kiểm tra dữ liệu xem tên tài khoản và mật khẩu có giống trong tài khoản không, nếu đúng thì chuyển sang giao diện menu để người chơi có thể bắt đầu chơi, còn không đúng thì sẽ hiện messbox thông báo người dùng nhập sai tên tài khoản hoặc mật khẩu.

SQLServerDataSource ds = new SQLServerDataSource();

ds.setServerName("DESKTOP-KSD365U");

ds.setDatabaseName("SnakeGame");

Mã nguồn bên dưới là để kết nối cơ sở dữ liệu kiểm tra xem tài khoản và mật khẩu đã được nhập đúng chưa.

try (Connection con = ds.getConnection())

{

Statement st = con.createStatement();

String sql = "select \* from PlayerAccounts where PlayerAccounts.Username = '" + username + "' and PlayerAccounts.Password = '"+ password +"'";

ResultSet rs = st.executeQuery(sql);

if (rs.next())

{ check = true }

con.close();

}

Nếu đúng thì biến check = true, sẽ đưa người chơi vào màn hình menu của trò chơi

if (check) {

final Node source = (Node) event.getSource();

final Stage stage = (Stage) source.getScene().getWindow();

stage.close();

FXMLLoader loader = new FXMLLoader(getClass().getResource("menu.fxml"));

Parent root = loader.load();

Scene scene = new Scene(root,1200,700);

Stage primaryStage = new Stage();

Image icon = new Image("img/icon\_snake.png");

primaryStage.getIcons().add(icon);

primaryStage.setScene(scene);

primaryStage.setTitle("Snake Game");

primaryStage.show();}

Nếu không thì sẽ hiện thông báo cho người dùng

else {

Alert alert = new Alert(Alert.AlertType.INFORMATION);

alert.setContentText("Tên đăng nhập hoặc mật khẩu không đúng");

alert.showAndWait();

}

Tiếp theo là xây dung menuController , để người chơi có thể chọn cấp độ cho trò chơi, sửa dụng switch case để xử lý sự kiện chọn của người chơi

switch (value) {

case 1:

ModeWall modeWall = new ModeWall();

GameMode = modeWall.initGame();

primaryStage = new Stage();

primaryStage.getIcons().add(icon);

primaryStage.setScene(GameMode);

primaryStage.setTitle("Snake Game");

primaryStage.show();

modeWall.startGameLoop();

break;

case 2:

ModeTimeAttack modeTimeAttack = new ModeTimeAttack();

GameMode = modeTimeAttack.initGame();

primaryStage = new Stage();

primaryStage.getIcons().add(icon);

primaryStage.setScene(GameMode);

primaryStage.setTitle("Snake Game");

primaryStage.show();

modeTimeAttack.startGameLoop();

break;

Về phần cửa hàng đổi thưởng, mã nguồn dưới sẽ truy xuất vào điểm số của tài khoản, mỗi lần đổi thưởng sẽ trừ điểm số để đổi lấy phần thưởng vô cùng giá trị,và mã đổi thưởng sau khi có 1 tài khoản đổi thưởng thì sẽ xóa khỏi cơ sở dữ liệu để tránh lỗi trùng lặp mã đổi thưởng

try(Connection con = ds.getConnection())

{

Statement st = con.createStatement();

String sql = "UPDATE Score SET Score = "+score+" FROM Score, PlayerAccounts WHERE Score.PlayerID = PlayerAccounts.PlayerID and PlayerAccounts.Username = '"+LoginController.globalUsername+"'";

if(score >=0)

{

int rs = st.executeUpdate(sql);

if(rs != 0)

{

lbScore.setText(String.valueOf(score));

}

} else {

Alert alert = new Alert(Alert.AlertType.INFORMATION, "Bạn không đủ điểm");

alert.showAndWait();

}

try(Connection con = ds.getConnection())

{

Statement st = con.createStatement();

String sql = "delete Reward where Reward.RewardID = "+ShopController.id+" ";

int rs = st.executeUpdate(sql);

con.close();

} catch (SQLServerException e) {

throw new RuntimeException(e);

} catch (SQLException e) {

throw new RuntimeException(e);

}

## 3.3. Xây dựng logic trò chơi con rắn

### 3.3.1. Xây dựng các class cơ bản của trò chơi

Đầu tiên xây dựng class Snake: trong Snake chứa code gồm hiển thị rắn lên màn hình, sử dựng bốn hình ảnh đầu rắn để giúp người chơi nhìn thấy hướng rắn đang di chuyển,

   public Snake() {

body = new LinkedList<>();

headUpImage = new Image("/img/snake/head\_up\_image.png");

headDownImage = new Image("/img/snake/head\_down\_image.png");

headLeftImage = new Image("/img/snake/head\_left\_image.png");

headRightImage = new Image("/img/snake/head\_right\_image.png");

body.add(new Point(5, 5, Color.GREEN));

currentDirection = RIGHT;

growNextMove = false;

originalColor = Color.GREEN;

}

Dưới đây là mã nguồn được xây dựng cho việc di chuyển của rắn

public void move() {

Point head = body.getFirst();

Point newHead = new Point(head.getX(), head.getY(), head.getColor());

switch (currentDirection) {

case RIGHT:

newHead.moveRight();

break;

case LEFT:

newHead.moveLeft();

break;

case UP:

newHead.moveUp();

break;

case DOWN:

newHead.moveDown();

break;

}

body.addFirst(newHead);

if (!growNextMove) {

body.removeLast();

} else {

growNextMove = false;

}

changeColor(originalColor);

}

Và việc kiểm tra xem rắn có đâm vào tường hay đâm vào chính cơ thể rắn hay không.

public boolean checkSelfCollision() {

Point head = body.getFirst();

for (int i = 1; i < body.size(); i++) {

Point segment = body.get(i);

if (head.getX() == segment.getX() && head.getY() == segment.getY()) {

return true;

}

}

return false;

}

public boolean checkWallCollision(int maxX, int maxY) {

Point head = body.getFirst();

int headX = (int) head.getX();

int headY = (int) head.getY();

return headX < 0 || headY < 0 || headX >= maxX || headY >= maxY;

}

Tiếp theo xây dựng class food: Để mồi có thể xuất hiện random thì cần phải thêm thư viện random của java:

import java.util.Random;

Khởi tạo một list hình ảnh trái cây để vừa lấy random hình ảnh trái cây:

private static final String[] FRUIT\_IMAGE\_PATHS = {

"/img/food/ic\_apple.png",

"/img/food/ic\_tomato.png",

"/img/food/ic\_orange.png",

"/img/food/ic\_cherry.png" };

Mã nguồn dưới viết về việc xuất hiện random của đồ ăn và vòng for cho việc lấy hình ảnh trái cây ra màn hình chơi

public Food(int maxX, int maxY) {

fruitImages = new Image[FRUIT\_IMAGE\_PATHS.length];

for (int i = 0; i < FRUIT\_IMAGE\_PATHS.length; i++) {

fruitImages[i] = new Image(FRUIT\_IMAGE\_PATHS[i]);

}

randomizePosition(maxX, maxY);

}

public void randomizePosition(int maxX, int maxY) {

Random random = new Random();

this.x = random.nextInt(maxX);

this.y = random.nextInt(maxY);

}

Sau khi xây dựng class food, tiếp đến sẽ xây dựng Background cho trò chơi

public Background(int width, int height, int squareSize, Color color1, Color color2) {

this.width = width;

this.height = height;

this.squareSize = squareSize;

this.color1 = color1;

this.color2 = color2;

}

public void draw(GraphicsContext gc) {

for (int i = 0; i < width / squareSize; i++) {

for (int j = 0; j < height / squareSize; j++) {

gc.setFill((i + j) % 2 == 0 ? color1 : color2);

gc.fillRect(i \* squareSize, j \* squareSize, squareSize, squareSize);

}

}

}

### 3.3.2. Chế độ chơi cơ bản

public Scene initGame() {

snake = new Snake(); // khởi tạo rắn

score = new Score(); // khởi tạo điểm số

snakeSpeed = 100.0;// khởi tạo tốc độ rắn ban đầu

canvas = new Canvas(WIDTH, HEIGHT);

gc = canvas.getGraphicsContext2D();

food = new Food(20, 20); // Khơi tạo mồi

food.randomizePosition(WIDTH / SQUARE\_SIZE, HEIGHT / SQUARE\_SIZE);

foodImage = food.getImage(); // lấy hình ảnh mồi

background = new Background(WIDTH, HEIGHT, SQUARE\_SIZE, Color.web("AAD751"), Color.web("A2D149")); // Khởi tạo bg

gameLoop = new Timeline(new KeyFrame(Duration.millis(100), e -> gameUpdate()));

gameLoop.setCycleCount(Animation.INDEFINITE);

Scene gameScene = new Scene(new BorderPane(canvas));

gameScene.setOnKeyPressed(event -> handleKeyPress(event.getCode()));

return gameScene;

}

Hàm cập nhật game, nhằm kiểm tra xem rắn có ăn dính mồi không hay va dính tường hoặc va dính bản thân

private void gameUpdate() {

snake.move();

if (snake.getBody().getFirst().getX() == food.getX() && snake.getBody().getFirst().getY() == food.getY()) {

snake.grow();

food.randomizePosition(WIDTH / SQUARE\_SIZE, HEIGHT / SQUARE\_SIZE);

foodImage = food.getImage();

score.increaseScore(10);

updateSnakeSpeed();

}

if (snake.checkWallCollision(WIDTH / SQUARE\_SIZE, HEIGHT / SQUARE\_SIZE)) {

gameLoop.stop();

displayGameOver();

return;

}

if (snake.checkSelfCollision()) {

gameLoop.stop();

displayGameOver();

} else {

snake.updateHeadImage();

renderGame();

}

}

private void renderGame() {

gc.clearRect(0, 0, WIDTH, HEIGHT);

background.draw(gc);

snake.draw(gc, SQUARE\_SIZE);

drawScore();

gc.drawImage(foodImage, food.getX() \* SQUARE\_SIZE, food.getY() \* SQUARE\_SIZE, SQUARE\_SIZE, SQUARE\_SIZE);

}

Mã nguồn bên dưới viết về khi trò chơi kết thúc, vẽ điểm số ở góc bên trái của trò chơi và tăng tốc độ cho rắn

private void displayGameOver() {

gc.clearRect(0, 0, WIDTH, HEIGHT);

gc.setFill(Color.RED);

gc.setFont(new Font("Arial", 50));

gc.fillText("Game Over", WIDTH / 2 - 100, HEIGHT / 2);

}

private void drawScore() {

gc.setFill(Color.WHITE);

gc.setFont(new Font("Arial", 20));

gc.fillText("Score: " + score.getScore(), 10, 30);

}

private void updateSnakeSpeed() {

if (scoreIncreaseCount > 0 && scoreIncreaseCount % SPEED\_INCREASE\_THRESHOLD == 0) {

snakeSpeed += 10.0;

gameLoop.setRate(snakeSpeed / 100.0);

}

}

### 3.3.3. Các chế độ chơi khác

Trước tiên là chế độ chươi vượt chướng ngại vật: Để các bức tường xuất hiện một cách random thì cũng sẽ sử dụng như thư viện đã đề cập ở trước đó ở class Food

public void generateRandomWalls() {

Random random = new Random();

int numWalls = random.nextInt(1) + 3;// Mỗi khi ăn 1 mồi sẽ tạo ra 3 bức tường

for (int i = 0; i < numWalls; i++) {

int wallX = random.nextInt(maxX);

int wallY = random.nextInt(maxY);

Point wall = new Point(wallX, wallY, Color.GRAY);

walls.add(wall);}}

Bên cạnh đó phải có hàm kiểm tra để không gây ra lỗi mồi và tường hiện random cùng một chỗ

public boolean checkCollision(Food food) {

for (Point wall : walls) {

if (wall.getX() == food.getX() && wall.getY() == food.getY()) {

return true;

}

}

return false;

}

Sau đó để có thể chơi chế độ này thì sẽ phải khởi tạo và bổ sung thêm vào hàm update

private void gameUpdate() {

snake.move();

if (snake.getBody().getFirst().getX() == food.getX() && snake.getBody().getFirst().getY() == food.getY()) {

while (wall.checkCollision(food)) {

food.randomizePosition(WIDTH / SQUARE\_SIZE, HEIGHT / SQUARE\_SIZE);

}

snake.grow();

food.randomizePosition(WIDTH / SQUARE\_SIZE, HEIGHT / SQUARE\_SIZE);

foodImage = food.getImage();

score.increaseScore(10);

if (score.getScore() % 10 == 0 && score.getScore() < 200 ){

wall.generateRandomWalls();

}

updateSnakeSpeed();

}

}

Trong đoạn mã nguồn trên đã đặt một lệnh if để khi người chơi đạt 200 điểm thì sẽ không tạo thêm tường ngẫu nhiên nữa nhằm giới hạn độ khó cho trò chơi.

Tiếp theo là xây dựng chế độ đua thời gian, đầu tiên khởi tạo khoảng thời gian để đếm ngược:

private static final int MAX\_TIME = 120;

public void updateTime(boolean snakeAteFood) {

if (timeRemaining > 0) {

timeRemaining--;

if (snakeAteFood) {

//Nếu rắn ăn thành công, thêm 30 giây vào thời gian còn lại

timeRemaining += 30;

if (timeRemaining > MAX\_TIME) {

timeRemaining = MAX\_TIME;

}

}

} else {

// Time's up

return;

}

}

Sau đó cập nhật chế độ này vào trò chơi bằng việc khởi tạo class trên và cập nhật lại hàm update

private void gameUpdate() {

snake.move();

if (snake.getBody().getFirst().getX() == food.getX() && snake.getBody().getFirst().getY() == food.getY()) {

snake.grow();

food.randomizePosition(WIDTH / SQUARE\_SIZE, HEIGHT / SQUARE\_SIZE);

foodImage = food.getImage();

score.increaseScore(10);

timeAttackMode.updateHighScore(snake.getBody().size());

// + thêm thời gian khi rắn ăn thành công

timeAttackMode.updateTime(true);

updateSnakeSpeed();

}

if (timeAttackMode.getTimeRemaining() == 0) {

gameLoop.stop();

displayGameOver();

return;

}

// Thời gian đếm ngược

timeAttackMode.updateTime();

Để tạo ra mã nguồn cho chế độ khó nhất trong trò chơi bằng cách kết hợp hai mã nguồn trên vào hàm update sẽ tạo ra chế độ chơi vừa có chướng ngại vật vừa có thời gian đếm ngược.

# IV. KIỂM THỬ VÀ KẾT QUẢ

## 4.1. Kết quả đạt được

Trong quá trình phát triển trò chơi Snake Game sử dụng Java và JavaFX, dự án đã đạt được một số kết quả quan trọng sau:

* Giao diện menu trò chơi đơn giản, thân thiện: Giao diện người dùng được thiết kế đơn giản, thân thiện, dễ tiếp cận với hầu hết người dùng.
* Hoàn thiện cơ bản các chức năng chính: Đảm bảo logic của trò chơi, kiểm tra đầy đủ các chức năng cơ bản như di chuyển rắn, ăn thức ăn, kết thúc trò chơi khi rắn chạm vào tường hoặc tự chạm vào cơ thể mình.
* Hoàn thiện các chế độ chơi: Đảm bảo game chạy mượt mà không gặp vấn đề về các logic của các chế độ chơi, hoàn thiện hai chế độ chơi đã đề ra.
* Liên kết với cơ sở dữ liệu: Trò chơi kết nối với cơ sở dữ liệu SqlServer để lưu trữ và truy xuất thông tin về tài khoản và các mã đổi thưởng. Điều này giúp bảo vệ và duy trì dữ liệu một cách hiệu quả.
* Thử nghiệm và xác minh: Thực hiện kiểm thử trò chơi để đảm bảo tính ổn định và hiệu quả. Trò chơi đã được kiểm tra trước khi triển khai để đảm bảo nó hoạt động theo đúng dự kiến.

## 4.2. Hạn chế

Mặc dù đã đạt được nhiều kết quả tích cực, trò chơi Snake Game sử dụng Java và JavaFX cũng có một số hạn chế:

* Giao diện hạn chế: JavaFX, mặc dù mạnh mẽ, nhưng có thể đối mặt với thách thức trong việc xử lý đồ họa đặc biệt là khi game có nhiều đối tượng di động hoặc đối tượng phức tạp.
* Tương thích nền tảng: Có thể xuất hiện vấn đề tương thích khi chạy game trên các nền tảng khác nhau, đặc biệt là do Java không luôn đảm bảo tương thích hoàn hảo trên mọi hệ điều hành.
* Khả năng mở rộng: Trò chơi hiện tại chỉ tập trung vào việc phát triển chế độ trò chơi. Để phát triển trò chơi hoàn chỉnh, có thể cần phải chỉnh sửa lại bảng cơ sở dữ liệu, sử dụng api để đổi mã thưởng thay cho Sql Server, nâng cấp giao diện trò chơi con rắn, tích hợp thêm những tính năng mới như kết hợp âm thanh để trò chơi có thể sinh động hơn. Chỉnh sửa việc lưu trữ điểm tùy theo các chế độ chơi và thêm nhiều phần quà sinh động hơn như đổi thưởng lấy trang phục cho rắn hay bản đồ mới … Phát triển sang nền tảng di động để mở rộng đối tượng người chơi và tận dụng tiềm năng thị trường di động.
* Hiệu suất khi rắn có chiều dài lớn: Trong trường hợp rắn có chiều dài rất lớn, trò chơi có thể trở nên chậm chạp và tốn tài nguyên hệ thống nếu không được tối ưu hóa đúng cách.

# V. KẾT LUẬN

Trong dự án này, trò chơi Snake Game sử dụng Java và JavaFX. Trò chơi này cho phép người chơi có thể giải trí sau những giờ làm việc, học tập mệt mỏi thông qua giao diện người dùng đơn giản và kết nối cơ sở dữ liệu SQL server.

Dự án đã đạt được một số kết quả quan trọng như sau:

* Xây dựng giao diện người dùng trực quan và dễ sử dụng cho việc giải trí cũng như đổi thưởng.
* Kết nối với cơ sở dữ liệu SQL server để lưu trữ và truy xuất thông tin.
* Xây dựng trò chơi cơ bản, bên cạnh đó phát triển các chế độ chơi thú vị.
* Kiểm tra trò chơi để đảm bảo tính ổn định và hiệu quả.

Tuy nhiên, dự án cũng có một số hạn chế:

* Giao diện người dùng có thể hạn chế trong việc tạo các giao diện phức tạp và đẹp mắt.
* Cần cải thiện về tính bảo mật, cải thiện các phần quà được nhận trong trò chơi để mang tính thú vị cho trò chơi.
* Yêu cầu tối ưu hóa để cải thiện hiệu suất, đặc biệt khi phát triển trò chơi nhiều chế độ hơn.

Như vậy, mặc dù còn nhiều hạn chế nhưng dự án đã cung cấp một cơ sở để phát triển trò chơi con rắn cơ bản. Dự án trò chơi có tiềm năng để tiếp tục phát triển và mở rộng để đáp ứng nhu cầu của ngành giải trí. Cá nhân sinh viên đã học được rất nhiều trong quá trình phát triển dự án này, từ quá trình thiết kế lên ý tưởng, xây dựng logic trò chơi đến kỹ năng làm việc nhóm, và hy vọng rằng nó có thể là một khởi đầu cho các trò chơi phức tạp hơn trong tương lai sử dụng JavaFX và ngôn ngữ Java.

# VI. TÀI LIỆU THAM KHẢO

6.1. Sách

- Schildt, H. (2021). Java: The Complete Reference. McGraw-Hill Education.

6.2. Trang web

- Oracle - JavaFX Documentation: https://docs.oracle.com/javafx/2/api/javafx/scene/doc-files/introduction\_to\_javafx.html

- Udemy - JavaFX for GUI Development: https://www.udemy.com/course/javafx-gui-development/

- LibGDX - Java Game Development Framework: https://libgdx.com/

- gameprogrammingpatterns.com – Basic Game Programming Patterns: <https://gameprogrammingpatterns.com/>

# VII. PHÂN CÔNG CÔNG VIỆC

|  |  |
| --- | --- |
| Tên thành viên nhóm | Công việc |
| 2111875 Lê Viết Đăng Quang | Lên ý tưởng, code logic game, nối cơ sở dữ liệu |
| 2115229 Nguyễn Nhật Linh | Lên ý tưởng, code giao diện menu, đăng nhập, shop, kết nối cơ sở dữ liệu |

# VIII. LỜI CẢM ƠN

Trong quá trình hoàn thành dự án này, nhóm chúng em muốn bày tỏ lòng biết ơn sâu sắc đến giáo viên hướng dẫn là thầy Đoàn Minh Khuê và các bạn sinh viên tham gia lớp học môn lập trình Java của Thầy - những người và tổ chức đã hỗ trợ, đóng góp và tạo điều kiện cho dự án này trở thành hiện thực. Em cảm ơn thầy Đoàn Minh Khuê vì sự hướng dẫn, kiến thức, sự hỗ trợ liên tục trong suốt quá trình nghiên cứu và các góp ý, ý kiến khuyến nghị của thầy đã đóng góp quan trọng vào quá trình xây dựng của dự án này. Bên cạnh đó, nhóm em muốn bày tỏ cảm ơn đến bạn bè đã đóng góp thời gian và kiến thức của họ để hỗ trợ dự án này. Các cuộc thảo luận và phản hồi từ cô và các bạn đã giúp nhóm chúng em nâng cao chất lượng nghiên cứu.

Cuối cùng, nhóm chúng em xin gửi lời cảm ơn đến tất cả những ai đã đóng góp vào dự án này. Công việc này không thể thực hiện được mà không có sự hỗ trợ và đóng góp của tất cả mọi người.

Trân trọng,

Lê Viết Đăng Quang, Nguyễn Nhật Linh