Yusuf Taha Öncü

Introduction to Drive And Survive

This project is a 2D car game called “Drive and Survive,” developed using Java’s GUI features. The objective of the game is to avoid colliding with incoming cars while maintaining a steady speed. The game gets progressively harder as the score increases by adding more obstacles and increasing the speed of the cars.

Technologies Used:

- Java programming language
- Swing for GUI components
- Intellij IDEA compiler
- Timer for game loop and speed control
- Random class for dynamic elements (car movement, obstacles)

Development Process

1. Game Setup:

The game screen is built with a road and obstacles that move down the screen. The player’s car is controlled with the left and right arrow keys, and the goal is to avoid collisions with other vehicles.

2. Graphics and Animation:

• The game uses images for the background, cars, and obstacles, which are continuously updated as they move downward.

• The game speed increases as the player progresses.

3. Collision Detection:

A key challenge in this project was detecting when the player’s car collides with other moving cars. This was accomplished by checking if the bounding boxes of the player’s car and other cars overlap.

4. Main Menu and Game Over Screen:

• The game starts with a main menu where the player can begin the game by pressing “Enter.”

• The game over screen appears when a collision occurs, displaying the final score and allowing the player to restart the game by pressing “Enter.”

Challenges and Solutions

1. Car Movement:

The car’s movement along the X-axis is controlled by the left and right arrow keys, but there were issues with keeping the car within the bounds of the screen. This was solved by adding checks to prevent the car from going off the screen.

2. Collision Detection:

The most significant challenge was ensuring that collisions were detected accurately. The method used checks the positions of the player’s car and the incoming cars to determine if they overlap. Fine-tuning this process was essential to ensure realistic gameplay.

3. Game Speed:

The game’s difficulty is tied to the score and the speed of obstacles. The speed increases every 50 points, but it had to be limited to prevent the game from becoming too fast to play.

Conclusion

Objectives Met: The main goal was to create a fun, challenging, and visually engaging game, which was successfully achieved. The gameplay is simple but engaging due to the increasing speed and random placement of obstacles.

Technical And Code Parts

1. Displaying the Main Menu:

• This part shows the main menu before the game starts, displaying “Get Ready!” and instructions to “Press Enter to Start.”

Code section:

FUNCTION displayMainMenu(graphics)

SET mainMenuBackground TO "menu.png"

DRAW mainMenuBackground ON SCREEN

SET textColor TO WHITE

SET font TO Arial, Italic, Size 50

DRAW TEXT "Get Ready!" AT (220, 250)

SET font TO Arial, Italic, Size 30

DRAW TEXT "Press Enter to Start" AT (220, 300)

END FUNCTION

```
private void displayMainMenu(Graphics g) {
    mainMenuBackground = new ImageIcon("assets/menu.png");
    mainMenuBackground.paintIcon(this, g, 0, 0);

    g.setColor(Color.WHITE);
    g.setFont(new Font("Arial", Font.ITALIC, 50));
    g.drawString("Get Ready!", 220, 250);
    g.setFont(new Font("Arial", Font.ITALIC, 30));
    g.drawString("Press Enter to Start", 220, 300);
}
```

2. Drawing the Road and Graphics:

• This section defines how the road, trees, and the car appear on the screen. Trees move on the road, and the car’s position is updated as it moves along.

Code section:

FUNCTION playGame(graphics)

SET backgroundColor TO GREEN

FILL RECTANGLE (0, 0, 700, 1000) WITH backgroundColor

SET roadColor TO GRAY

DRAW LEFT EDGE OF ROAD AT (90, 0, 10, 1000)

DRAW RIGHT EDGE OF ROAD AT (600, 0, 10, 1000)

SET mainRoadColor TO BLACK

DRAW MAIN ROAD AT (100, 0, 500, 1000)

LOAD TREE IMAGE "tree1.png"

DRAW TREE IMAGE AT (0, tree1ypos)

INCREASE tree1ypos BY 50

END FUNCTION

```
private void playGame(Graphics g) {
    g.setColor(new Color(34, 139, 34));  // Green background
    g.fillRect(0, 0, 700, 1000);  // Background
    g.setColor(Color.gray);
    g.fillRect(90, 0, 10, 1000);  // Left edge of the road
    g.fillRect(600, 0, 10, 1000);  // Right edge of the road
    g.setColor(Color.black);
    g.fillRect(100, 0, 500, 1000);  // The main black road

    // Moving trees on the road
    tree1 = new ImageIcon("./assets/tree1.png");
    tree1.paintIcon(this, g, 0, tree1ypos);
    tree1ypos += 50;  // Move the trees
}
```

3. Car Movement and Collisions:

• The car moves left and right. Arrow keys control the car, and collisions with trees or other cars are checked.

Code section:

IF LEFT_KEY_PRESSED AND NOT gameover AND NOT inMainMenu THEN

DECREASE xpos BY 100

IF xpos < 100 THEN

SET xpos TO 100

END IF

END IF

IF RIGHT_KEY_PRESSED AND NOT gameover AND NOT inMainMenu THEN

INCREASE xpos BY 100

IF xpos > 500 THEN

SET xpos TO 500

END IF

END IF

// Collision Detection

IF car1ypos < carCurrentYPosition AND car1ypos + 175 > carCurrentYPosition AND car1xpos[cxpos1] == carCurrentXPosition THEN

SET gameover TO TRUE

END IF

```
if (e.getKeyCode() == KeyEvent.VK_LEFT && !gameover && !inMainMenu) {
    xpos -= 100;
    if (xpos < 100) {
        xpos = 100;
    }
}
if (e.getKeyCode() == KeyEvent.VK_RIGHT && !gameover && !inMainMenu) {
    xpos += 100;
    if (xpos > 500) {
        xpos = 500;
    }
}

// Collision detection
if (y1pos < ypos && y1pos + 175 > ypos && carxpos[cxpos1] == xpos) {
    gameover = true;
}
```

4. Score and Speed:

• The score increases over time, and the speed gradually increases. This ensures that the game gets more difficult as time passes.

Code section:

DRAW SCORE_BOX AT (120, 35, 220, 50) WITH COLOR GRAY

DRAW SPEED_BOX AT (125, 40, 210, 40) WITH COLOR DARK_GRAY

SET textColor TO WHITE

SET font TO Arial, Bold, Size 30

DISPLAY "Score : " + score AT (130, 67)

DISPLAY speed + " Km/h" AT (400, 67)

INCREASE score BY 1

INCREASE speed BY 1

```
g.setColor(Color.gray);
g.fillRect(120, 35, 220, 50);  // Score box
g.setColor(Color.DARK_GRAY);
g.fillRect(125, 40, 210, 40);  // Speed box
g.setColor(Color.white);
g.setFont(new Font("Arial", Font.BOLD, 30));
g.drawString("Score : " + score, 130, 67);
g.drawString(speed + " Km/h", 400, 67);

score++;  // Increase score
speed++;  // Increase speed
```

5. Game Over Screen:

• When the game ends, a “Game Over!” message appears, and the player is instructed to press Enter to restart the game.

Code section:

IF gameover THEN

DRAW GAME_OVER_BOX AT (120, 210, v460, 200) WITH COLOR GRAY

DRAW BACKGROUND_BOX AT (130, 220, 440, 180) WITH COLOR DARK_GRAY

SET font TO Serif, Bold, Size 50

SET textColor TO YELLOW

DISPLAY "Game Over!" AT (210, 270)

SET font TO Arial, Bold, Size 30

SET textColor TO WHITE

DISPLAY "Press Enter to Restart" AT (190, 340)

END IF

```
if (gameover) {
    g.setColor(Color.gray);
    g.fillRect(120, 210, 460, 200);  // Game over box
    g.setColor(Color.DARK_GRAY);
    g.fillRect(130, 220, 440, 180);  // Game over background
    g.setFont(new Font("Serif", Font.BOLD, 50));
    g.setColor(Color.yellow);
    g.drawString("Game Over !", 210, 270);
    g.setColor(Color.white);
    g.setFont(new Font("Arial", Font.BOLD, 30));
    g.drawString("Press Enter to Restart", 190, 340);
}
```

6. Playing Background Music:

• The game plays background music when it starts and keeps playing during the game. Also we created separate class just for playing background music just for apply inheritance rule.

Code section:

FUNCTION playBackgroundMusic()

TRY

LOAD soundFile FROM "background.wav"

INITIALIZE audioStream WITH soundFile

SET backgroundMusic TO LOOP CONTINUOUSLY

CATCH EXCEPTION

PRINT ERROR

END TRY

END FUNCTION

```
public static void playBackgroundMusic() {
    try {
        File soundFile = new File("assets/background.wav");
        AudioInputStream audioStream = AudioSystem.getAudioInputStream(soundFil
        Clip backgroundMusic = AudioSystem.getClip();
        backgroundMusic.open(audioStream);
        backgroundMusic.loop(Clip.LOOP_CONTINUOUSLY);
    } catch (Exception e) {
        e.printStackTrace();
    }
}
```