GURU NANAK COLLEGE BUDHLADA

DEPARTMENT: COMPUTER

NAME OF PROJECT: Snake Game

Submitted to:

HOD

Dr. Rekha Kalra

Submitted by:

Arvaaz Khan (321853)

Harkirat Singh(321845)

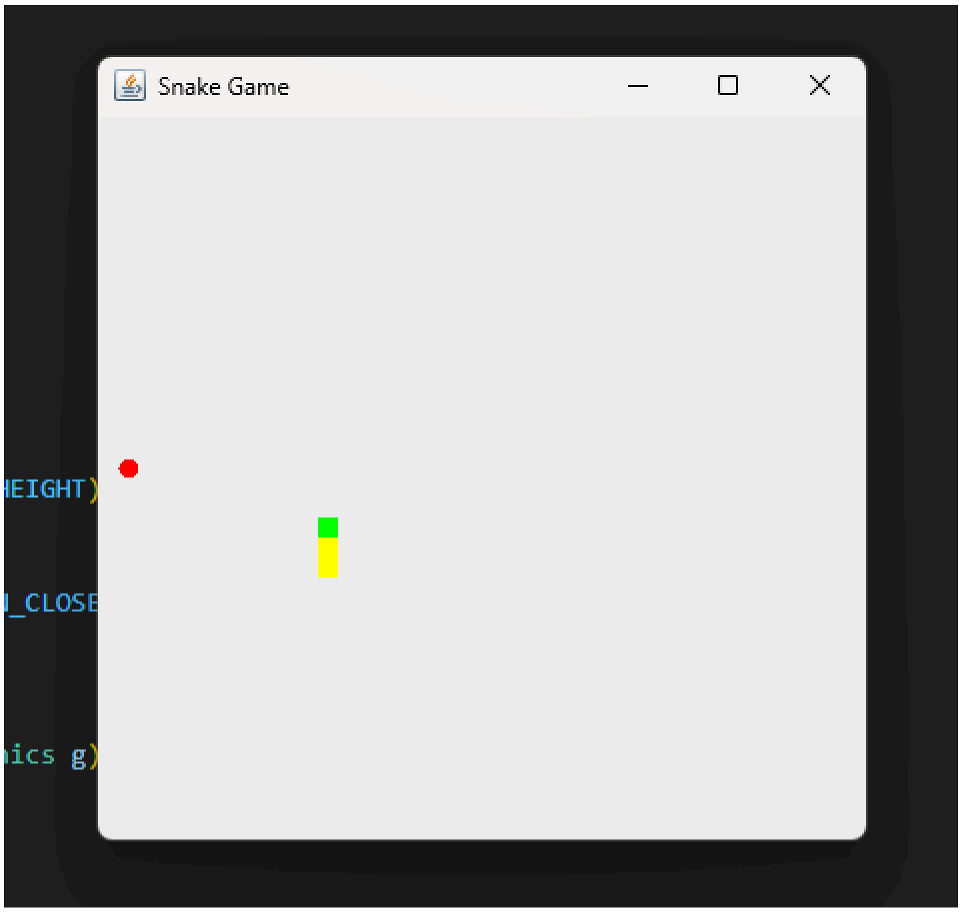
Laddi Singh(321839)

Table of Contents

1. Introduction

A classic Snake game using Java with a graphical user interface (GUI). The objective of the game is for the player to control a snake on a grid, guiding it to eat apples while avoiding collision with walls

Preview: 1

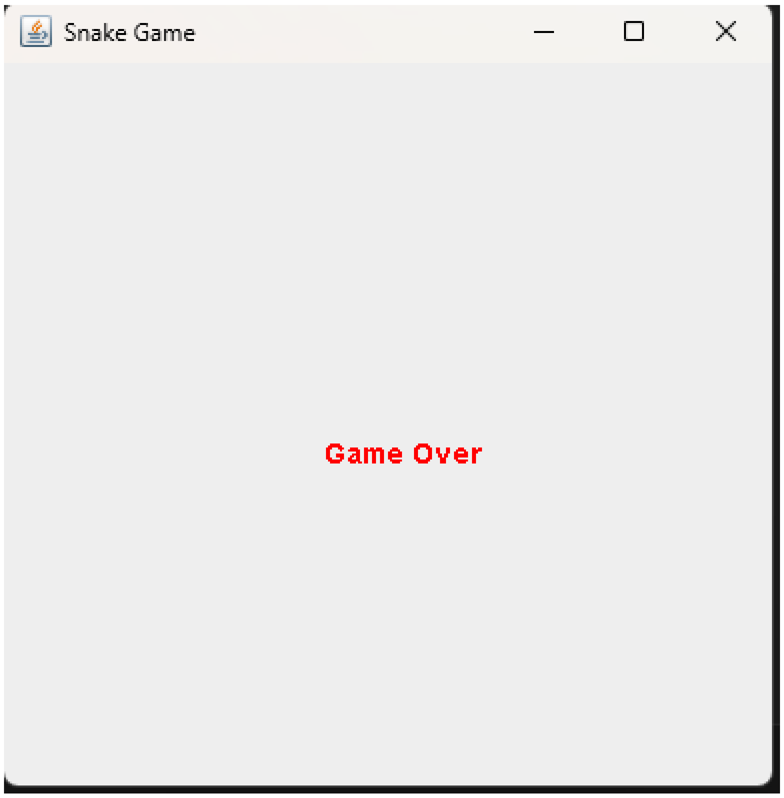


2. Functionality

The main functionality of the application includes:

- The game features a simple GUI window that displays the game grid, the snake, and the apple.
- The snake moves in the direction specified by the arrow keys.
- When the snake eats an apple, its length increases, and a new apple appears at a random location.
- The game ends if the snake collides with the walls of the game area or with its own body.
- Upon game over, a "Game Over" message is displayed.

Preview 2:



3. Code Explanation

The game is implemented in Java using the Swing library for the GUI.
Key components of the code include:

- Initialization of game variables and components.
- Drawing the game grid, snake, and apple on the screen.
- Handling user input to control the snake's movement.
- Checking for collisions with the walls, snake's body, and apples.
- Updating the game state and redrawing the screen at regular intervals using a timer.

CODE

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;

public class SnakeGame extends JFrame {
   private static final int WIDTH = 400;
   private static final int HEIGHT = 400;
   private static final int DOT_SIZE = 10;
   private static final int ALL_DOTS = (WIDTH * HEIGHT) / (DOT_SIZE * DOT_SIZE);
   private static final int RAND_POS = 29;
   private final int x[] = new int[ALL_DOTS];
   private final int y[] = new int[ALL_DOTS];
   private int dots;
   private int appleX;
   private int appleY;
   private boolean leftDirection = false;
   private boolean rightDirection = true;
   private boolean upDirection = false;
   private boolean downDirection = false;
   private boolean inGame = true;

   public SnakeGame() {
      initGame();
      addKeyListener(new MyKeyAdapter());
      setFocusable(true);
      setBackground(Color.BLACK);
```

```
        setPreferredSize(new Dimension(WIDTH, HEIGHT));
        setTitle("Snake Game");
        setLocationRelativeTo(null);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        JPanel gamePanel = new JPanel() {
            @Override
            protected void paintComponent(Graphics g) {
                super.paintComponent(g);
                doDrawing(g);
            }
        };
        getContentPane().add(gamePanel);
        pack();

        Timer timer = new Timer(100, e -> {
            if (inGame) {
                checkApple();
                checkCollision();
                move();
                repaint();
            }
        });
        timer.start();
    }

    private void initGame() {
        dots = 3;
        for (int z = 0; z < dots; z++) {
            x[z] = 50 - z * DOT_SIZE;
            y[z] = 50;
        }
        locateApple();
    }

    private void doDrawing(Graphics g) {
        if (inGame) {
            g.setColor(Color.RED);
            g.fillOval(appleX, appleY, DOT_SIZE, DOT_SIZE);
            for (int z = 0; z < dots; z++) {
                if (z == 0) {
                    g.setColor(Color.GREEN);
                    g.fillRect(x[z], y[z], DOT_SIZE, DOT_SIZE);
                } else {
```

```
                g.setColor(Color.YELLOW);
                g.fillRect(x[z], y[z], DOT_SIZE, DOT_SIZE);
            }
        }
        Toolkit.getDefaultToolkit().sync();
    } else {
        gameOver(g);
    }
}

private void gameOver(Graphics g) {
    String msg = "Game Over";
    Font small = new Font("Helvetica", Font.BOLD, 14);
    FontMetrics metr = getFontMetrics(small);
    g.setColor(Color.RED);
    g.setFont(small);
    g.drawString(msg, (WIDTH - metr.stringWidth(msg)) / 2, HEIGHT / 2);
}

private void checkApple() {
    if ((x[0] == appleX) && (y[0] == appleY)) {
        dots++;
        locateApple();
    }
}

private void move() {
    for (int z = dots; z > 0; z--) {
        x[z] = x[(z - 1)];
        y[z] = y[(z - 1)];
    }
    if (leftDirection) {
        x[0] -= DOT_SIZE;
    }
    if (rightDirection) {
        x[0] += DOT_SIZE;
    }
    if (upDirection) {
        y[0] -= DOT_SIZE;
    }
    if (downDirection) {
        y[0] += DOT_SIZE;
    }
}
```

```
private void checkCollision() {
    for (int z = dots; z > 0; z--) {
        if ((z > 4) && (x[0] == x[z]) && (y[0] == y[z])) {
            inGame = false;
        }
    }
    if (y[0] >= HEIGHT) {
        inGame = false;
    }
    if (y[0] < 0) {
        inGame = false;
    }
    if (x[0] >= WIDTH) {
        inGame = false;
    }
    if (x[0] < 0) {
        inGame = false;
    }
    if (!inGame) {
        Timer timer = new Timer(2000, e -> System.exit(0));
        timer.setRepeats(false);
        timer.start();
    }
}

private void locateApple() {
    int r = (int) (Math.random() * RAND_POS);
    appleX = ((r * DOT_SIZE));
    r = (int) (Math.random() * RAND_POS);
    appleY = ((r * DOT_SIZE));
}

private class MyKeyAdapter extends KeyAdapter {
    @Override
    public void keyPressed(KeyEvent e) {
        int key = e.getKeyCode();
        if ((key == KeyEvent.VK_LEFT) && (!rightDirection)) {
            leftDirection = true;
            upDirection = false;
            downDirection = false;
        }
        if ((key == KeyEvent.VK_RIGHT) && (!leftDirection)) {
            rightDirection = true;
```

```
                upDirection = false;
                downDirection = false;
            }
            if ((key == KeyEvent.VK_UP) && (!downDirection)) {
                upDirection = true;
                rightDirection = false;
                leftDirection = false;
            }
            if ((key == KeyEvent.VK_DOWN) && (!upDirection)) {
                downDirection = true;
                rightDirection = false;
                leftDirection = false;
            }
        }
    }

    public static void main(String[] args) {
        EventQueue.invokeLater(() -> {
            JFrame ex = new SnakeGame();
            ex.setVisible(true);
        });
    }
}
```

4. Guide and Rules

- Control Snake with Arrow Kays
- Each Apple grow snake by one block
- Don’t let snake hit or collide with walls
- When Game is over Application will quit itself

6 . Conclusion

This project provides a basic implementation of the Snake game in Java with a simple GUI. While the game lacks advanced features. It is very simple and minimal.