BUIDUCMANH – ITITIU21076

Flappy Bird Game - OOP Project Report

1. Project Overview

A Java implementation of Flappy Bird using OOP principles, featuring a bird navigating through pipes while avoiding collisions.

2. Core Algorithms

2.1 Game Loop Algorithm

```java
Timer gameLoop = new Timer(1000/60, this);
```

- Runs at 60 FPS
- Each frame:

1. Updates game state
2. Renders graphics
3. Checks game conditions

- Ensures smooth, consistent gameplay

2.2 Physics Algorithm

```java
// Bird Movement
velocityY += gravity;          // Apply gravi
bird.y += velocityY;          // Update posit
bird.y = Math.max(bird.y, 0); // Prevent go

// Jump
if (spacePressed) {
    velocityY = -9;           // Upward veloc
}
```

- Implements simple physics:
- Gravity: Constant downward acceleration
- Jump: Instant upward velocity
- Position: Updated based on velocity
- Creates parabolic motion for realistic bird movement

2.3 Pipe Generation Algorithm

```java
void placePipes() {
    // Random position for top pipe
    int randomPipeY = (int) (pipeY - pipeHe
    int openingSpace = boardHeight/4;   // F

    // Create pipe pair
    Pipe topPipe = new Pipe(topPipeImg);
    topPipe.y = randomPipeY;

    Pipe bottomPipe = new Pipe(bottomPipeIm
    bottomPipe.y = topPipe.y + pipeHeight +
}
```

- Creates challenging but fair obstacles:
- Random vertical positions

- Fixed gap size
- Paired top and bottom pipes
- Ensures playable difficulty

2.4 Collision Detection Algorithm

```java
boolean collision(Bird a, Pipe b) {
    return a.x < b.x + b.width &&
           a.x + a.width > b.x &&
           a.y < b.y + b.height &&
           a.y + a.height > b.y;
}
```

- Uses AABB (Axis-Aligned Bounding Box) detection
- Checks rectangle intersection
- Efficient and accurate collision detection

2.5 Scoring Algorithm

```java
if (!pipe.passed && bird.x > pipe.x + pipe.
    score += 0.5;  // 0.5 points per pipe (
    pipe.passed = true;
}
```

- Tracks pipe passage
- Prevents double counting
- Simple but effective scoring system

3. OOP Implementation

3.1 Class Hierarchy

```java
public class FlappyBird extends JPanel impl
    // Main game class
    class Bird { /* Bird implementation */
    class Pipe { /* Pipe implementation */
}
```

3.2 Inheritance Structure

```java
// Class Inheritance
FlappyBird extends JPanel
    └── Inherits graphical capabilities
    └── Gets panel properties and methods

// Interface Implementation
FlappyBird implements ActionListener
    └── Handles game loop events
    └── Manages game state updates

FlappyBird implements KeyListener
    └── Processes keyboard input
    └── Controls bird movement
```

3.3 Encapsulation Details

3.3.1 Bird Class Encapsulation

```java
class Bird {
    // Private properties
    private int x = birdX;
    private int y = birdY;
    private int width = birdWidth;
    private int height = birdHeight;
    private Image img;

    // Constructor
    Bird(Image img) {
        this.img = img;
    }

    // Methods
    public void update() {
        // Bird movement logic
    }
}
```

- Properties are encapsulated within the class
- Access controlled through methods
- Clear separation of concerns

3.3.2 Pipe Class Encapsulation

```java
class Pipe {
    // Private properties
    private int x = pipeX;
    private int y = pipeY;
    private int width = pipeWidth;
    private int height = pipeHeight;
    private Image img;
    private boolean passed = false;

    // Constructor
    Pipe(Image img) {
        this.img = img;
    }

    // Methods
    public void move() {
        // Pipe movement logic
    }
}
```

- Internal state management
- Controlled access to properties
- Clear interface for interaction

3.4 Polymorphism Implementation

3.4.1 Interface Polymorphism

```java
// ActionListener implementation
@Override
public void actionPerformed(ActionEvent e)
    move();
    repaint();
    if (gameOver) {
        placePipeTimer.stop();
        gameLoop.stop();
    }
}

// KeyListener implementation
@Override
public void keyPressed(KeyEvent e) {
    if (e.getKeyCode() == KeyEvent.VK_SPACE
        velocityY = -9;
        // Game restart logic
    }
}
```

- Multiple interface implementations
- Different behaviors for different events
- Flexible event handling

3.4.2 Method Overriding

```java
@Override
public void paintComponent(Graphics g) {
    super.paintComponent(g);
    draw(g);
}

@Override
public void keyTyped(KeyEvent e) {}
@Override
public void keyReleased(KeyEvent e) {}
```

- Custom rendering behavior

- Specialized event handling
- Parent class method extension

4. Conclusion

The implementation successfully demonstrates:

- Core game algorithms
- OOP principles
- Efficient game mechanics
- Clean code structure

The game's success relies heavily on these well-implemented algorithms, particularly the physics and collision detection systems, which create the challenging yet fair gameplay that makes Flappy Bird engaging.