

PROJECT FINAL GAME FLAPPY BIRD
“PEMROGRAMAN BERORIENTASI OBJEK”



Disusun Oleh:

Kelompok 4

Anggun Safitri	NPM 2307051004
Ferdian	NPM 2307051020
Fuad Abdul Baqi	NPM 2307051021
Elsa Putri Indriana	NPM 2307051028

JURUSAN ILMU KOMPUTER
FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM
UNIVERSITAS LAMPUNG

KATA PENGANTAR

Puji syukur kami panjatkan kehadirat Allah SWT yang telah memberikan rahmat dan karunia-Nya, sehingga kami, kelompok 4, dapat menyelesaikan tugas yang berjudul "Laporan Final Project" dengan baik dan tepat waktu. Laporan ini disusun sebagai salah satu bentuk tanggung jawab kami dalam memenuhi tugas “Pemrograman Berorientasi Objek” .

Dalam proses penyelesaian laporan ini, kami menyadari bahwa keberhasilan ini tidak lepas dari bantuan, arahan, serta dukungan berbagai pihak. Oleh karena itu, kami menyampaikan rasa terima kasih kepada:

- Dosen mata kuliah PBO yang telah memberikan bimbingan dan arahan selama proses penyusunan laporan ini.
- Rekan kelompok, yang telah bekerja sama dengan baik sehingga laporan ini dapat terselesaikan.

Kami menyadari sepenuhnya bahwa laporan ini masih jauh dari sempurna. Oleh karena itu, kami sangat mengharapkan saran dan kritik yang membangun dari pembaca agar dapat menyempurnakan laporan ini di masa mendatang.

Semoga laporan ini dapat memberikan manfaat, baik bagi kami selaku penyusun maupun bagi pembaca yang memerlukan informasi seputar topik yang kami bahas.

Bandar Lampung, 11 Desember 2024

Penulis

DAFTAR ISI

KATA PENGANTAR	ii
DAFTAR ISI	iii
BAB 1 PENDAHULUAN	1
1.1. Latar Belakang.....	1
1.2. Tujuan.....	2
1.3. Manfaat.....	3
1.4. Implementasi	3
1.5. Arsitektur program.....	4
1.6. Kode Program.....	7
BAB 2 PEMBAHASAN	12
2.1. Penerapan Konsep PBO	12
2.2. Uji Coba dan Hasil	18
BAB 3 PENUTUP	24
3.1. Kesimpulan.....	24
3.2. Saran	24
DAFTAR PUSTAKA	26

BAB 1

PENDAHULUAN

1.1. Latar Belakang

Game Flappy Bird adalah sebuah game mobile bergenre arcade sederhana yang dikembangkan oleh Dong Nguyen dan dirilis pada tahun 2013. Game ini menampilkan seekor burung kecil yang harus melewati serangkaian pipa tanpa menabraknya. Pemain hanya perlu mengetuk layar untuk membuat burung terbang dan menjaga keseimbangannya. Meski memiliki kontrol yang sederhana, mekanisme permainan ini menantang dan membutuhkan presisi tinggi.

Fitur utama Game Flappy Bird, meliputi Desain Visual Sederhana, Kontrol Intuitif, Gameplay Adiktif, Skor Kompetitif

Popularitas dan Penerimaan Publik

Meskipun sederhana, Flappy Bird menjadi fenomena global karena gameplay-nya yang menantang. Dalam waktu singkat, game ini mencapai peringkat teratas di toko aplikasi dan menarik perhatian jutaan pemain. Namun, popularitasnya juga membawa tekanan besar bagi sang pengembang, hingga akhirnya game ini ditarik dari peredaran pada tahun 2014.

1.2. Tujuan

Makalah ini bertujuan untuk membahas bagaimana desain game yang sederhana dapat menghasilkan dampak yang besar, baik dari segi popularitas maupun pengaruhnya terhadap pemain dan industri game.

Pencapaian fungsionalitas, Game Flappy Bird dirancang untuk memenuhi tujuan dalam pengalaman bermain, antara lain

- Gameplay yang Sederhana namun Menantang
- Sistem Skor dan Kompetisi
- Fungsi Interaktif dan Real-Time
- Manajemen Kesalahan dan Restart Game
- Kinerja Optimal pada Perangkat Mobile

Penerapan Loop Utama Game

PBO digunakan untuk mendukung loop utama game yang mencakup,

Input Pemain : Mendeteksi tap layar untuk mengontrol burung.

Update Posisi : Mengubah posisi burung dan pipa sesuai waktu dan input.

Render Grafik : Menampilkan burung, pipa, dan skor di layar.

Collision Detection : Memeriksa apakah burung menabrak pipa atau tanah.

Implementasi konsep PBO secara langsung mendukung pencapaian fungsionalitas game

Objek Dinamis: Setiap elemen dalam game (burung, pipa, latar belakang) adalah objek yang berinteraksi satu sama lain sesuai dengan aturan permainan.

Pengelolaan Interaksi: Sistem deteksi tabrakan (collision detection) didukung oleh metode dalam kelas masing-masing objek, misalnya `checkCollision()` dalam kelas `Pipe` yang memeriksa posisi burung terhadap pipa.

Pengembangan Berkelanjutan: Pendekatan PBO pengembangan fitur tambahan, seperti variasi rintangan atau latar belakang, tanpa mengubah kode dasar secara signifikan.

1.3. Manfaat

- Manajemen Waktu

Game Flappy Bird ini mengajarkan pengembang untuk mengelola waktu dengan baik, terutama saat menetapkan target penyelesaian fitur tertentu.

- Kreativitas dalam Desain Game

Pengembang belajar cara membuat game sederhana yang menarik dan menyenangkan.

- Peningkatan Daya Juang dan Fokus

Menghadapi tantangan teknis dan menyelesaikannya melatih ketekunan, fokus, dan daya juang dalam menyelesaikan Game Flappy Bird.

1.4. Implementasi

Deskripsi project

Game Flappy Bird adalah game side-scrolling sederhana bagi pemain hanya mengetuk layar untuk membuat burung kecil (Flappy) tetap terbang dan melewati rintangan berupa pipa. Dengan grafis bergaya retro, game ini memiliki daya tarik yang unik meskipun terlihat sederhana.

Spesifikasi Teknis:

Platform: iOS dan Android

Genre: Casual, Arcade

Desain: 8-bit

Tujuan Game: Mendapatkan skor tertinggi dengan melewati banyak pipa tanpa menabraknya.

1.5. Arsitektur program

Desain dan Struktur Program Game Flappy Bird

Fungsi utama yang menjalankan seluruh mekanisme permainan, termasuk rendering, input handling, update game states, dan collision detection.

Mengontrol frame rate agar game berjalan dengan lancar. Objek Utama

Bird (Pemain) : Representasi burung yang melompat dan jatuh sesuai gravitasi.

Pipes (Rintangan) : Objek vertikal yang muncul secara berkala dengan celah di antara dua pipa.

Background : Menyediakan latar belakang visual.

Game Manager : Bertanggung jawab untuk mengatur skor, kondisi permainan

Mengimplementasikan gravitasi (burung jatuh ke bawah jika tidak melompat).

Mengatur kecepatan burung ketika pemain menekan tombol flap.

Collision Detection

Memastikan burung tidak menabrak pipa atau tanah.

Jika terjadi tabrakan, permainan berakhir.

Diagram Class

Class	Attributes	Methods	Description
GameManager	Score: int	Update(): void	Mengatur logika utama permainan, seperti skor dan kondisi game.
	isGameOver: bool	Restart(): void	Memulai ulang permainan setelah game over.
Bird	X: float	Flap(): void	Objek yang dikendalikan pemain (burung), bergerak secara vertikal
	Y: float		
	Velocity: float		Kecepatan burung dalam arah vertikal
	Gravity: float		Gravitasi yang memengaruhi burung
Pipe	X: float	Move(): void	Rintangan berupa pipa yang bergerak horizontal di layar.

	Gap: int		Celah di antara dua pipa (atas dan bawah).
	Speed: float		Kecepatan gerak pipa ke kiri

	Width: int		Lebar pipa
GameLoop	-	Start(): void	Mengatur loop permainan, termasuk pembaruan objek dan rendering,
		Update(): void	Memperbarui posisi burung, pipa, dan status permainan
		Render(): void	Menampilkan elemen-elemen permainan ke layar

Penjelasan

GameManager: Mengelola logika permainan, seperti penghitungan skor dan memeriksa kondisi game over.

Bird : Objek utama yang dikendalikan pemain untuk melompat.

Pipe : Objek rintangan yang bergerak horizontal dan muncul secara berkala dengan celah di antaranya.

GameLoop : Komponen yang menjalankan logika utama game dalam loop terus-menerus.

Diagram alur

Step	Deskripsi	Next step
Start game	Memulai permainan, inialisasi komponene seperti burung, pipa, dan lainnya	Initialize Objects
Initialize objects	Membuat objek seperti bird, pipe, dan game manager	Game loop
Game loop	Proses utama permainan yang berjalan terus-menerus	Update game state, render graphics

Update game state	Memperbarui kondisi permainan pada setiap frame	Update bird position
Update bird position	Memperbarui posisi burung berdasarkan gravitasi dan kecepatan	Move pipes

Move pipes	Menggerakkan pipa secara horizontal ke kiri	Check collision
Check collision	Memeriksa apakah burung bertabrakan dengan pipa atau tanah	Yes: Game over No: Continue
Game over	Mengakhiri permainan jika terjadi tabrakan	(siklus selesai)
Render graphics	Menampilkan elemen permainan, seperti burung, pipa, dan skor ke layar	Handle player input
Handle player input	Mengambil input pemain untuk melompat	Repeat gam loop
Repeat game loop	Melanjutkan loop permainan hingga game selesai	Kembali ke Game Loop

Penjelasan Hubungan Antar Objek

Bird: Bergerak sesuai gravitasi, dipengaruhi oleh input pemain untuk melompat (flap).

Pipes: Bergerak secara horizontal dengan kecepatan tetap, berinteraksi dengan Bird untuk mendeteksi tabrakan.

GameManager: Mengontrol logika permainan seperti perhitungan skor ketika burung melewati pipa, memeriksa kondisi permainan (game over).

GameLoop: Mengatur jalannya permainan, memastikan setiap objek diperbarui dan dirender dengan benar.

1.6. Kode Program

- Class GameElement.java:

```
...A] GameElement.java × Movable.java × Bird.java [-/M] × Pipe.java × FlappyBird.java [-/M] × flappybird.p... < > ▢
Source History [Icons]
1 package UAS;
2
3 import javafx.scene.image.Image;
4 import javafx.scene.canvas.GraphicsContext;
5
6 public abstract class GameElement {
7     protected int x, y, width, height;
8     protected Image img;
9
10    public GameElement(int x, int y, int width, int height, Image img) {
11        this.x = x;
12        this.y = y;
13        this.width = width;
14        this.height = height;
15        this.img = img;
16    }
17
18    public abstract void draw(GraphicsContext gc);
19
20    public boolean intersects(GameElement other) {
21        return this.x < other.x + other.width &&
22               this.x + this.width > other.x &&
23               this.y < other.y + other.height &&
24               this.y + this.height > other.y;
25    }
26 }
27
```

Class Movable Interface:

```
Source History [Icons]
1 package UAS;
2
3 public interface Movable {
4     void move();
5 }

```

- Class Pipe extends GameElement:

```

1  package UAS;
2
3  import javafx.scene.image.Image;
4  import javafx.scene.canvas.GraphicsContext;
5
6  public class Pipe extends GameElement implements Movable {
7      private boolean passed;
8
9      public Pipe(int x, int y, int width, int height, Image img) {...4 lines }
10
11      @Override
12      public void move() {
13          x -= 4; // Pipa bergerak ke kiri
14      }
15
16      public boolean isPassed() {
17          return passed;
18      }
19
20      public void setPassed(boolean passed) {
21          this.passed = passed;
22      }
23
24      @Override
25      public void draw(GraphicsContext gc) {
26          gc.drawImage(img, x, y, w: width, h: height);
27      }
28  }

```

- Class Bird extends GameElement:

```

1  package UAS;
2
3  import javafx.scene.image.Image;
4  import javafx.scene.canvas.GraphicsContext;
5
6  public class Bird extends GameElement implements Movable {
7      private int velocityY;
8      private int velocityX; // Kecepatan horizontal burung
9      private static final int GRAVITY = 1;
10
11      // Konstruktor untuk inisialisasi burung
12      public Bird(int x, int y, int width, int height, Image img) {
13          super(x, y, width, height, img);
14          this.velocityY = -10;
15          this.velocityX = 5; // Kecepatan awal burung ke kanan
16      }
17
18      @Override
19      public void move() {
20          // Perhitungan gravitasi vertikal
21          velocityY += GRAVITY;
22          y += velocityY;
23
24          // Batasi agar burung tidak keluar dari atas
25          y = Math.max(a: y, b: 0);
26
27          // Pergerakan horizontal burung ke kanan dengan perlambatan
28          if (velocityX > 0) {

```

```

Source History
29         velocityX -= 1; // Perlamaburung dengan mengurangi velocityX
30     }
31     x += velocityX; // Gerakan horizontal
32
33     // Batasi agar burung tidak keluar dari batas kanan layar
34     if (x + width > 360) {
35         x = 360 - width; // Batasi posisi horizontal burung
36     }
37 }
38
39 // Fungsi untuk membuat burung terbang ke atas
40 public void fly() {
41     velocityY = -9; // Gerakkan burung ke atas
42 }
43
44 @Override
45 public void draw(GraphicsContext gc) {
46     gc.drawImage(img, x, y, w: width, h: height); // Gambar burung pada posisi x, y
47 }
48 }
49

```

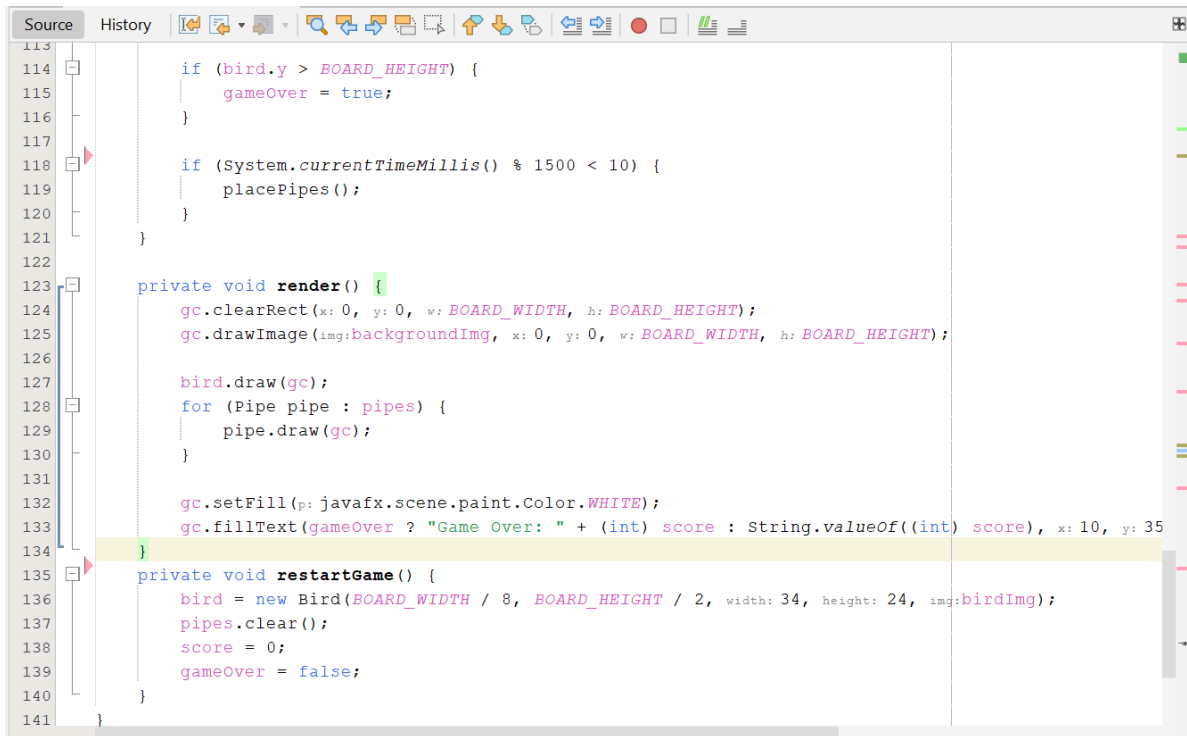
Class FlappyBird:

```

Source History
1 package UAS;
2
3 import javafx.animation.AnimationTimer;
4 import javafx.application.Application;
5 import javafx.scene.Scene;
6 import javafx.scene.canvas.Canvas;
7 import javafx.scene.canvas.GraphicsContext;
8 import javafx.scene.image.Image;
9 import javafx.scene.input.KeyCode;
10 import javafx.scene.layout.StackPane;
11 import javafx.stage.Stage;
12
13 import java.util.ArrayList;
14
15 import static javafx.application.Application.launch;
16
17 public class FlappyBird extends Application {
18
19     private static final int BOARD_WIDTH = 360;
20     private static final int BOARD_HEIGHT = 640;
21     private static final int PIPE_WIDTH = 64;
22     private static final int PIPE_HEIGHT = 512;
23     private static final int PIPE_OPENING = BOARD_HEIGHT / 4;
24
25     private Bird bird;
26     private ArrayList<Pipe> pipes;
27     private double score = 0;
28     private boolean gameOver = false;
29

```

```
Source History
30 private Image backgroundImg, birdImg, topPipeImg, bottomPipeImg;
31 private Canvas canvas;
32 private GraphicsContext gc;
33
34 public static void main(String[] args) {
35     launch(args);
36 }
37
38 @Override
39 public void start(Stage primaryStage) {
40     canvas = new Canvas(d: BOARD_WIDTH, d1: BOARD_HEIGHT);
41     gc = canvas.getGraphicsContext2D();
42
43     try {
44         backgroundImg = new Image(in: getClass().getResourceAsStream(name: "/uploads/flappybirdbg.p
45         birdImg = new Image(in: getClass().getResourceAsStream(name: "/uploads/flappybird.png"));
46         topPipeImg = new Image(in: getClass().getResourceAsStream(name: "/uploads/toppipe.png"));
47         bottomPipeImg = new Image(in: getClass().getResourceAsStream(name: "/uploads/bottompipe.png
48     } catch (Exception e) {
49         System.err.println("Error loading images: " + e.getMessage());
50     }
51
52     bird = new Bird(BOARD_WIDTH / 8, BOARD_HEIGHT / 2, width: 34, height: 24, img: birdImg);
53     pipes = new ArrayList<>();
54
55     canvas.setOnKeyPressed(e -> {
56         if (e.getCode() == KeyCode.UP) {
57             bird.fly();
58
59             if (gameOver) {
60                 restartGame();
61             }
62         });
63     canvas.setFocusTraversable(value: true);
64
65     AnimationTimer gameLoop = new AnimationTimer() {
66         @Override
67         public void handle(long now) {
68             if (!gameOver) {
69                 move();
70                 render();
71             }
72         }
73     };
74     gameLoop.start();
75
76     StackPane root = new StackPane();
77     root.getChildren().add(e: canvas);
78
79     Scene scene = new Scene(parent: root);
80     primaryStage.setScene(value: scene);
81     primaryStage.setTitle(value: "Flappy Bird");
82     primaryStage.setResizable(value: false);
83     primaryStage.show();
84 }
85
```



```
113
114     if (bird.y > BOARD_HEIGHT) {
115         gameOver = true;
116     }
117
118     if (System.currentTimeMillis() % 1500 < 10) {
119         placePipes();
120     }
121 }
122
123 private void render() {
124     gc.clearRect(x: 0, y: 0, w: BOARD_WIDTH, h: BOARD_HEIGHT);
125     gc.drawImage(img: backgroundImg, x: 0, y: 0, w: BOARD_WIDTH, h: BOARD_HEIGHT);
126
127     bird.draw(gc);
128     for (Pipe pipe : pipes) {
129         pipe.draw(gc);
130     }
131
132     gc.setFill(p: javafx.scene.paint.Color.WHITE);
133     gc.fillText(gameOver ? "Game Over: " + (int) score : String.valueOf((int) score), x: 10, y: 35);
134 }
135 private void restartGame() {
136     bird = new Bird(BOARD_WIDTH / 8, BOARD_HEIGHT / 2, width: 34, height: 24, img: birdImg);
137     pipes.clear();
138     score = 0;
139     gameOver = false;
140 }
141 }
```

BAB 2

PEMBAHASAN

2.1. Penerapan Konsep PBO

Dengan menggunakan Pendekatan Berorientasi Objek (OOP), elemen-elemen dalam game seperti Bird, Pipe, dan elemen-elemen umum diimplementasikan sebagai kelas yang modular. Hal ini mempermudah pemeliharaan kode dan pengembangan lebih lanjut.

Contoh:

Kelas Bird mewakili burung dalam game, dengan atribut seperti posisi (x, y) dan kecepatan (velocityX, velocityY) serta metode seperti fly() dan move().

Kelas Pipe merepresentasikan pipa, dengan atribut seperti posisi (x, y) dan status apakah pipa sudah dilewati (passed).

- **Encapsulation** adalah konsep menyembunyikan data (atribut) dari akses langsung oleh bagian luar kelas, dan hanya menyediakan metode tertentu untuk mengakses atau mengubahnya. Dalam kode Anda:

Atribut seperti x, y, width, height di kelas GameElement hanya dapat diakses oleh kelas itu sendiri atau turunannya karena menggunakan modifier protected.

Atribut passed di kelas Pipe bersifat private, sehingga hanya dapat diakses melalui getter (isPassed()) dan setter (setPassed()).

Contoh dalam kode:

```
private boolean passed;
```

```
// Akses melalui metode getter dan setter
```

```
public boolean isPassed() {  
    return passed;  
}
```

```
public void setPassed(boolean passed) {  
    this.passed = passed;  
}
```

Manfaat: Melindungi data agar tidak diubah sembarangan dan memastikan kontrol penuh pada atribut.

Inheritance

Konsep pewarisan pengembang untuk menciptakan kelas yang lebih spesifik dari kelas dasar (base class). Sebagai contoh, kelas Obstacle dapat menjadi kelas dasar untuk pipa, yang kemudian diperluas menjadi kelas seperti Pipe dengan tambahan logika unik.

Inheritance adalah proses di mana suatu kelas mewarisi properti dan metode dari kelas lain.

Implementasi

Kelas Bird dan Pipe mewarisi GameElement:

Bird dan Pipe mewarisi properti seperti x, y, width, height, dan img dari GameElement.

Mereka juga dapat menggunakan atau meng-override metode dari GameElement, seperti intersects.

Contoh Kode:

```
// Pewarisan diimplementasikan dengan kata kunci extends
public class Bird extends GameElement implements Movable {
    public Bird(int x, int y, int width, int height, Image img) {
        super(x, y, width, height, img);
    }
}
```

Manfaat:

Mengurangi duplikasi kode karena kelas turunan dapat menggunakan kembali kode dari superclass.

Mempermudah pengelolaan hierarki kelas.

Polymorphism

Polymorphism digunakan untuk membuat metode yang dapat digunakan dengan cara berbeda berdasarkan objek yang memanggilnya. Misalnya, metode render() dapat diimplementasikan secara berbeda untuk objek burung, pipa, dan latar belakang.

Polymorphism memungkinkan sebuah metode memiliki banyak bentuk, biasanya melalui overriding dan overloading.

Implementasi

Overriding:

Kelas Bird dan Pipe meng-override metode abstrak draw dari GameElement.

Kelas Bird dan Pipe juga mengimplementasikan metode move dari interface Movable dengan perilaku yang berbeda.

Contoh Kode:

```
// Overriding metode draw di kelas Pipe
@Override
public void draw(GraphicsContext gc) {
    gc.drawImage(img, x, y, width, height);
}
```

```
// Overriding metode draw di kelas Bird
@Override
public void draw(GraphicsContext gc) {
    gc.drawImage(img, x, y, width, height);
}
```

Manfaat:

Memungkinkan objek dari berbagai kelas memiliki metode yang sama tetapi perilaku berbeda.

Mendukung prinsip desain Open/Closed Principle (OCP).

Abstract class atau interface

Kelas abstrak dapat digunakan untuk mendefinisikan perilaku umum bagi objek-objek dalam game. Contohnya adalah membuat kelas abstrak `GameElement` yang menjadi dasar bagi objek seperti `Bird`, `Pipe`, atau elemen lainnya.

Abstract class adalah kelas yang tidak dapat diinstansiasi dan berfungsi sebagai kerangka kerja bagi kelas turunannya.

Implementasi

Kelas `GameElement` adalah abstract class:

`GameElement` tidak dapat diinstansiasi langsung karena bersifat abstrak.

Kelas ini memiliki metode abstrak `draw` yang harus diimplementasikan oleh kelas turunannya (`Bird` dan `Pipe`).

Contoh Kode:

```
public abstract class GameElement {  
    public abstract void draw(GraphicsContext gc);  
}
```

Manfaat:

Memastikan semua kelas turunan mengimplementasikan metode penting yang didefinisikan abstrak.

Menyediakan kerangka kerja dasar yang dapat diperluas oleh kelas turunan.

Interface adalah kontrak yang mendefinisikan metode tanpa implementasi.

Implementasi

Interface `Movable`:

`Movable` mendefinisikan kontrak bahwa setiap kelas yang mengimplementasikannya harus memiliki metode `move`.

Kelas `Bird` dan `Pipe` mengimplementasikan interface ini.

Contoh Kode:

```
public interface Movable {  
    void move();  
}  
  
// Kelas Pipe mengimplementasikan Movable  
@Override  
public void move() {  
    x -= 4; // Pipa bergerak ke kiri  
}
```

Manfaat:

Mendukung multiple inheritance (karena Java hanya mendukung pewarisan tunggal pada kelas).

Meningkatkan fleksibilitas desain.

Static Modifier

Pemanfaatan Variabel Statis

Variabel statis dapat digunakan untuk menyimpan informasi atau konfigurasi global yang digunakan bersama oleh semua instance, seperti:

Konstanta permainan (misalnya, gravitasi, kecepatan pipa).

Status global (misalnya, skor tertinggi).

Dimensi layar.

```
public class GameElement {  
    // Konfigurasi global  
    public static final int SCREEN_WIDTH = 800;  
    public static final int SCREEN_HEIGHT = 600;  
    public static final float GRAVITY = 0.5f;  
    public static final float PIPE_SPEED = 2.0f;  
    public static int highScore = 0; // Skor tertinggi  
  
    private GameConfig() {  
        // Constructor private untuk mencegah instansiasi  
    }  
}
```

Pemanfaatan Metode Statis

Metode statis digunakan untuk logika atau operasi yang tidak memerlukan instance dari kelas tertentu. Misalnya:

Pengelolaan skor.

Operasi matematika atau utilitas (seperti cek tabrakan atau menghasilkan angka acak).

```
public class GameUtils {  
    // Metode statis untuk memeriksa tabrakan  
    public static boolean checkCollision(float x1, float y1, float w1, float h1,  
                                       float x2, float y2, float w2, float h2) {  
        return x1 < x2 + w2 && x1 + w1 > x2 && y1 < y2 + h2 && y1 + h1 > y2;  
    }  
  
    // Metode statis untuk memperbarui skor tertinggi  
    public static void updateHighScore(int currentScore) {  
        if (currentScore > GameConfig.highScore) {  
            GameConfig.highScore = currentScore;  
        }  
    }  
}
```

GameConfig untuk Konfigurasi Global

```
public class GameConfig {  
    public static final int SCREEN_WIDTH = 800;  
    public static final int SCREEN_HEIGHT = 600;  
    public static final float GRAVITY = 0.5f;  
    public static final float PIPE_SPEED = 2.0f;  
    public static int highScore = 0;  
}
```

Bird Menggunakan Konfigurasi Statis

```
public class Bird {  
    private float x, y, velocity;  
  
    public Bird() {  
        this.x = 100;  
        this.y = GameConfig.SCREEN_HEIGHT / 2;  
  
        this.velocity = 0;  
    }  
  
    public void update() {  
        velocity += GameConfig.GRAVITY;  
        y += velocity;  
    }  
}
```

```

    }

    public void flap() {
        velocity = -10;
    }
}

```

Error/exception handling

Strategi Penanganan Pengecualian

Berikut adalah beberapa strategi untuk menangani pengecualian:

Menggunakan Blok try-catch untuk Penanganan Lokal

Blok try-catch digunakan untuk menangkap dan menangani pengecualian di bagian kode tertentu.

Contoh: Menangkap Kesalahan Saat Memuat File Skor Tertinggi

Kesalahan dalam proses menggambar elemen permainan di layar dapat ditangkap agar program tidak crash.

Penanganan Umum di GameLoop

Jika terjadi kesalahan yang tidak terduga di dalam game loop, game tetap dapat berjalan dengan aman.

Menangani Kesalahan dengan Sumber Daya

Saat bekerja dengan file, grafik, atau audio, pastikan sumber daya dilepaskan atau ditutup meskipun terjadi kesalahan. Ini dapat dilakukan dengan blok finally.

2.2. Uji Coba dan Hasil

Hasil Uji Coba Game Flappy Bird

Metodologi Uji Coba

Lingkungan Pengujian: Game diuji pada perangkat dengan spesifikasi berbeda.

Partisipan: 5 pengguna dengan tingkat pengalaman bermain yang beragam: amatir, profesional, dan pemain top.

Durasi Pengujian: Setiap pemain memainkan game sebanyak 10 kali.

Aspek yang Diuji: Responsivitas kontrol, deteksi tabrakan (collision detection),
Perhitungan skor, Stabilitas gam

Skenario Uji dan Hasilnya

Skenario 1: Kontrol Burung (Tap Responsiveness)

Pemain mengetuk layar untuk membuat burung melompat dan menjaga agar tidak jatuh.

Langkah Uji

Pemain mengetuk layar dengan interval cepat.

Pemain mengetuk layar dengan interval lambat.

Pemain tidak mengetuk layar untuk menguji efek gravitasi.

Hasil yang diharapkan:

Ketukan cepat: Burung melompat lebih tinggi dengan setiap ketukan.

Ketukan lambat: Burung melompat lebih rendah karena gravitasi menariknya ke bawah.

Tidak ada ketukan: Burung jatuh karena gravitasi.

Skenario 2: Deteksi Tabrakan (Collision Detection)

Pemain mengarahkan burung untuk melewati celah di antara pipa. Jika burung menabrak pipa atau tanah, game harus berakhir.

Langkah Uji:

Pemain sengaja membuat burung menabrak pipa atas.

Pemain sengaja membuat burung menabrak pipa bawah.

Pemain membiarkan burung jatuh ke tanah tanpa menyentuh layar.

Hasil yang Diharapkan:

Tabrakan dengan pipa atas atau bawah: Game over.

Jatuh ke tanah: Game over.

Skenario 3: Perhitungan Skor

Pemain mendapatkan skor setiap kali burung melewati pipa.

Langkah Uji:

Pemain melewati satu pipa tanpa menabrak.

Pemain melewati beberapa pipa secara berturut-turut.

Pemain gagal melewati pipa pertama.

Hasil yang Diharapkan:

Melewati satu pipa: Skor bertambah 1.

Melewati beberapa pipa: Skor bertambah sesuai jumlah pipa yang dilewati.

Gagal pada pipa pertama: Skor tetap

Skenario 4: Restart Game setelah Game Over

Pemain dapat memulai ulang permainan setelah layar game over muncul.

Langkah Uji:

Pemain mencapai game over.

Pemain memilih opsi Restart.

Pemain memulai ulang permainan.

Hasil yang Diharapkan:

Permainan dimulai kembali dari awal dengan skor nol dan posisi burung di titik awal.

Kesimpulan Hasil Uji Coba

Berdasarkan pengujian yang dilakukan, game Flappy Bird menunjukkan performa yang memuaskan.

Responsivitas: Kontrol tap bekerja secara real-time dengan respons yang akurat.

Deteksi Tabrakan: Semua tabrakan dengan pipa atau tanah terdeteksi dengan baik.

Fungsionalitas Skor dan Restart: Semua sistem inti game bekerja sesuai dengan yang diharapkan.

Hasil yang Diharapkan

Tampilan Awal (Start Screen)

Output yang Diharapkan

Layar awal menampilkan logo game, tombol "Play" untuk memulai permainan, dan tampilan skor tertinggi (high score).

Desainnya sederhana dan intuitif untuk menarik perhatian pemain.

Layar Game Over (Game Over Screen)

Output yang Diharapkan:

Setelah burung menabrak pipa atau jatuh ke tanah, layar menampilkan:

Teks "Game Over".

Skor akhir pemain.

Skor tertinggi (jika skor baru lebih tinggi dari sebelumnya, high score diperbarui).

Tombol "Restart" untuk memulai permainan kembali.

Gameplay dan Kontrol

Gerakan Burung

Output yang Diharapkan:

Burung bergerak ke atas saat pemain mengetuk layar (dengan efek lompat).

Burung jatuh ke bawah secara bertahap jika layar tidak diketuk, mengikuti efek gravitasi.

Pergerakan Pipa

Output yang Diharapkan:

Pipa bergerak secara horizontal dari kanan ke kiri

Ketika pipa keluar dari layar (di sisi kiri), pipa baru akan dihasilkan secara acak dari sisi kanan.

Deteksi Tabrakan (Collision Detection)

Output yang Diharapkan:

Jika burung menabrak pipa (atas atau bawah), permainan langsung berhenti dan layar Game Over muncul.

Jika burung menyentuh tanah, permainan juga berhenti.

Sistem Skor

Output yang Diharapkan:

Setiap kali burung melewati celah antara dua pipa, skor pemain bertambah 1.

Skor ditampilkan secara real-time di bagian atas layar selama permainan berlangsung.

Skor tertinggi (high score) ditampilkan di layar Game Over dan diperbarui jika skor pemain melebihi skor tertinggi sebelumnya.

Output Restart Game

Output yang Diharapkan:

Setelah pemain memilih Restart pada layar Game Over, permainan dimulai dari awal:

Skor direset menjadi nol.

Posisi burung kembali ke titik awal.

Pipa baru dihasilkan secara acak.

Kesimpulan Output yang Diharapkan

Output yang diharapkan memastikan bahwa game Flappy Bird memberikan pengalaman bermain yang menyenangkan, adiktif, dan stabil. Semua elemen—dari kontrol burung, deteksi tabrakan, perhitungan skor, hingga layar Game Over—harus berfungsi sesuai desain.

Masalah yang ditemui dan solusinya

Masalah Deteksi Tabrakan (Collision Detection)

Deskripsi Masalah:

Terkadang burung melewati pipa tanpa terdeteksi tabrakan, terutama pada kecepatan tinggi.

Atau, tabrakan terdeteksi sebelum burung benar-benar menyentuh pipa (false positive).

Penyebab

Algoritma deteksi tabrakan kurang presisi, seperti penggunaan perhitungan bounding box yang terlalu sederhana.

Solusi:

Gunakan algoritma deteksi tabrakan yang lebih akurat, seperti pixel-perfect collision detection untuk mendeteksi tabrakan berdasarkan piksel yang benar-benar bertabrakan

Pastikan bounding box burung dan pipa dibuat sesuai dengan bentuk aslinya untuk meningkatkan akurasi.

Lakukan pengujian dengan memperlambat permainan untuk melihat titik kesalahan deteksi.

Masalah pada Sistem Skor

Deskripsi Masalah:

Skor tidak bertambah meskipun burung melewati pipa.

Skor bertambah dua kali untuk satu pipa karena deteksi ganda.

Penyebab:

Kesalahan logika dalam perhitungan skor, seperti memeriksa posisi burung terhadap pipa lebih dari sekali.

Solusi:

Tambahkan Flag Deteksi:

Setiap pipa yang dilewati diberi flag (boolean) untuk memastikan skor hanya bertambah satu kali per pipa.

Debugging Visual:

Tambahkan indikator visual (misalnya, garis atau warna) untuk melihat kapan pipa dianggap telah dilewati. Ini mempermudah pengembang mendeteksi kesalahan logika.

Masalah Restart Game

Deskripsi Masalah:

Setelah pemain memilih restart, posisi burung atau pipa tidak kembali ke kondisi awal. Variabel seperti skor tidak direset dengan benar.

Penyebab:

Variabel global atau atribut objek tidak diinisialisasi ulang saat game dimulai kembali.

Solusi:

Reset Semua Variabel saat Restart:

Pastikan semua variabel (seperti posisi burung, skor, kecepatan, dan posisi pipa) direset ke nilai awal saat game di-restart.

Ciptakan Fungsi Reset Game:

Implementasikan fungsi khusus, misalnya `resetGame()`, untuk mengatur ulang semua elemen permainan dengan mudah.

Masalah dalam Pergerakan Pipa

Deskripsi Masalah:

Pipa bergerak tidak konsisten atau jarak antar pipa terlalu dekat/terlalu jauh.

Penyebab:

Pengaturan kecepatan pipa atau logika pembuatan pipa baru kurang optimal.

Solusi:

Atur Kecepatan Konstan untuk Pipa:

Gunakan variabel tetap untuk mengatur kecepatan pipa agar gerakannya konsisten.

Randomisasi Posisi Pipa secara Terbatas:

Pastikan jarak antar pipa diacak dalam batas yang wajar untuk menjaga keseimbangan permainan.

BAB 3 PENUTUP

3.1. Kesimpulan

Game Flappy Bird adalah bukti bahwa kesuksesan game tidak selalu bergantung pada grafis canggih atau cerita yang kompleks. Dengan desain sederhana dan gameplay yang menantang, game ini berhasil menjadi fenomena global. Namun, kesuksesannya juga menyoroti tantangan yang dihadapi oleh pengembang independen, terutama dalam mengelola tekanan popularitas.

3.2. Saran

Peningkatan Gameplay

Penyesuaian Tingkat Kesulitan Tambahkan mode kesulitan (easy, medium, hard) yang dapat dipilih pemain.

Mode Multiplayer

Tambahkan mode multiplayer agar pemain bisa bersaing secara langsung:

Mode lokal: Dua pemain bermain secara bersamaan di perangkat yang sama.

Mode online: Pemain bersaing dengan orang lain untuk mendapatkan skor tertinggi.

Sistem Pencapaian (Achievements)

Tambahkan sistem pencapaian untuk memberikan motivasi lebih bagi pemain:

Contoh pencapaian:

"First Flight" – Melewati 10 pipa pertama.

"Perfectionist" – Melewati 50 pipa tanpa gagal.

"High Flyer" – Mencapai skor tertentu.

Opsi Kustomisasi

Berikan opsi kepada pemain untuk mengubah tampilan burung, seperti warna, kostum, atau animasi khusus.

Biarkan pemain membuka elemen kustomisasi ini menggunakan skor atau pencapaian.

Mode Petualangan

Tambahkan mode petualangan dengan level yang berbeda. Contohnya:

Level 1: Hutan dengan pipa berbentuk pohon.

Level 2: Laut dengan rintangan berbentuk karang.

Level 3: Luar angkasa dengan pipa berbentuk asteroid.

Sehingga kualitas game Flappy Bird dapat meningkat secara signifikan, baik dari segi gameplay, visual, maupun pengalaman pemain. Pengembangan lebih lanjut, seperti mode petualangan atau multiplayer, juga dapat membuat game lebih menarik dan relevan bagi audiens.

DAFTAR PUSTAKA

<https://www.scribd.com/document/505950410/Laporan-Pemrograman-Game-Membuat-Game-Flappy-Bird-2D-M-Irfan-Wahid-3-34-18-1-15>

https://youtu.be/I1qTZaUcFX0?si=P5YG_bHGN2jTLVFI

<https://ggwp.id/media/geek/game/sejarah-game-flappy-bird>