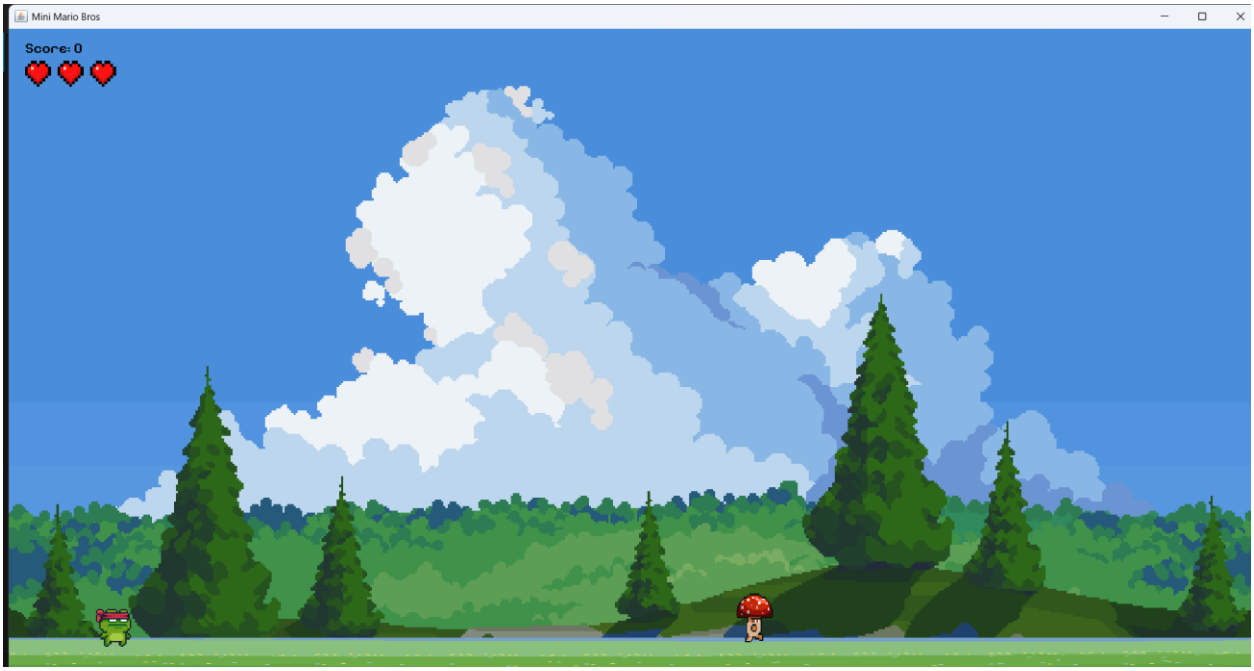
PORTOFOLIO TP-PBO 2025

A. Identitas Proyek

Nama Mahasiswa:
Muhammad Akbar Riyan Hartoto
Muhammad Miko Ardiansyah
Bintang Alfathir Daud

NIM:
2400018052
2400018056
2400018054

Program Studi / Kelas: Informatika / A
Mata Kuliah: Pemrograman Berorientasi Objek
Tahun: 2025
Judul Proyek: Game platformers 2 dimensi berbasis Java mengungkapkan konsep pemograman berorientasi objek
Link Repository (GitHub/GitLab): [Miko-Ard/Project_PBO]
Capture Tampilan Awal / Tampilan Utama Aplikasi:



B. Persoalan Bisnis dan Deskripsi Proyek

Persoalan Bisnis :
Perkembangan industri hiburan digital, khususnya game 2D sederhana, semakin pesat dan menjadi salah satu media hiburan yang banyak diminati oleh berbagai kalangan, termasuk pelajar dan mahasiswa. Namun, terdapat beberapa persoalan bisnis yang sering muncul, antara lain:

1. **Minimnya Game Edukatif yang Sederhana dan Ringan**

   Banyak game yang tersedia saat ini membutuhkan spesifikasi perangkat yang tinggi, sehingga tidak semua pengguna dapat mengaksesnya dengan nyaman. Hal ini menjadi kendala bagi pengguna dengan perangkat terbatas.

2. **Kurangnya Media Pembelajaran Pemrograman Berbasis Proyek Nyata**

   Dalam dunia pendidikan, pembelajaran pemrograman sering kali hanya berfokus pada teori dan contoh kecil, sehingga mahasiswa kesulitan memahami penerapan konsep Object-Oriented Programming (OOP), event handling, dan game loop dalam proyek nyata.

3. **Rendahnya Interaktivitas dan Motivasi Pengguna**

   Aplikasi atau game yang tidak memiliki sistem skor, nyawa, animasi, dan efek suara cenderung kurang menarik, sehingga pengguna cepat merasa bosan.

4. **Keterbatasan Game Lokal Berbasis Desktop Java**

   Saat ini, game desktop berbasis Java masih relatif sedikit dibandingkan platform lain, padahal Java memiliki potensi besar untuk pengembangan game 2D sederhana yang stabil dan cross-platform.

Deskripsi Proyek :
Proyek ini merupakan pengembangan sebuah game 2D platformer sederhana berbasis Java Swing yang berjudul “Mini Mario Bros”. Game ini dirancang sebagai media hiburan ringan sekaligus sarana pembelajaran pemrograman berbasis objek.
Dalam game ini, pemain mengendalikan karakter utama untuk bergerak ke kiri, kanan, dan melompat guna menghindari atau mengalahkan musuh. Sistem permainan dilengkapi dengan mekanisme skor, nyawa (lives), animasi karakter, collision detection, efek suara, serta tampilan game over dan respawn.
Proyek ini mengimplementasikan berbagai konsep penting dalam pengembangan perangkat lunak, antara lain:

1. Object-Oriented Programming (OOP)
2. Game loop menggunakan Timer
3. Event handling (keyboard dan mouse)
4. Manajemen state (player, enemy, game over)

5. Animasi sprite berbasis frame
6. Collision detection menggunakan bounding box

Tujuan Proyek

Tujuan dari proyek ini adalah:

1. Mengembangkan game desktop 2D yang ringan dan mudah dimainkan.
2. Menerapkan konsep OOP dalam pengembangan aplikasi nyata.
3. Menjadi media pembelajaran untuk memahami alur kerja game (update, render, collision).
4. Meningkatkan pengalaman pengguna melalui animasi, skor, dan efek suara.

Manfaat Proyek

Manfaat yang diharapkan dari proyek ini:

1. Sebagai media hiburan sederhana bagi pengguna.
2. Sebagai contoh implementasi game 2D berbasis Java.
3. Membantu mahasiswa memahami penerapan konsep pemrograman secara praktis.
4. Menjadi dasar pengembangan game yang lebih kompleks di masa depan.

C. Daftar Seluruh Spesifikasi Aplikasi

1. Spesifikasi Umum Aplikasi
    - Nama Aplikasi: Mini Mario Bros
    - Jenis Aplikasi: Game Desktop 2D
    - Platform: Desktop (Windows / Linux / macOS)
    - Bahasa Pemrograman: Java
    - Framework / Library: Java Swing, Java AWT
    - Tipe Game: Platformer 2D
    - Target Pengguna: Pelajar, Mahasiswa, dan Pengguna Umum
2. Spesifikasi Fungsional
    - 2.1 Kontrol Pemain
        - Pemain dapat bergerak ke kiri menggunakan tombol A atau ←
        - Pemain dapat bergerak ke kanan menggunakan tombol D atau →
        - Pemain dapat melompat menggunakan tombol W, ↑, atau Space
        - Pemain tidak dapat melompat saat berada di udara
    - 2.2 Mekanisme Permainan
        - Pemain memiliki 3 nyawa (lives)
        - Pemain akan kehilangan nyawa jika menyentuh musuh dari samping
        - Pemain dapat mengalahkan musuh dengan melompat dari atas
        - Musuh akan respawn setelah dikalahkan
        - Game berakhir ketika nyawa pemain habis (Game Over)
    - 2.3 Sistem Skor
        - Skor bertambah setiap kali pemain mengalahkan musuh
        - Sistem menghitung jumlah musuh yang dikalahkan (kill count)
        - Skor dan kill count ditampilkan secara real-time

- Skor akan di-reset saat game dimulai ulang

2.4 Animasi

- Animasi player:
    1. Idle
    2. Run
    3. Jump
    4. Fall
    5. Hit
- Animasi musuh:
    1. Idle
    2. Run
    3. Die
- Animasi berbasis sprite sheet dan frame

2.5 Efek Visual
- Background statis
- Tanah (ground) dengan sprite khusus
- Floating text saat pemain mengalahkan musuh
- Tampilan Game Over dengan tombol restart
- Indikator nyawa menggunakan ikon hati

2.6 Efek Suara
- Efek suara saat pemain melompat
- Efek suara saat musuh dikalahkan
- Manajemen audio menggunakan Java Sound API

2.7 Input dan Event Handling
- Input keyboard menggunakan KeyListener
- Input mouse untuk tombol restart menggunakan MouseListener
- Game loop menggunakan Timer

3. Spesifikasi Non-Fungsional

3.1 Performa
- Game berjalan pada kecepatan tetap menggunakan interval Timer 20 ms
- Aplikasi ringan dan tidak membutuhkan spesifikasi tinggi
- Tidak memerlukan koneksi internet

3.2 Keamanan dan Stabilitas
- Validasi input keyboard
- Penanganan error saat load asset
- Fallback font jika font utama gagal dimuat

3.3 Usability
- Kontrol sederhana dan mudah dipahami
- Tampilan visual jelas dan informatif
- Feedback visual dan audio saat interaksi

4. Spesifikasi Teknis

4.1 Resolusi dan Tampilan
- Resolusi default: 1920 × 1020
- Mendukung resize window
- Rendering menggunakan paintComponent (Graphics g)

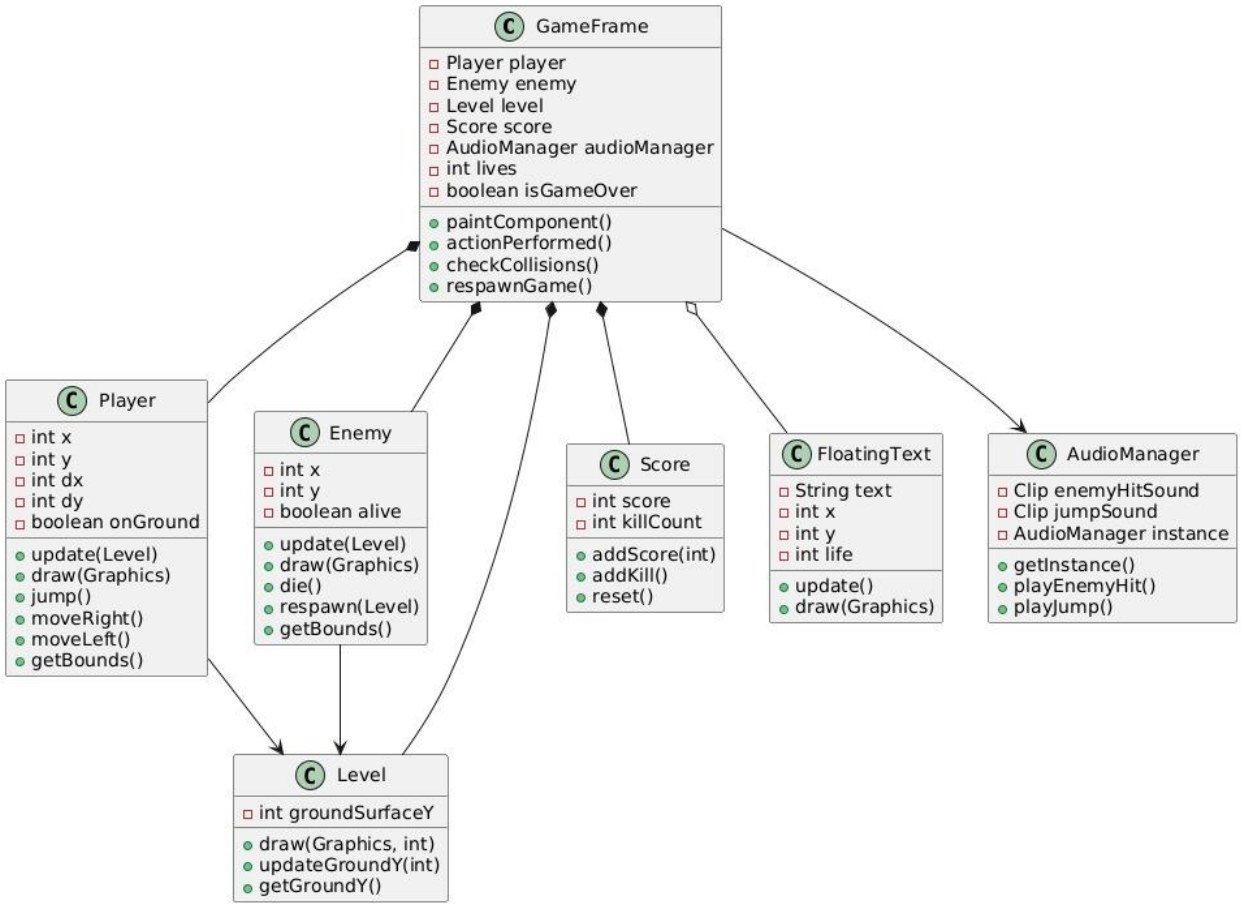
4.2 Struktur Kelas
- GameFrame → Mengatur game loop, input, dan rendering
- Player → Mengatur pergerakan dan animasi pemain
- Enemy → Mengatur musuh dan animasinya
- Level → Mengatur posisi dan tampilan tanah
- Score → Mengatur skor dan kill count
- FloatingText → Menampilkan teks animasi
- AudioManager → Mengatur efek suara

5. Spesifikasi Asset
- Format gambar: PNG

- Format audio: WAV
- Font: TrueType Font (TTF)
- Asset disimpan dalam folder /assets

6. Spesifikasi Pengembangan
   - Paradigma pemrograman: Object-Oriented Programming (OOP)
   - Menggunakan konsep:
     1. Encapsulation
     2. Abstraction
     3. Enum
     4. Event-driven programming
   - Mudah dikembangkan untuk fitur lanjutan (level baru, boss, item)
7. Spesifikasi Keterbatasan
   - Belum mendukung multiplayer
   - Belum menggunakan physics engine khusus
   - Belum ada penyimpanan skor permanen (database / file)

D. Rancangan Model Diagram UML



E. Rancangan Antar Muka Berbasis GUI

1. Gambaran Umum Antarmuka

Aplikasi Mini Mario Bros menggunakan antarmuka berbasis Graphical User Interface (GUI) yang dikembangkan dengan Java Swing. Antarmuka dirancang secara sederhana namun interaktif, dengan fokus pada kemudahan penggunaan dan pengalaman bermain yang nyaman.

Seluruh elemen antarmuka ditampilkan dalam satu jendela utama (JFrame) yang berisi panel permainan (JPanel) sebagai area utama interaksi pengguna.

2. Struktur Antarmuka Utama

2.1 Jendela Utama (JFrame)

- Judul jendela: Mini Mario Bros
- Ukuran awal: 1920 × 1020
- Dapat di-resize
- Menjadi container utama untuk seluruh elemen GUI

2.2 Panel Permainan (Game Panel / JPanel)

Panel permainan merupakan area utama tempat seluruh komponen game direnderer, meliputi:

- Background permainan
- Karakter pemain
- Musuh
- Tanah (ground)
- Floating text
- Skor dan nyawa
- Tampilan Game Over

Panel ini menggunakan metode paintComponent(Graphics g) untuk melakukan rendering.

3. Komponen GUI Utama

3.1 Background

- Background ditampilkan memenuhi seluruh area panel
- Berfungsi sebagai latar visual permainan
- Menggunakan gambar statis format PNG

3.2 Karakter Pemain (Player)

- Ditampilkan sebagai sprite animasi
- Memiliki beberapa state visual:
    1. Idle
    2. Run
    3. Jump
    4. Fall
    5. Hit
- Posisi berubah berdasarkan input keyboard

3.3 Musuh (Enemy)

- Ditampilkan sebagai sprite animasi

- Bergerak secara otomatis ke kiri dan kanan
- Memiliki animasi mati (die)

3.4 Tanah (Ground)
- Ditampilkan di bagian bawah layar
- Menjadi batas jatuh karakter
- Digambar menggunakan sprite khusus

3.5 Informasi Skor
- Ditampilkan di pojok kiri atas layar
- Menampilkan:
    1. Total skor
    2. Jumlah musuh yang dikalahkan (kill count)
- Menggunakan font khusus (Pixel Font)

3.6 Indikator Nyawa
- Ditampilkan menggunakan ikon hati
- Setiap ikon mewakili satu nyawa
- Berkurang saat pemain terkena musuh

3.7 Floating Text
- Teks animasi yang muncul saat musuh dikalahkan
- Bergerak ke atas dan memudar
- Digunakan sebagai feedback visual kepada pemain

4. Antarmuka Game Over
4.1 Tampilan Game Over
- Muncul saat nyawa pemain habis
- Ditampilkan di tengah layar
- Menggunakan frame visual khusus

4.2 Elemen Game Over
- Teks GAME OVER
- Informasi skor akhir
- Tombol Restart

4.3 Tombol Restart
- Dapat diklik menggunakan mouse
- Mengulang permainan dari awal
- Mengatur ulang skor, nyawa, dan posisi karakter

5. Interaksi Pengguna
5.1 Input Keyboard
- Mengontrol pergerakan dan aksi pemain
- Input langsung memengaruhi animasi dan posisi karakter

5.2 Input Mouse

- Digunakan untuk menekan tombol restart saat Game Over

6. Alur Interaksi Antarmuka
Aplikasi dijalankan
JFrame tampil
Panel permainan dirender
Player & Enemy muncul
Skor dan nyawa ditampilkan
Input keyboard aktif

Pemain bermain
GUI update setiap frame
Animasi berjalan
Feedback visual & audio muncul

Nyawa habis
Tampilan Game Over muncul
Pemain klik Restart
Game dimulai ulang

7. Prinsip Desain Antarmuka

- Sederhana dan intuitif
- Responsif terhadap input
- Feedback visual yang jelas
- Konsisten dalam penggunaan warna dan font
- Mudah dikembangkan untuk fitur tambahan

8. Keterbatasan Antarmuka

- Belum menggunakan menu utama terpisah
- Belum mendukung pengaturan (settings)
- Belum mendukung fullscreen otomatis

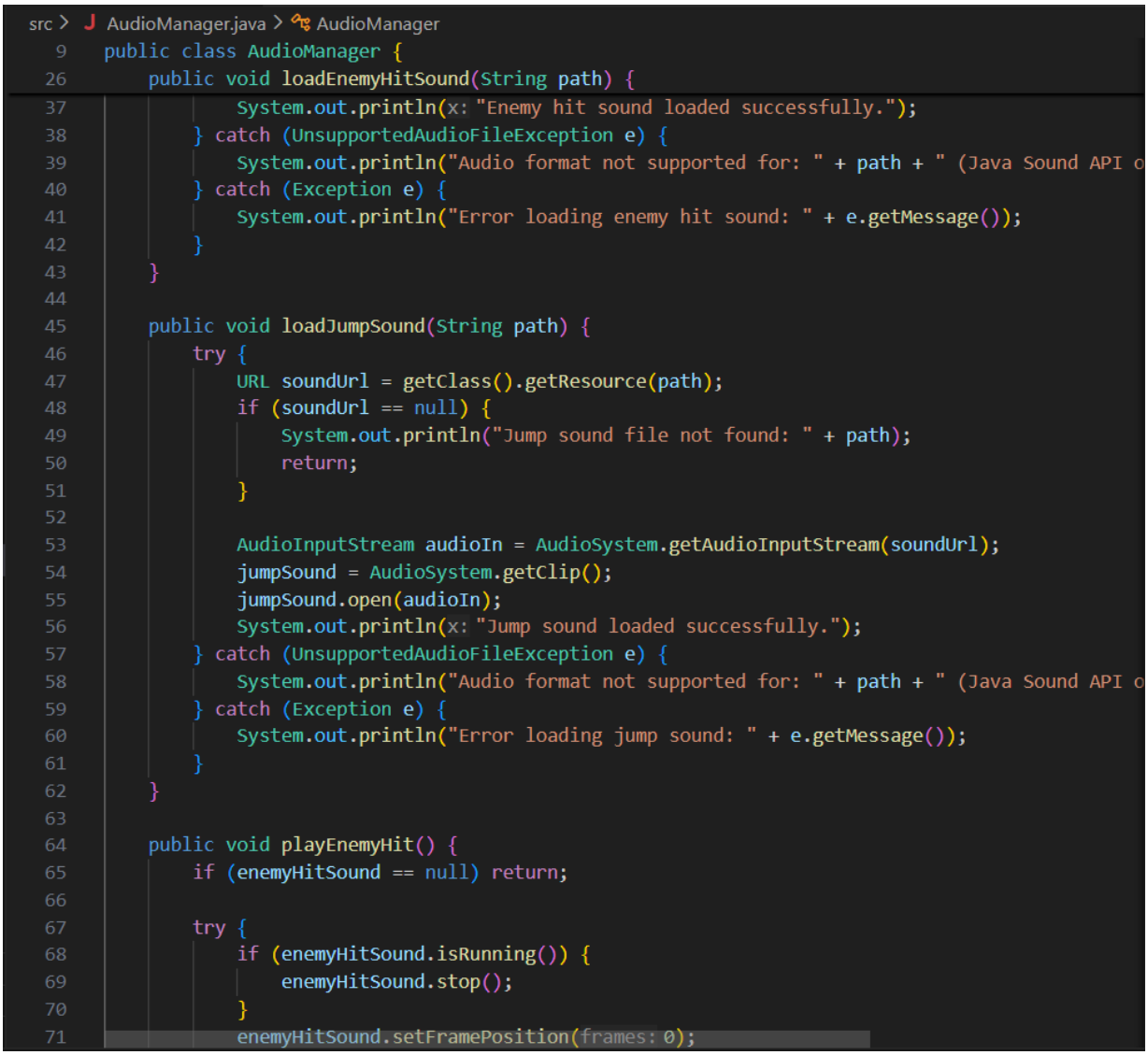
F. Skrip Program dan Penjelasannya

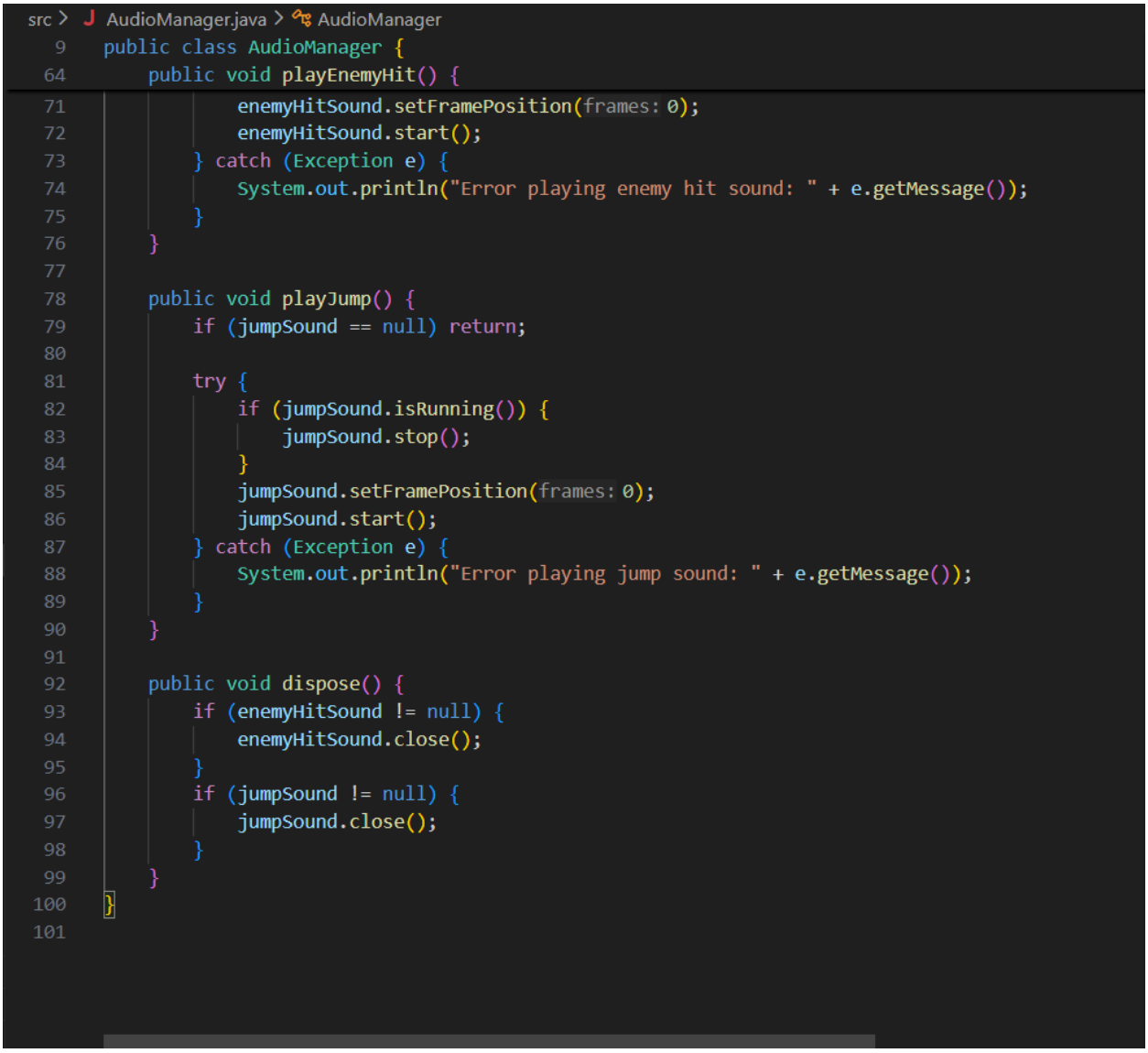
Class audioManager
Laptop file buka di bawah

AudioManager.java

src > AudioManager.java > AudioManager

```java
import javax.sound.sampled.*;
import java.io.File;
import java.net.URL;

/**
 * Manages all audio playback for the game.
 * Uses Java Sound API for audio playback.
 */
public class AudioManager {
    private Clip enemyHitSound;
    private Clip jumpSound;

    private static AudioManager instance;

    private AudioManager() {
        // Private constructor for Singleton pattern.
    }

    public static AudioManager getInstance() {
        if (instance == null) {
            instance = new AudioManager();
        }
        return instance;
    }

    public void loadEnemyHitSound(String path) {
        try {
            URL soundUrl = getClass().getResource(path);
            if (soundUrl == null) {
                System.out.println("Enemy hit sound file not found: " + path);
                return;
            }

            AudioInputStream audioIn = AudioSystem.getAudioInputStream(soundUrl);
            enemyHitSound = AudioSystem.getClip();
            enemyHitSound.open(audioIn);
            System.out.println(x: "Enemy hit sound loaded successfully.");
```


src > AudioManager.java > AudioManager
9 public class AudioManager {
26 public void loadEnemyHitSound(String path) {
37 System.out.println(x: "Enemy hit sound loaded successfully.");
38 } catch (UnsupportedAudioFileException e) {
39 System.out.println("Audio format not supported for: " + path + " (Java Sound API o
40 } catch (Exception e) {
41 System.out.println("Error loading enemy hit sound: " + e.getMessage());
42 }
43 }
44
45 public void loadJumpSound(String path) {
46 try {
47 URL soundUrl = getClass().getResource(path);
48 if (soundUrl == null) {
49 System.out.println("Jump sound file not found: " + path);
50 return;
51 }
52
53 AudioInputStream audioIn = AudioSystem.getAudioInputStream(soundUrl);
54 jumpSound = AudioSystem.getClip();
55 jumpSound.open(audioIn);
56 System.out.println(x: "Jump sound loaded successfully.");
57 } catch (UnsupportedAudioFileException e) {
58 System.out.println("Audio format not supported for: " + path + " (Java Sound API o
59 } catch (Exception e) {
60 System.out.println("Error loading jump sound: " + e.getMessage());
61 }
62 }
63
64 public void playEnemyHit() {
65 if (enemyHitSound == null) return;
66
67 try {
68 if (enemyHitSound.isRunning()) {
69 enemyHitSound.stop();
70 }
71 enemyHitSound.setFramePosition(frames: 0);



```
src > AudioManager.java > AudioManager
  9  public class AudioManager {
 64      public void playEnemyHit() {
 71              enemyHitSound.setFramePosition(frames: 0);
 72              enemyHitSound.start();
 73          } catch (Exception e) {
 74              System.out.println("Error playing enemy hit sound: " + e.getMessage());
 75          }
 76      }
 77
 78      public void playJump() {
 79          if (jumpSound == null) return;
 80
 81          try {
 82              if (jumpSound.isRunning()) {
 83                  jumpSound.stop();
 84              }
 85              jumpSound.setFramePosition(frames: 0);
 86              jumpSound.start();
 87          } catch (Exception e) {
 88              System.out.println("Error playing jump sound: " + e.getMessage());
 89          }
 90      }
 91
 92      public void dispose() {
 93          if (enemyHitSound != null) {
 94              enemyHitSound.close();
 95          }
 96          if (jumpSound != null) {
 97              jumpSound.close();
 98          }
 99      }
100  }
101
```

AudioManager.java pejelasan yang ada dalam sintaks file diatas

- Singleton : cuma 1 instance AudioManager di game.
- Clip enemyHitSound & jumpSound : simpan suara efek.
- loadEnemyHitSound(path) / loadJumpSound(path) : muat file audio.
- playEnemyHit() / playJump() : mainkan suara, mulai dari awal.
- dispose() : tutup clip, bersihkan memori.
- Error handling : file tidak ada atau format salah tetap aman.

Class Player

Buka file di laptop

Player.java

src > Player.java > ...

```java
import javax.swing.*;
import java.awt.*;
import java.net.URL;

/**
 * Represents the player character, handling its state, movement, and animation.
 */
public class Player {

    private enum AnimationState {
        IDLE, RUN, JUMP, FALL, HIT
    }

    private int x, y;
    private final int width = 60, height = 60;
    private int dx = 0, dy = 0;
    private boolean onGround = false;
    private boolean facingRight = true;

    private AnimationState currentState = AnimationState.IDLE;
    private int hitTimer = 0;

    private Image idleSheet, runSheet, jumpSheet, fallSheet, hitSheet;

    private final int[] frameCounts = {11, 12, 1, 1, 7}; // Corresponds to AnimationState enum
    private final int[] animSpeeds = {2, 2, 1, 1, 1};

    private int frameIndex = 0;
    private int animCounter = 0;

    public Player(int x, int y) {
        this.x = x;
        this.y = y;
        loadSpriteSheets();
    }

    private void loadSpriteSheets() {
```

src > Player.java > ...

```java
public class Player {
    private void loadSpriteSheets() {
        try {
            URL idleUrl = getClass().getResource(name: "/assets/idle.png");
            if (idleUrl != null) idleSheet = new ImageIcon(idleUrl).getImage();

            URL runUrl = getClass().getResource(name: "/assets/run.png");
            if (runUrl != null) runSheet = new ImageIcon(runUrl).getImage();

            URL jumpUrl = getClass().getResource(name: "/assets/Jump (32x32).png");
            if (jumpUrl != null) jumpSheet = new ImageIcon(jumpUrl).getImage();

            URL fallUrl = getClass().getResource(name: "/assets/fall.png");
            if (fallUrl != null) fallSheet = new ImageIcon(fallUrl).getImage();

            URL hitUrl = getClass().getResource(name: "/assets/hit.png");
            if (hitUrl != null) hitSheet = new ImageIcon(hitUrl).getImage();
        } catch (Exception e) {
            System.err.println(x: "Failed to load player sprite sheets.");
            e.printStackTrace();
        }
    }

    public void update(Level level) {
        x += dx;
        dy += 1; // Gravity
        y += dy;

        if (y >= level.getGroundY() - height) {
            y = level.getGroundY() - height;
            dy = 0;
            onGround = true;
        } else {
            onGround = false;
        }

        if (hitTimer > 0) {
```

```
72          if (hitTimer > 0) {
73              hitTimer--;
74          }
75
76          updateState();
77          updateAnimation();
78      }
79
80      private void updateState() {
81          if (hitTimer > 0) {
82              currentState = AnimationState.HIT;
83          } else if (!onGround) {
84              currentState = (dy < 0) ? AnimationState.JUMP : AnimationState.FALL;
85          } else {
86              currentState = (dx != 0) ? AnimationState.RUN : AnimationState.IDLE;
87          }
88      }
89
90      private void updateAnimation() {
91          animCounter++;
92          int stateIndex = currentState.ordinal();
93          if (animCounter >= animSpeeds[stateIndex]) {
94              animCounter = 0;
95              frameIndex = (frameIndex + 1) % frameCounts[stateIndex];
96          }
97      }
98
99      public void draw(Graphics g) {
100         Image currentSheet = getCurrentImageSheet();
101         if (currentSheet == null) return;
102
103         int frameCount = frameCounts[currentState.ordinal()];
104         int frameW = currentSheet.getWidth(observer: null) / frameCount;
105         int sx1 = frameIndex * frameW;
106         int sy1 = 0;
```

src > Player.java > ...

```
 8  public class Player {
99      public void draw(Graphics g) {
106         int sy1 = 0;
107         int sx2 = sx1 + frameW;
108         int sy2 = currentSheet.getHeight(observer: null);
109
110         if (facingRight) {
111             g.drawImage(currentSheet, x, y, x + width, y + height, sx1, sy1, sx2, sy2, observe
112         } else {
113             g.drawImage(currentSheet, x + width, y, x, y + height, sx1, sy1, sx2, sy2, observe
114         }
115     }
116
117     private Image getCurrentImageSheet() {
118         switch (currentState) {
119             case IDLE: return idleSheet;
120             case RUN: return runSheet;
121             case JUMP: return jumpSheet;
122             case FALL: return fallSheet;
123             case HIT: return hitSheet;
124             default: return null;
125         }
126     }
127
128     public void moveRight() {
129         dx = 5;
130         facingRight = true;
131     }
132
133     public void moveLeft() {
134         dx = -5;
135         facingRight = false;
136     }
137
138     public void stop() {
139         dx = 0;
140     }
```

```
src > Player.java > ...
  8  public class Player {
138      public void stop() {
140      }
141
142      public void jump() {
143          if (onGround) {
144              dy = -15;
145              onGround = false;
146          }
147      }
148
149      public void bounce() {
150          dy = -10;
151      }
152
153      public void takeHit() {
154          if (hitTimer <= 0) { // Prevent taking hits while already in hit state
155              hitTimer = 30; // Hit animation duration
156          }
157      }
158
159      public boolean isOnGround() { return onGround; }
160      public Rectangle getBounds() { return new Rectangle(x, y, width, height); }
161      public int getDY() { return dy; }
162  }
163
```

Player.java pejelasan yang ada dalam sintaks file diatas

- AnimationState enum : IDLE, RUN, JUMP, FALL, HIT
- Variabel posisi & kecepatan : x, y, dx, dy, ukuran width/height
- Status : onGround, facingRight, currentState, hitTimer
- Gambar sprite : idleSheet, runSheet, jumpSheet, fallSheet, hitSheet
- Frame & animasi : frameCounts, animSpeeds, frameIndex, animCounter
- Constructor : set posisi awal, load sprite sheets
- loadSpriteSheets() : muat gambar sprite dari folder /assets/
- update(Level level) : update posisi, gravitasi, deteksi tanah, hit timer, state, dan animasi
- updateState() : tentukan state berdasarkan dx/dy, onGround, hitTimer
- updateAnimation() : pindah frame animasi sesuai kecepatan animasi
- draw(Graphics g) : gambar frame animasi saat ini, flip jika menghadap kiri
- getCurrentImageSheet() : ambil sprite sheet sesuai state
- Kontrol pemain
- moveRight() / moveLeft() : geser posisi dan arah hadap
- stop() : berhenti horizontal
- jump() : lompat kalau di tanah
- bounce() : lompat kecil (misal setelah musuh kena)
- takeHit() : masuk state HIT selama beberapa frame
- Getter : isOnGround(), getBounds(), getDY()

Classs GameFrame
Buka file di laptop

GameFrame.java

src > GameFrame.java > ...

```java
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
import java.io.InputStream;
import java.net.URL;
import java.util.ArrayList;
import java.util.Iterator;

/**
 * The main panel for the game, handling the game loop, rendering, and input.
 */
public class GameFrame extends JPanel implements ActionListener, KeyListener {

    private Image background;
    private Timer timer;
    private Player player;
    private Enemy enemy;
    private Level level;
    private Score score;

    private Image heartImage;
    private int lives = 3;

    private Font gameFont;
    private Font gameFontBold;

    private Image frameImage;
    private Image buttonImage;
    private Rectangle buttonBounds;

    private boolean isGameOver = false;
    private boolean enemyRespawning = false;

    private ArrayList<FloatingText> floatingTexts = new ArrayList<>();
    private AudioManager audioManager;

    public static final int WIDTH = 1920;
```

src > GameFrame.java > ...

```
12  public class GameFrame extends JPanel implements ActionListener, KeyListener {
37      public static final int WIDTH = 1920;
38      public static final int HEIGHT = 1020;
39
40      public GameFrame() {
41          setLayout(mgr: null);
42
43          loadAssets();
44
45          audioManager = AudioManager.getInstance();
46          audioManager.loadEnemyHitSound(path: "/assets/enemy-hit.wav");
47          audioManager.loadJumpSound(path: "/assets/jump-se.wav");
48
49          level = new Level();
50          level.updateGroundY(HEIGHT);
51
52          int groundY = level.getGroundY();
53          player = new Player(x: 100, groundY - 60);
54          enemy = new Enemy(x: 600, groundY - 60);
55          score = new Score();
56
57          timer = new Timer(delay: 20, this);
58          timer.start();
59
60          addMouseListener(new MouseAdapter() {
61              @Override
62              public void mouseClicked(MouseEvent e) {
63                  if (isGameOver && buttonBounds != null && buttonBounds.contains(e.getPoint()))
64                      respawnGame();
65                  }
66              }
67          });
68
69          setFocusable(focusable: true);
70          addKeyListener(this);
71      }
72
```

src > GameFrame.java > ...

```
12  public class GameFrame extends JPanel implements ActionListener, KeyListener {
72
73      private void loadAssets() {
74          try {
75              URL bgUrl = getClass().getResource(name: "/assets/bg.png");
76              if (bgUrl != null) {
77                  background = new ImageIcon(bgUrl).getImage();
78              }
79
80              URL heartUrl = getClass().getResource(name: "/assets/hearth.png");
81              if (heartUrl != null) {
82                  heartImage = new ImageIcon(heartUrl).getImage();
83              }
84
85              URL frameUrl = getClass().getResource(name: "/assets/frame.png");
86              if (frameUrl != null) {
87                  frameImage = new ImageIcon(frameUrl).getImage();
88              }
89
90              URL buttonUrl = getClass().getResource(name: "/assets/button.png");
91              if (buttonUrl != null) {
92                  buttonImage = new ImageIcon(buttonUrl).getImage();
93              }
94
95              InputStream is = getClass().getResourceAsStream(name: "/assets/PixelifySans-Medium
96              if (is != null) {
97                  Font baseFont = Font.createFont(Font.TRUETYPE_FONT, is);
98                  gameFont = baseFont.deriveFont(size: 18f);
99                  gameFontBold = baseFont.deriveFont(Font.BOLD, size: 32f);
100                 is.close();
101             } else {
102                 gameFont = new Font(name: "Arial", Font.PLAIN, size: 18);
103                 gameFontBold = new Font(name: "Arial", Font.BOLD, size: 32);
104             }
105         } catch (Exception e) {
106             System.err.println(x: "Failed to load one or more assets. Using fallback.");
107             e.printStackTrace();
```

src > GameFrame.java > ...

```
12  public class GameFrame extends JPanel implements ActionListener, KeyListener {
73      private void loadAssets() {
108             // Use default font as a fallback
109             gameFont = new Font(name: "Arial", Font.PLAIN, size: 18);
110             gameFontBold = new Font(name: "Arial", Font.BOLD, size: 32);
111         }
112     }
113
114     @Override
115     protected void paintComponent(Graphics g) {
116         super.paintComponent(g);
117
118         if (background != null) {
119             g.drawImage(background, x: 0, y: 0, getWidth(), getHeight(), this);
120         } else {
121             g.setColor(Color.CYAN);
122             g.fillRect(x: 0, y: 0, getWidth(), getHeight());
123         }
124
125         level.draw(g, getWidth());
126         player.draw(g);
127         enemy.draw(g);
128
129         for (FloatingText text : floatingTexts) {
130             text.draw(g);
131         }
132
133         g.setColor(Color.BLACK);
134         g.setFont(gameFont);
135         g.drawString("Score: " + score.getScore(), x: 20, y: 30);
136
137         if (heartImage != null) {
138             int heartSize = 32;
139             for (int i = 0; i < lives; i++) {
140                 g.drawImage(heartImage, 20 + i * (heartSize + 8), y: 40, heartSize, heartSize,
141             }
142         }
```

src > GameFrame.java > ...

```
12  public class GameFrame extends JPanel implements ActionListener, KeyListener {
115     protected void paintComponent(Graphics g) {
142         }
143
144         if (isGameOver) {
145             drawGameOver((Graphics2D) g);
146         }
147     }
148
149     private void drawGameOver(Graphics2D g2) {
150         int panelWidth = getWidth();
151         int panelHeight = getHeight();
152
153         int frameWidth = 650;
154         int frameHeight = 550;
155         int frameX = (panelWidth - frameWidth) / 2;
156         int frameY = (panelHeight - frameHeight) / 2;
157
158         if (frameImage != null) {
159             g2.drawImage(frameImage, frameX, frameY, frameWidth, frameHeight, observer: null)
160         }
161
162         g2.setFont(gameFontBold.deriveFont(size: 56f));
163         FontMetrics fmGameOver = g2.getFontMetrics();
164         String gameOverText = "GAME OVER";
165         int gameOverX = frameX + (frameWidth - fmGameOver.stringWidth(gameOverText)) / 2;
166         int gameOverY = frameY + 100;
167
168         g2.setColor(Color.BLACK);
169         g2.drawString(gameOverText, gameOverX + 2, gameOverY + 2);
170         g2.setColor(Color.RED);
171         g2.drawString(gameOverText, gameOverX, gameOverY);
172
173         g2.setFont(gameFont.deriveFont(size: 28f));
174         FontMetrics fmScore = g2.getFontMetrics();
175         String scoreText = "Score: " + score.getScore();
176         int scoreX = frameX + (frameWidth - fmScore.stringWidth(scoreText)) / 2;
```

src > GameFrame.java > ...

```
12  public class GameFrame extends JPanel implements ActionListener, KeyListener {
149     private void drawGameOver(Graphics2D g2) {
177         int scoreY = gameOverY + 100;
178
179         g2.setColor(Color.BLACK);
180         g2.drawString(scoreText, scoreX + 2, scoreY + 2);
181         g2.setColor(Color.WHITE);
182         g2.drawString(scoreText, scoreX, scoreY);
183
184         if (buttonImage != null) {
185             int buttonWidth = 180;
186             int buttonHeight = 70;
187             int buttonX = frameX + (frameWidth - buttonWidth) / 2;
188             int buttonY = scoreY + 70;
189
190             buttonBounds = new Rectangle(buttonX, buttonY, buttonWidth, buttonHeight);
191             g2.drawImage(buttonImage, buttonX, buttonY, buttonWidth, buttonHeight, observer: nu
192         }
193     }
194
195     @Override
196     public void actionPerformed(ActionEvent e) {
197         if (isGameOver) return;
198
199         level.updateGroundY(getHeight());
200         player.update(level);
201         enemy.update(level);
202
203         Iterator<FloatingText> textIterator = floatingTexts.iterator();
204         while (textIterator.hasNext()) {
205             FloatingText text = textIterator.next();
206             text.update();
207             if (!text.isAlive()) {
208                 textIterator.remove();
209             }
210         }
211
```

src > GameFrame.java > ...

```
12  public class GameFrame extends JPanel implements ActionListener, KeyListener {
196     public void actionPerformed(ActionEvent e) {
211
212         checkCollisions();
213         repaint();
214     }
215
216     private void checkCollisions() {
217         if (!enemy.isAlive() || !player.getBounds().intersects(enemy.getBounds())) {
218             return;
219         }
220
221         if (player.getDY() > 0 && player.getBounds().y + player.getBounds().height < enemy.get
222             enemy.die();
223             score.addScore(value: 100);
224             score.addKill();
225             player.bounce();
226             audioManager.playEnemyHit();
227
228             String killText = "+" + score.getKillCount() + " Kill";
229             floatingTexts.add(new FloatingText(killText, enemy.getBounds().x, enemy.getBounds(
230
231             if (!enemyRespawning) {
232                 enemyRespawning = true;
233                 Timer respawnTimer = new Timer(delay: 1000, ev -> {
234                     enemy.respawn(level);
235                     enemyRespawning = false;
236                 });
237                 respawnTimer.setRepeats(flag: false);
238                 respawnTimer.start();
239             }
240         } else {
241             player.takeHit();
242             loseLife();
243         }
244     }
245
```

src > GameFrame.java > ...

```
12  public class GameFrame extends JPanel implements ActionListener, KeyListener {
245
246     private void loseLife() {
247         lives--;
248         if (lives <= 0) {
249             lives = 0;
250             gameOver();
251         } else {
252             enemy.respawn(level);
253         }
254     }
255
256     private void gameOver() {
257         isGameOver = true;
258         timer.stop();
259     }
260
261     private void respawnGame() {
262         lives = 3;
263         isGameOver = false;
264
265         int groundY = level.getGroundY();
266         player = new Player(x: 100, groundY - 60);
267         enemy.respawn(level);
268         score.reset();
269         floatingTexts.clear();
270
271         enemyRespawning = false;
272         buttonBounds = null;
273         timer.start();
274     }
275
276     @Override
277     public void keyPressed(KeyEvent e) {
278         int code = e.getKeyCode();
279
280         if (code == KeyEvent.VK_RIGHT || code == KeyEvent.VK_D) {
```

```
src > GameFrame.java > ...
 12  public class GameFrame extends JPanel implements ActionListener, KeyListener {
277      public void keyPressed(KeyEvent e) {
280          if (code == KeyEvent.VK_RIGHT || code == KeyEvent.VK_D) {
281              player.moveRight();
282          } else if (code == KeyEvent.VK_LEFT || code == KeyEvent.VK_A) {
283              player.moveLeft();
284          } else if (code == KeyEvent.VK_SPACE || code == KeyEvent.VK_W || code == KeyEvent.VK_U
285              if (player.isOnGround()) {
286                  player.jump();
287                  audioManager.playJump();
288              }
289          }
290      }
291
292      @Override
293      public void keyReleased(KeyEvent e) {
294          int code = e.getKeyCode();
295          if (code == KeyEvent.VK_RIGHT || code == KeyEvent.VK_D ||
296              code == KeyEvent.VK_LEFT || code == KeyEvent.VK_A) {
297              player.stop();
298          }
299      }
300
301      @Override
302      public void keyTyped(KeyEvent e) {}
303
         Run | Debug
304      public static void main(String[] args) {
305          SwingUtilities.invokeLater(() -> {
306              JFrame frame = new JFrame(title: "Mini Mario Bros");
307              GameFrame game = new GameFrame();
308              frame.add(game);
309              frame.setSize(WIDTH, HEIGHT);
310              frame.setLocationRelativeTo(c: null);
311              frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
312              frame.setResizable(resizable: true);
313              frame.setVisible(b: true);
310              frame.setLocationRelativeTo(c: null);
311              frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
312              frame.setResizable(resizable: true);
313              frame.setVisible(b: true);
314          });
315      }
316  }
```

Inheritance & Interfaces : extends JPanel, implements ActionListener, KeyListener
Variabel game :

- background, frameImage, buttonImage, heartImage : gambar & UI
- Player player, Enemy enemy, Level level, Score score : objek game
- lives : nyawa pemain
- floatingTexts : teks efek (+Kill, dll.)

- AudioManager audioManager : play sound efek
- timer : game loop
- isGameOver, enemyRespawning : status game

Ukuran game : WIDTH=1920, HEIGHT=1020
Constructor GameFrame()

- Load assets (gambar, font)
- Inisialisasi AudioManager dan load suara
- Inisialisasi level, player, enemy, score
- Setup timer loop (20 ms)
- Mouse listener : klik tombol respawn saat game over
- Key listener : kontrol player

loadAssets() : muat semua gambar & font, fallback pakai default kalau gagal
paintComponent(Graphics g) :

- Gambar background, level, player, enemy
- Gambar floatingTexts
- Gambar score & nyawa (hearts)
- Jika isGameOver : panggil drawGameOver()

drawGameOver(Graphics2D g2):
Gambar frame game over, teks "GAME OVER", skor, dan tombol respawn
actionPerformed(ActionEvent e) :

- Update level, player, enemy
- Update floatingTexts dan hapus yang mati
- Cek collision
- Repaint panel

checkCollisions() :

- Jika player menabrak enemy dari atas : kill enemy, add score, bounce player, play sound, floatingText + respawn enemy
- Jika tidak : player takeHit, kurangi nyawa (loseLife())

loseLife() :

- lives--
- Jika 0 : gameOver()
- Jika masih ada nyawa : respawn enemy

gameOver() : stop timer, set isGameOver = true
respawnGame() : reset lives, reset player/enemy/score/floatingTexts, start timer
Key controls :

- moveRight() / moveLeft() : geser player
- stop() : berhenti horizontal
- jump() : lompat, play sound jump

Main method :

- Buat JFrame, tambahkan GameFrame, set ukuran & visible

Main method :

- Buat JFrame, tambahkan GameFrame, set ukuran & visible

Class level
Buka link dari laptop


Level.java

src > Level.java > ...

```java
import javax.swing.*;
import java.awt.*;
import java.net.URL;

/**
 * Manages the game's ground level.
 */
public class Level {
    private int groundSurfaceY;
    private final int groundHeight = 250;
    private final int surfaceThickness = 85;
    private Image groundImage;

    public Level() {
        try {
            URL groundUrl = getClass().getResource(name: "/assets/Ground.png");
            if (groundUrl != null) {
                groundImage = new ImageIcon(groundUrl).getImage();
            }
        } catch (Exception e) {
            System.err.println(x: "Failed to load ground image.");
            groundImage = null;
        }
    }

    /**
     * Draws the ground, stretching to the panel's width.
     */
    public void draw(Graphics g, int panelWidth) {
        if (groundImage == null) return;

        int yDraw = groundSurfaceY - (groundHeight - surfaceThickness);
        g.drawImage(groundImage, x: 0, yDraw, panelWidth, groundHeight, observer: null);
    }

    public int getGroundY() {
        return groundSurfaceY;
    }

    public int getGroundY() {
        return groundSurfaceY;
    }

    /**
     * Updates the ground's Y position based on the panel's height.
     */
    public void updateGroundY(int panelHeight) {
        groundSurfaceY = panelHeight - surfaceThickness;
    }
}
```

Variabel :

- groundSurfaceY : posisi permukaan tanah
- groundHeight : tinggi tanah
- surfaceThickness : ketebalan permukaan tanah
- groundImage : gambar tanah

Constructor Level() :

- Load gambar tanah dari /assets/Ground.png

- Jika gagal : groundImage = null

draw(Graphics g, int panelWidth) :

- Gambar tanah dengan lebar panel
- Y posisi digeser sesuai tinggi & ketebalan permukaan

getGroundY() :

- Mengembalikan posisi permukaan tanah

updateGroundY(int panelHeight) :

- Update groundSurfaceY berdasarkan tinggi panel

Class Enemy
Buka file di laptop


Enemy.java

src > Enemy.java > ...

```java
import javax.swing.*;
import java.awt.*;
import java.awt.geom.AffineTransform;
import java.net.URL;

public class Enemy {

    private enum AnimationState {
        IDLE, RUN, DIE
    }

    private int x, y;
    private int width = 120, height = 120;
    private int hitboxWidth = 60, hitboxHeight = 60;
    private int dx = 2;
    private int dy = 0;
    private boolean alive = true;
    private boolean facingRight = true;

    private AnimationState currentState = AnimationState.IDLE;
    private int dieAnimTimer = 0;

    private Image idleSheet, runSheet, dieSheet;

    private final int idleFrameCount = 8;
    private final int runFrameCount = 8;
    private final int dieFrameCount = 15;

    private final int idleAnimSpeed = 8;
    private final int runAnimSpeed = 2;
    private final int dieAnimSpeed = 3;

    private int frameIndex = 0;
    private int animCounter = 0;

    public Enemy(int x, int y) {
        this.x = x;
```

src > Enemy.java > ...

```
6   public class Enemy {
36      public Enemy(int x, int y) {
38          this.y = y;
39          loadSpriteSheets();
40      }
41
42      private void loadSpriteSheets() {
43          try {
44              URL idleUrl = getClass().getResource(name: "/assets/Mushroom-Idle.png");
45              if (idleUrl != null) idleSheet = new ImageIcon(idleUrl).getImage();
46
47              URL runUrl = getClass().getResource(name: "/assets/Mushroom-Run.png");
48              if (runUrl != null) runSheet = new ImageIcon(runUrl).getImage();
49
50              URL dieUrl = getClass().getResource(name: "/assets/Mushroom-Die.png");
51              if (dieUrl != null) dieSheet = new ImageIcon(dieUrl).getImage();
52          } catch (Exception e) {
53              System.err.println(x: "Failed to load enemy sprite sheets.");
54              e.printStackTrace();
55          }
56      }
57
58      public void update(Level level) {
59          if (!alive) {
60              if (dieAnimTimer > 0) {
61                  dieAnimTimer--;
62                  updateAnimation();
63              }
64              return;
65          }
66
67          x += dx;
68          if (x <= 0 || x + width >= GameFrame.WIDTH) {
69              dx *= -1;
70          }
71          facingRight = (dx > 0);
```

src > Enemy.java > ...

```
6   public class Enemy {
58      public void update(Level level) {
73          dy += 1; // Gravity
74          y += dy;
75
76          int groundY = level.getGroundY();
77          if (y >= groundY - height) {
78              y = groundY - height;
79              dy = 0;
80          }
81
82          currentState = (dx != 0) ? AnimationState.RUN : AnimationState.IDLE;
83          updateAnimation();
84      }
85
86      private void updateAnimation() {
87          animCounter++;
88          int currentAnimSpeed = 1;
89          int currentFrameCount = 1;
90
91          switch (currentState) {
92              case IDLE:
93                  currentAnimSpeed = idleAnimSpeed;
94                  currentFrameCount = idleFrameCount;
95                  break;
96              case RUN:
97                  currentAnimSpeed = runAnimSpeed;
98                  currentFrameCount = runFrameCount;
99                  break;
100             case DIE:
101                 currentAnimSpeed = dieAnimSpeed;
102                 currentFrameCount = dieFrameCount;
103                 break;
104         }
105
106         if (animCounter >= currentAnimSpeed) {
107             animCounter = 0;
```

```
src > Enemy.java > ...
  6  public class Enemy {
 86      private void updateAnimation() {
107              animCounter = 0;
108              if (currentState == AnimationState.DIE) {
109                  if (frameIndex < currentFrameCount - 1) {
110                      frameIndex++;
111                  }
112              } else {
113                  frameIndex = (frameIndex + 1) % currentFrameCount;
114              }
115          }
116      }
117
118      public void draw(Graphics g) {
119          Image currentSheet = getCurrentImageSheet();
120          int currentFrameCount = getCurrentFrameCount();
121
122          if (currentSheet == null || currentSheet.getWidth(observer: null) <= 0) {
123              return;
124          }
125
126          int frameW = currentSheet.getWidth(observer: null) / currentFrameCount;
127          int sx1 = frameIndex * frameW;
128
129          Graphics2D g2d = (Graphics2D) g.create();
130          if (facingRight) {
131              g2d.drawImage(currentSheet, x, y, x + width, y + height, sx1, sy1: 0, sx1 + frameW
132          } else {
133              g2d.drawImage(currentSheet, x + width, y, x, y + height, sx1, sy1: 0, sx1 + frameW
134          }
135          g2d.dispose();
136      }
137
138      private Image getCurrentImageSheet() {
139          if (!alive && dieAnimTimer > 0) {
140              return dieSheet;
141          }
```

src > Enemy.java > ...

```
  6  public class Enemy {
138      private Image getCurrentImageSheet() {
141          }
142          if (alive) {
143              switch (currentState) {
144                  case IDLE: return idleSheet;
145                  case RUN: return runSheet;
146                  default: return idleSheet;
147              }
148          }
149          return null;
150      }
151
152      private int getCurrentFrameCount() {
153          if (!alive && dieAnimTimer > 0) {
154              return dieFrameCount;
155          }
156          if (alive) {
157              switch (currentState) {
158                  case IDLE: return idleFrameCount;
159                  case RUN: return runFrameCount;
160                  default: return idleFrameCount;
161              }
162          }
163          return 1;
164      }
165
166      public Rectangle getBounds() {
167          int hitboxX = x + (width - hitboxWidth) / 2;
168          int hitboxY = y + (height - hitboxHeight); // Align hitbox with bottom
169          return new Rectangle(hitboxX, hitboxY, hitboxWidth, hitboxHeight);
170      }
171
172      public boolean isAlive() {
173          return alive;
174      }
175
```

```
src > Enemy.java > ...
  6  public class Enemy {
170      }
171
172      public boolean isAlive() {
173          return alive;
174      }
175
176      public void die() {
177          if (!alive) return;
178          alive = false;
179          currentState = AnimationState.DIE;
180          dieAnimTimer = dieFrameCount * dieAnimSpeed; // Animation duration
181          frameIndex = 0;
182          animCounter = 0;
183      }
184
185      public void respawn(Level level) {
186          alive = true;
187          currentState = AnimationState.IDLE;
188          dieAnimTimer = 0;
189          frameIndex = 0;
190
191          int maxX = GameFrame.WIDTH - width;
192          x = (int) (Math.random() * (maxX > 0 ? maxX : 0));
193
194          y = level.getGroundY() - height;
195          dy = 0;
196
197          dx = (Math.random() > 0.5) ? 2 : -2;
198      }
199  }
200
```

AnimationState enum : IDLE, RUN, DIE
Variabel posisi & ukuran : x, y, width, height, hitboxWidth, hitboxHeight
Variabel gerak & status : dx, dy, alive, facingRight
Animasi : currentState, dieAnimTimer, frameIndex, animCounter
Sprite sheets : idleSheet, runSheet, dieSheet
Frame & animasi speed : idleFrameCount/Speed, runFrameCount/Speed, dieFrameCount/Speed
Constructor Enemy(x, y) :

- Set posisi awal
- Load sprite sheets

loadSpriteSheets() :

- Muat gambar idle, run, die dari /assets/

update(Level level) :

- Jika mati : jalankan animasi DIE sampai selesai
- Jika hidup : update posisi horizontal (patah balik di tepi layar)
- Update arah hadap (facingRight)
- Terapkan gravitasi (dy)
- Check tanah (y >= ground)

- Set state IDLE / RUN sesuai dx
- Update animasi

updateAnimation() :

- Tambah animCounter
- Update frameIndex sesuai speed & jumlah frame
- DIE : frameIndex berhenti di akhir animasi

draw(Graphics g) :

- Ambil sheet saat ini (getCurrentImageSheet())
- Hitung frame width
- Draw frame, flip horizontal jika menghadap kiri

getCurrentImageSheet() : ambil sprite sheet sesuai state & status alive
getCurrentFrameCount() : ambil jumlah frame sesuai state & alive
getBounds() :

- Hitbox rectangle untuk collision detection

isAlive() : return status alive
die() :

- Set alive = false
- Set state DIE
- Set dieAnimTimer = durasi animasi
- Reset frameIndex & animCounter

respawn(Level level) :

- Set alive = true, state = IDLE, reset animasi
- Set x random di dalam layar
- Set y di atas ground
- Set dx random arah kiri/kanan

Classs Floating
Buka file dari laptop

FloatingText.java

src > FloatingText.java > FloatingText > update()

```java
import java.awt.*;

public class FloatingText {
    private String text;
    private int x, y;
    private int life;
    private final int maxLife;
    private Color color;
    private Font font;

    public FloatingText(String text, int x, int y, int duration, Color color, Font font) {
        this.text = text;
        this.x = x;
        this.y = y;
        this.maxLife = duration;
        this.life = duration;
        this.color = color;
        this.font = font;
    }


    public void update() {
        if (isAlive()) {
            life--;
            y -= 2;
        }
    }

    public boolean isAlive() {
        return life > 0;
    }


    public void draw(Graphics g) {
        if (!isAlive()) return;

        Graphics2D g2d = (Graphics2D) g.create();
```

```
33
34      public void draw(Graphics g) {
35          if (!isAlive()) return;
36
37          Graphics2D g2d = (Graphics2D) g.create();
38
39          float alpha = (float) life / maxLife;
40          g2d.setComposite(AlphaComposite.getInstance(AlphaComposite.SRC_OVER, Math.max(a: 0, al
41
42          g2d.setFont(font);
43          FontMetrics fm = g2d.getFontMetrics();
44          int textWidth = fm.stringWidth(text);
45          int drawX = x - textWidth / 2;
46
47          // Draw outline
48          g2d.setColor(Color.BLACK);
49          g2d.drawString(text, drawX + 1, y + 1);
50          g2d.drawString(text, drawX - 1, y - 1);
51          g2d.drawString(text, drawX + 1, y - 1);
52          g2d.drawString(text, drawX - 1, y + 1);
53
54          // Draw main text
55          g2d.setColor(color);
56          g2d.drawString(text, drawX, y);
57
58          g2d.dispose();
59      }
60  }
61
```

Variabel :

- text : teks yang ditampilkan
- x, y : posisi teks di layar
- life : sisa waktu teks tampil
- maxLife : durasi teks
- color : warna teks
- font : font teks

Constructor FloatingText(text, x, y, duration, color, font) :

- Inisialisasi teks, posisi, durasi, warna, font

update() :

- Kurangi life tiap frame
- Geser teks ke atas (y -= 2)

isAlive() :

- Return true jika life > 0

draw(Graphics g) :

- Hentikan jika !isAlive()
- Gunakan Graphics2D untuk alpha/fade-out
- Hitung alpha : life / maxLife
- Draw outline hitam di sekitar teks

- Draw teks utama dengan warna color
- Dispose Graphics2D setelah selesai

Cllass Score
Buka file dari laptop


Score.java

```java
public class Score {

    private int score = 0;
    private int killCount = 0;

    public void addScore(int value) {
        if (value > 0) {
            score += value;
        }
    }

    public int getScore() {
        return score;
    }

    public void addKill() {
        killCount++;
    }

    public int getKillCount() {
        return killCount;
    }

    public void reset() {
        score = 0;
        killCount = 0;
    }
}
```

Variabel :

- score : skor pemain
- killCount : jumlah musuh yang dikalahkan

addScore(int value) :

- Tambahkan nilai ke score jika value > 0

getScore() :

- Mengembalikan skor saat ini

addKill() :

- Tambahkan 1 ke killCount

getKillCount() :

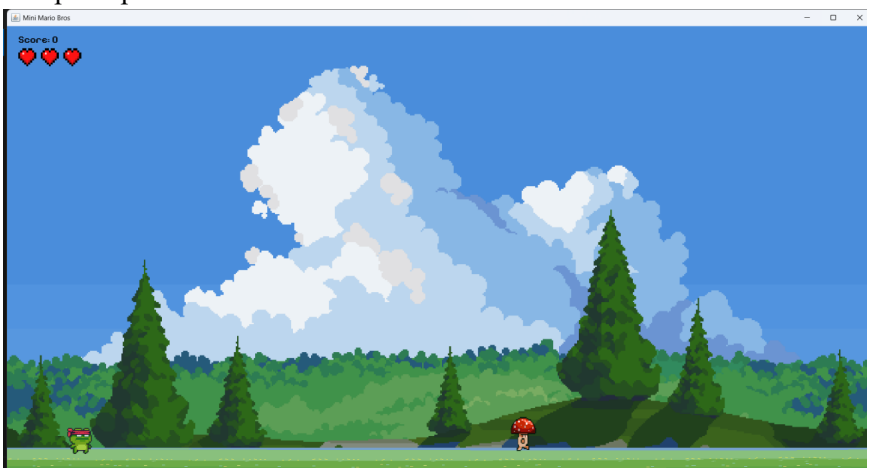
- Mengembalikan jumlah kill saat ini

reset() :

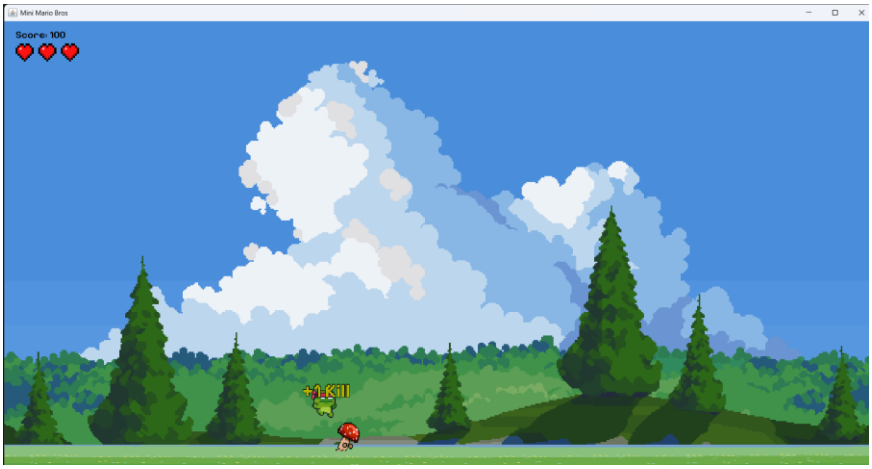
- Reset score dan killCount ke 0

G. Penjelasan Screenshot Tampilan yang Dihasilkan

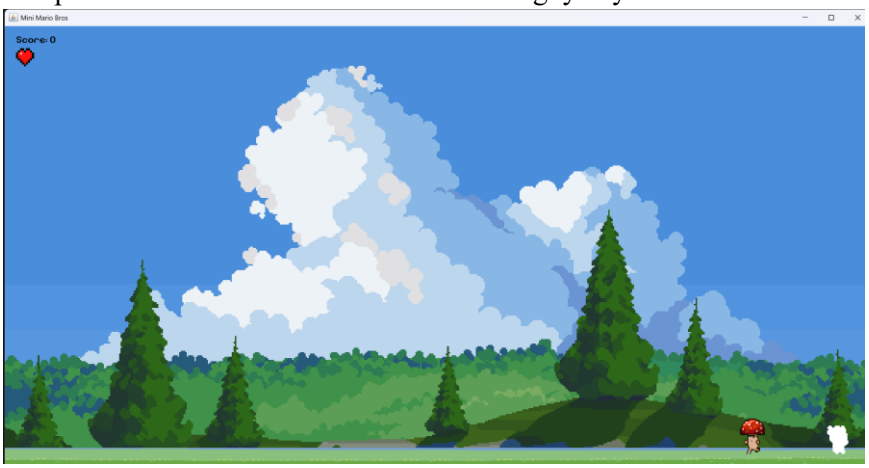
Tampilan pertama ketika run code



Tampilan ketika kill musuh



Tampilan ketika terkena musuh dan berkurangnya nyawa

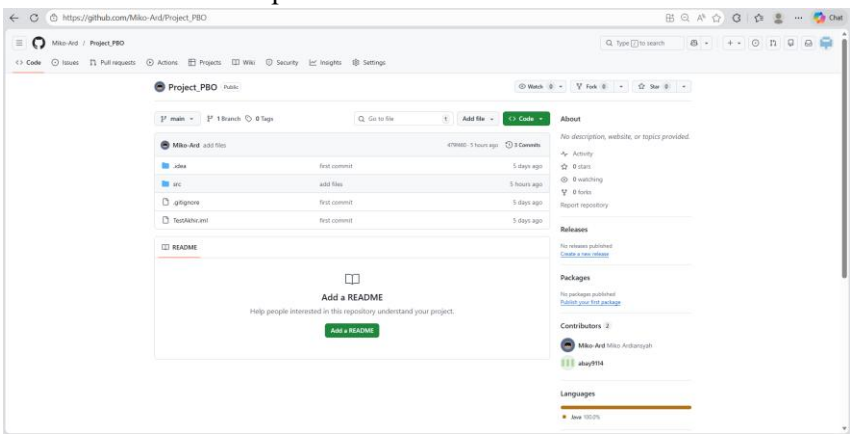


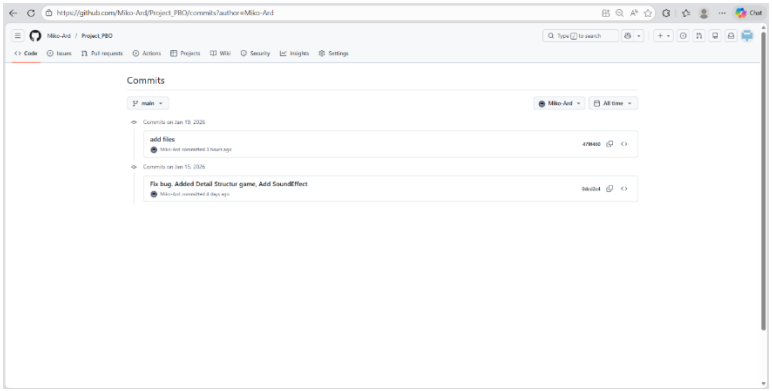
Tampilan ketika game over dan tampilan tombol play untuk bermain ulang

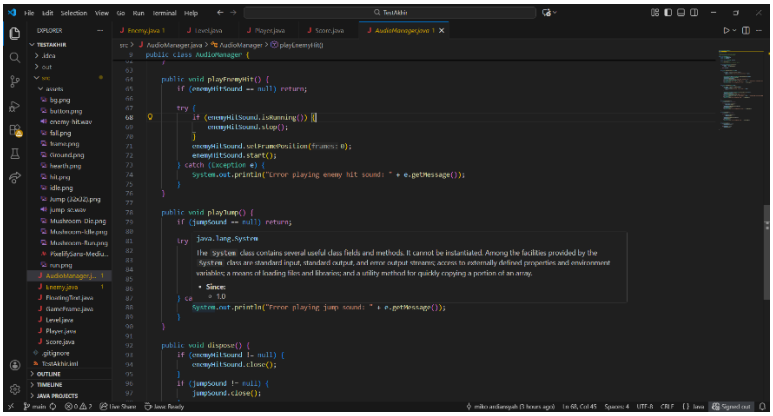
H. Penjelasan Screenshot Status Unggah GitHub/GitLab

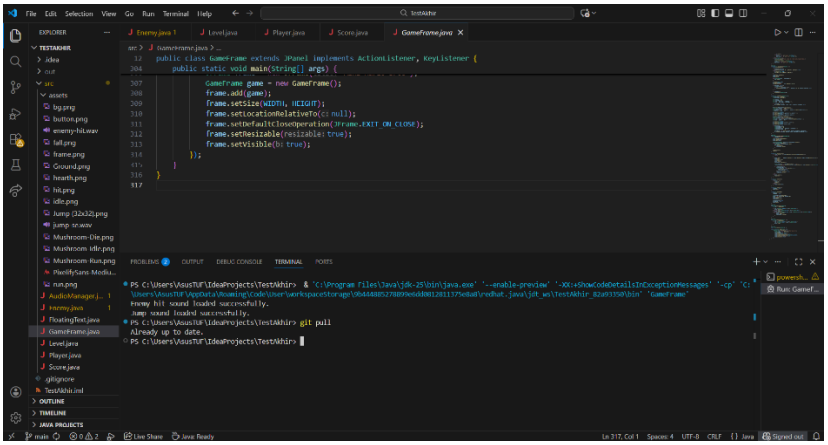
1. **File/Folder Project Terbaru**
    - Semua file yang sudah dibuat dan dimodifikasi berhasil diunggah.
    - Menunjukkan struktur project di repository sama dengan lokal.
2. **Commit Terakhir**
    - Terdapat pesan commit sesuai yang dibuat saat push.
    - Membuktikan perubahan terbaru berhasil dicatat di repository.
3. **Tanggal dan Waktu Commit (Opsional)**
    - Memberikan bukti kapan project terakhir diunggah.

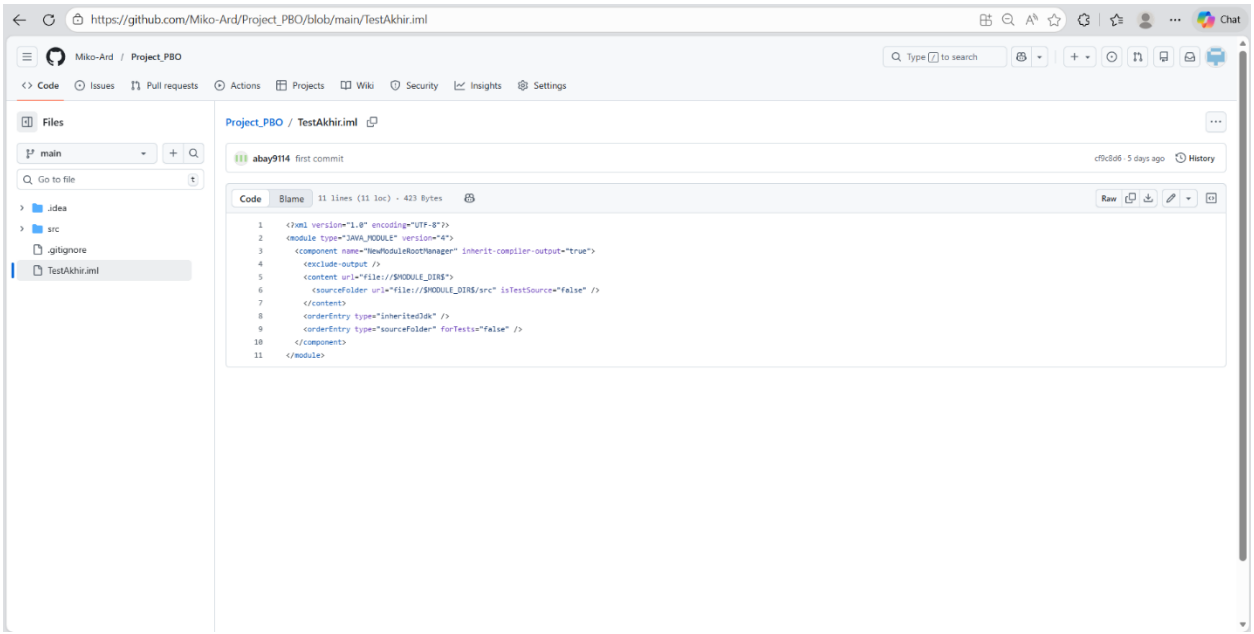
Dengan screenshot ini, dapat dipastikan bahwa project telah berhasil diunggah ke repository GitHub/GitLab dan siap untuk diakses atau dinilai.




Commits
add files
Fix bug, Added Detail Structur game, Add SoundEffect


AudioManager.java
public class AudioManager {
public void playEnemyHit() {
public void playJump() {
public void dispose() {


File Edit Selection View Go Run Terminal Help
TESTAKHIR
Enemy.java 1
Level.java
Player.java
Score.java
GameFrame.java
public class GameFrame extends JPanel implements ActionListener, KeyListener {
public static void main(String[] args) {
GameFrame game = new GameFrame();
frame.add(game);
frame.setSize(WIDTH, HEIGHT);
frame.setLocationRelativeTo(null);
frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
frame.setResizable(true);
frame.setVisible(true);
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
PS C:\Users\AsusTUF\IdeaProjects\TestAkhir> & 'C:\Program Files\Java\jdk-25\bin\java.exe' '--enable-preview' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp'
Enemy hit sound loaded successfully.
Jump sound loaded successfully.
PS C:\Users\AsusTUF\IdeaProjects\TestAkhir> git pull
Already up to date.
PS C:\Users\AsusTUF\IdeaProjects\TestAkhir>


https://github.com/Miko-Ard/Project_PBO/blob/main/TestAkhir.iml
Chat
Miko-Ard / Project_PBO
Code
Issues
Pull requests
Actions
Projects
Wiki
Security
Insights
Settings
Files
main
Go to file
.idea
src
.gitignore
TestAkhir.iml
Project_PBO / TestAkhir.iml
abay9114 first commit
cf9c8d6 · 5 days ago
History
Code
Blame
11 lines (11 loc) · 423 Bytes
Raw
<?xml version="1.0" encoding="UTF-8"?>
<module type="JAVA_MODULE" version="4">
<component name="NewModuleRootManager" inherit-compiler-output="true">
<exclude-output />
<content url="file://$MODULE_DIR$">
<sourceFolder url="file://$MODULE_DIR$/src" isTestSource="false" />
</content>
<orderEntry type="inheritedJdk" />
<orderEntry type="sourceFolder" forTests="false" />
</component>
</module>

I. Analisis Pengerjaan Proyek

1. Waktu

- Proyek ini dikerjakan selama **1 minggu**.
- Terdapat kendala waktu karena **banyak tugas mata kuliah lain** yang juga harus dikerjakan.
- Urutan pengerjaan proyek:
    1. Membuat repository (GitHub/GitLab)
    2. Analisis game dan spesifikasi
    3. Pembuatan kode program
    4. Desain GUI dan pengujian

2. Ketercapaian Spesifikasi

- Beberapa fitur **belum berhasil diimplementasikan**, seperti:
    - Musik latar (background music)
    - Halaman play/menu awal
    - Tipe musuh yang lebih banyak
- Fitur yang **paling sukses** adalah **penataan GUI** (tampilan antarmuka, skor, indikator nyawa, game over, respawn).
- Fitur yang **paling sulit** adalah **sound effect** karena harus mencari dan mengatur aset audio yang sesuai.

3. Biaya

- Biaya yang dibutuhkan: **0 rupiah**, karena semua software dan aset menggunakan versi gratis.
- Secara keseluruhan, biaya **sesuai estimasi** awal.
- Strategi efisiensi biaya: menggunakan **asset gratis**, termasuk gambar, sprite, font, dan sound effect.

4. Kendala

- Kendala teknis:
    - Error dalam pengaturan sprite animasi dan audio
    - Sound effect sulit disinkronisasi
- Kendala non-teknis:
    - Waktu terbatas karena banyak tugas lain
    - Mencari aset gratis yang sesuai dan berkualitas
- Cara mengatasi:
    - Mencari lebih banyak aset gratis melalui internet
    - Mengatur prioritas pengerjaan fitur yang paling penting (GUI, logika game)

5. Tantangan Masa Depan

- Fitur yang dapat ditambahkan:
    - Musik latar (background music)
    - Halaman play/menu awal
    - High score (leaderboard)
    - Tipe musuh yang lebih bervariasi
    - Mode permainan tambahan (misal time attack, endless mode)

- Optimisasi lain:
    - Meningkatkan performa game agar animasi lebih smooth
    - Penambahan AI musuh lebih kompleks
- Peluang pengembangan:
    - Bisa dijadikan **proyek portofolio** untuk tugas akhir atau pameran
    - Dapat dikembangkan menjadi game kecil komersial dengan asset berlisensi dan musik original

6. Lain-lain

- **Pelajaran penting**:
    - Perencanaan awal sangat membantu (membuat repo → analisis → coding → GUI)
    - Mengatur prioritas fitur penting membuat pengerjaan tetap selesai walau terbatas waktu
    - Manajemen waktu menjadi kunci saat mengerjakan proyek dengan banyak tugas lain
- **Jika dikerjakan ulang**:
    - Akan memulai lebih awal agar bisa menambahkan semua fitur seperti musik, menu, dan tipe musuh tambahan
    - Mencari asset lebih banyak sejak awal
- **Saran untuk pengembang lain**:
    - Gunakan repository versi kontrol (Git) dari awal
    - Prioritaskan fitur inti (GUI, logika game) sebelum menambahkan fitur tambahan
    - Manfaatkan asset gratis untuk prototipe, tapi pastikan kualitasnya sesuai kebutuhan game