JOIN (Jurnal Online Informatika)
p-ISSN: 2528-1682, e-ISSN: 2527-9165
Volume 8 Number 1 | June 2023: 1-3
DOI: 10.15575/join.xxxxx.xx

Pembuatan Mini Project Sebagai Awal Perjalanan Menjadi Full Stack Development

Raka Alpiansyah[1]
Department of Informatics, UIN Sunan Gunung Djati Bandung, Indonesia

Article Info

Article history:

Received , 2023
Revised , 2023
Accepted , 2023

Keywords:

Full Stack Development
Mini project
Pembelajaran coding
Bahasa pemrograman

ABSTRACT

Full-stack development adalah bidang yang melibatkan pengembangan baik front-end (client-side) maupun back-end (server-side) dari sebuah aplikasi. Bagian aplikasi dan game adalah salah satu bidang yang menantang dan menarik bagi full-stack developer, karena membutuhkan kreativitas, logika, dan performa tinggi. Full-stack developer yang bergerak di bidang ini harus mampu mengembangkan antarmuka pengguna yang menarik dan intuitif, serta mengelola data, logika bisnis, dan fungsi-fungsi lain yang berjalan di belakang layar. Selain itu, full-stack developer juga harus memperhatikan aspek keamanan, skalabilitas, dan kompatibilitas dari aplikasi dan game yang dibuat. Mini project adalah proyek Game yang berskala kecil dan sederhana, yang mampu untuk mempelajari bahasa pemrograman, metode pengembangan suatu project, desain dan fungsionalitas project, serta testing dan debugging. Dalam laporan ini, penulis akan menjelaskan pembuatan mini project dengan menggunakan bahasa pemograman java. Penulis juga akan memberikan contoh dan sumber belajar yang dapat dijadikan referensi. Laporan ini bertujuan untuk memberikan gambaran jika ingin menjadi fullstack developer.

Corresponding Author:

Raka Alpiansyah,
Informatics Department, Faculty of Science & Technology, UIN Sunan Gunung Djati Bandung
Jl. A. H. Nasution No. 105, Cibiru, Bandung, Indonesia. 40614
Email: rm.rizki.mauln@gmail.com

1. PENDAHULUAN

Dalam era digital saat ini, peran Full Stack Developer menjadi semakin penting dalam industri teknologi. Seorang Full Stack Developer adalah seorang profesional yang mampu mengelola semua aspek pengembangan aplikasi web, mulai dari front-end (antarmuka pengguna) hingga back-end (server dan database), serta pengujian dan debugging. Dalam konteks pengembangan game dan aplikasi, peran mereka menjadi semakin kompleks dan menantang.

Pengembangan game dan aplikasi memerlukan pemahaman mendalam tentang berbagai teknologi dan bahasa pemrograman. Dari pembuatan antarmuka pengguna yang interaktif dengan HTML, CSS, dan JavaScript, hingga pengelolaan logika bisnis dan interaksi data dengan berbagai bahasa pemrograman seperti Java, Python, atau Ruby. Selain itu, pengetahuan tentang server dan bagaimana mengelolanya, serta pemahaman tentang database dan bagaimana data disimpan dan diambil, juga sangat penting.

Dalam pengembangan game, pengetahuan tentang engine game seperti Unity atau Unreal, serta pemahaman tentang fisika game dan kecerdasan buatan, menjadi sangat penting. Selain itu, kemampuan untuk melakukan pengujian unit dan debugging juga menjadi keterampilan penting yang harus dimiliki oleh seorang Full Stack Developer.

Proyek mini ini bertujuan untuk memberikan gambaran mendalam tentang peran dan keterampilan yang dibutuhkan oleh seorang Full Stack Developer dalam bidang pengembangan game dan aplikasi.

ini diharapkan dapat memberikan manfaat yang sangat berharga bagi yang membaca untuk bisa semangat dalam memulai kariernya di dunia fullstack developer.

2. METODE

Metode yang di gunakan pada pembuatan Mini Project ini terdiri dari beberapa tahapan, yaitu:

2.1 Studi literatur.

Tahap ini dilakukan untuk mengumpulkan referensi yang berkaitan dengan topik , yaitu fullstack development, mini project. Sumber-sumber yang digunakan berasal dari Youtube, dan Website. Studi literatur ini bertujuan untuk mendapatkan gambaran, konsep-konsep yang akan di buat, dan perkembangan terbaru di bidang web development.

2.2 Pengembangan mini project.

Tahap ini dilakukan untuk mulai membuat sebuah mini project. Mini project yang dibuat adalah Game sederhana. Mini project ini dibuat dengan menggunakan bahasa pemrograman JAWA serta mengikuti prinsip-prinsip tool/OOP yang cukup baik.

2.3 Evaluasi dan analisis.

Tahap ini dilakukan untuk mengevaluasi dan menganalisis proses dan hasil pengembangan mini project. Evaluasi dan analisis ini meliputi aspek-aspek seperti kelebihan, kekurangan, kesulitan, hambatan, solusi, dan saran perbaikan yang ditemukan selama pengembangan mini project. Evaluasi dan analisis ini juga bertujuan untuk mengetahui manfaat dan kekurangan dari pembuatan mini project bagi seorang pemula yang ingin menjadi profesional Full-Stack Development.

3. HASIL DAN PEMBAHASAN

3.1 Pengertian Game

Game, dalam bahasa Indonesia berarti permainan secara umum termasuk juga berbagai permainan tradisional. Lebih spesifik lagi adalah kata video game atau permainan video yang merujuk pada game yang kita kenal seperti sekarang. game adalah permainan video yang menggunakan interaksi dengan antarmuka pengguna melalui gambar.

Beberapa ahli dan pakar memiliki pendapat masing-masing mengenai pengertian game. Misalnya, Ivan C. Sibero berpendapat bahwa game adalah salah satu aplikasi yang paling banyak dipakai dan dinikmati oleh pengguna media elektronik saat ini. Fauzia A menyatakan bahwa game adalah salah satu bentuk hiburan yang dapat dijadikan sebagai penyegar pikiran dari kepenatan akibat dari padatnya aktivitas sehari-hari.

3.1.1 Bahasa Pemograman JAVA

1. Pengertian JAVA

Java adalah bahasa pemrograman yang dapat dijalankan di berbagai komputer termasuk telepon genggam1. Bahasa ini awalnya dibuat oleh James Gosling saat masih bergabung di Sun Microsystems, yang saat ini merupakan bagian dari Oracle dan dirilis tahun 19951. Java biasanya digunakan untuk pengembangan bagian back-end dari software, aplikasi Android, dan juga website.

2. Fungsi JAVA
   Bahasa pemrograman Java memiliki berbagai fungsi, antara lain:
   - Pengembangan Game: Banyak game seluler, komputer, dan video populer yang dibangun menggunakan Java.
   - Komputasi Cloud: Java menjadi bahasa pemrograman yang cocok untuk aplikasi berbasis cloud yang terdesentralisasi.
   - Big Data: Java digunakan untuk memproses data yang kompleks serta real time dalam jumlah besar.
   - Pengembangan AI (Artificial Intelligence): Java juga digunakan dalam pengembangan AI.
   - Internet of Things: Java digunakan dalam pengembangan perangkat IoT.

   Selain itu, Java juga digunakan untuk mengembangkan back-end dalam pembuatan software, website, hingga aplikasi Android2. Dalam konteks pemrograman, fungsi dalam Java adalah kode program yang dirancang untuk menyelesaikan sebuah tugas tertentu, dan merupakan bagian dari program utama.

3. Struktur Dasar JAVA

   - Deklarasi Package: Package adalah folder yang berisi sekumpulan program Java. Deklarasi package biasanya dilakukan saat membuat program atau aplikasi besar.
   - Impor Library: Pada bagian ini, kita melakukan impor library yang dibutuhkan pada program. Library adalah sekumpulan class dan fungsi yang bisa kita gunakan dalam membuat program.
   - Bagian Class: Java adalah bahasa pemrograman yang menggunakan paradigma OOP (Object Oriented Programming). Setiap program harus dibungkus di dalam class agar nanti bisa dibuat menjadi objek.
   - Method Main: Method main() atau fungsi main() merupakan blok program yang akan dieksekusi pertama kali. Ini adalah titik masuk dari program.

3.1.2 Source Code

```
/*
 * Nama : Raka Alpiansyah
 * NIM : 1237050112
 */

import java.awt.*;
import java.time.LocalDateTime;
import java.util.ArrayList;
import java.util.Iterator;

import javax.swing.*;

public class GamePanel extends JPanel implements Runnable  {

    int FPS = 60;
    int CharSize = 25 * 3;
    int screenWidth = 40 * 25;
    int screenHeight = 40 *25;
    Tombol tombolY = new Tombol();
    Thread gameThread;

    Map map = new Map(this);
```

```
    Player player = new Player(this,tombolY, CharSize, null);
    FallingMeteor fallingBox;
    ArrayList<FallingMeteor> boxes = new ArrayList<>();
    public GamePanel(){
        this.setPreferredSize(new Dimension(screenWidth, screenHeight));
        this.setBackground(Color.gray);
        this.setDoubleBuffered(true);
        this.addKeyListener(tombolY);
        this.setFocusable(true);
    }

   public void startGameThread(){
        gameThread = new Thread(this);
        gameThread.start();
    }

    public void run() {
        double GambarFPS = 1000/FPS;
        long WaktuTrakhir= System.currentTimeMillis();
        long Waktuskrg;
        long timer = 0;
        int drawCount = 0;
        double delta = 0;
        while(gameThread != null) {
            Waktuskrg = System.currentTimeMillis();
            delta += (Waktuskrg - WaktuTrakhir) / GambarFPS;
            timer += (Waktuskrg- WaktuTrakhir);
            WaktuTrakhir = Waktuskrg;

          if(delta >= 1) {
              delta--;
              drawCount++;
              update();
              repaint();
             if(timer >= 1000) {
                System.out.println("FPS: " + drawCount);
                drawCount = 0;
                timer = 0;
                LocalDateTime waktu = LocalDateTime.now();
                System.out.println(waktu);
                boxes.add(new FallingMeteor((int)(Math.random() * screenWidth), 0,
13, "./Explosion_5.png"));
                }
            }
        }
   }

    public void update(){
        player.update();
```

```
        Iterator<FallingMeteor> it = boxes.iterator();
        while (it.hasNext()) {
            FallingMeteor box = it.next();
            box.update();
            if (box.getY() > screenHeight) {
                it.remove();
            }
        }
    }

    public void paintComponent(Graphics g) {
        super.paintComponent(g);
        Graphics2D g2 = (Graphics2D)g;
        map.draw(g2);
        player.draw(g2);
        for (FallingMeteor box : boxes) {
            box.draw(g2);
        }
        g2.dispose();
    }

    public static void main(String[] args) {
        // tampilan window
        JFrame window = new JFrame(null, null);
        GamePanel gamePanel = new GamePanel();
        window.setResizable(false);
        window.setTitle("GAME Apa?");
        window.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        window.add(gamePanel);
        window.pack();
        window.setLocationRelativeTo(null);
        gamePanel.startGameThread();
        window.setVisible(true);
    }
}
```

```
import java.awt.*;
import java.awt.image.*;
import java.io.*;
import javax.imageio.*;

public class Player {

    BufferedImage up1, up2, down1, down2, left1, left2, right1, right2;
    int x, y;
    int speed;
    int sp1 = 0, sp2 = 1;
    GamePanel gp;
    Tombol tombol;
    String gerakan;
    Map Map;
    Hp hp;
    boolean isJumping = false;
    private Hp health;
    private static final int GRAVITY = 1;
    private int velocityY = 0;

    public Player(GamePanel gp, Tombol tombolY, int maxHp, Hp health) {
        this.gp = gp;
        this.tombol = tombolY;
        setDefaultValues();
        GambarChar();
        this.hp = new Hp(maxHp);
        this.health = health;
    }

    public Hp getHealth() {
        return this.health;
    }

    public void setDefaultValues(){
        x = 750;
        y = 700;
        speed = 4;
        this.gerakan = "0" ;
    }

    public void GambarChar() {
        try {
            up1 = ImageIO.read(getClass().getResourceAsStream("./idle.png"));
            up2 = ImageIO.read(getClass().getResourceAsStream("./idle.png"));
            down1 = ImageIO.read(getClass().getResourceAsStream("./idle.png"));
            down2 = ImageIO.read(getClass().getResourceAsStream("./idle.png"));
            left1 = ImageIO.read(getClass().getResourceAsStream("./kiri1.png"));
            left2 = ImageIO.read(getClass().getResourceAsStream("./kiri2.png"));
```

```
            right1 = ImageIO.read(getClass().getResourceAsStream("./kanan1.png"));
            right2 = ImageIO.read(getClass().getResourceAsStream("./kanan2.png"));
        } catch(IOException e) {
            e.printStackTrace();
        }
    }

    public void getBounds() {
        y += velocityY;
        velocityY += GRAVITY;
        if (x <= 0) {
            x = 0;
        } else if (x + gp.CharSize > gp.screenWidth) {
            x = gp.screenHeight - gp.CharSize;
        }
        if (y <= 0) {
            y = 0;
        } else if (y + gp.CharSize > gp.screenWidth) {
            y = gp.screenHeight - gp.CharSize;
            isJumping = false;
        }
    }

    public void update() {
        getBounds();
        updatePos();
    }

    private void updatePos() {
        if(tombol.TekanLoncat == true || tombol.TekanBawah == true ||
tombol.TekanKiri == true || tombol.TekanKanan == true) {
            sp1++;
            if(tombol.TekanLoncat == true && !isJumping) {
                velocityY = -15;
                gerakan = "up";
                isJumping = true;
            } else if(tombol.TekanBawah == true) {
                y += speed;
                gerakan = "down";
            } else if(tombol.TekanKiri == true) {
                x -= speed;
                gerakan = "left";
            } else if(tombol.TekanKanan == true) {
                x += speed;
                gerakan = "right";
            }
            if(sp1 > 15) {
                if(sp2 == 1) {
                    sp2  = 2;
```

```
                } else if(sp2 == 2) {
                    sp2 = 1;
                }
                sp1 = 0;
            }
        }
    }

    public void draw(Graphics2D g2) {
        BufferedImage image = null;
        hp.draw(g2, x, y - 10, gp.CharSize, 4);
        switch(gerakan) {
            case "up":
                if(sp2 == 1) {
                    image = up1;
                } else if(sp2 == 2) {
                    image = up2;
                }
                default:
            case "down":
                if(sp2 == 1) {
                    image = down1;
                } else if(sp2 == 2) {
                    image = down2;
                }
                break;
            case "left":
                if(sp2 == 1) {
                    image = left1;
                } else if(sp2 == 2) {
                    image = left2;
                }
                break;
            case "right":
                if(sp2 == 1) {
                    image = right1;
                } else if(sp2 == 2) {
                    image = right2;
                }
                break;
        }
        g2.drawImage(image, x, y, gp.CharSize, gp.CharSize, null);
    }
}
```

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

public class Tombol implements KeyListener {
    public boolean TekanLoncat, TekanBawah, TekanKanan, TekanKiri;

    @Override
    public void keyTyped(KeyEvent e) {
    }

    @Override
    public void keyPressed(KeyEvent e) {
        int code = e.getKeyCode();

        if(code == KeyEvent.VK_UP) {
            TekanLoncat = true;
        }
        if(code == KeyEvent.VK_DOWN) {
            TekanBawah = true;
        }
        if(code == KeyEvent.VK_RIGHT) {
            TekanKanan = true;
        }
        if(code == KeyEvent.VK_LEFT) {
            TekanKiri = true;
        }
    }

    @Override
    public void keyReleased(KeyEvent e) {
        int code = e.getKeyCode();

        if(code == KeyEvent.VK_UP) {
            TekanLoncat = false;
        }
        if(code == KeyEvent.VK_DOWN) {
            TekanBawah = false;
        }
        if(code == KeyEvent.VK_RIGHT) {
            TekanKanan = false;
        }
        if(code == KeyEvent.VK_LEFT) {
            TekanKiri = false;
        }
    }
}
```

p-ISSN: 2528-1682
e-ISSN: 2527-9165

```
import java.awt.*;
import java.awt.image.*;
import java.io.*;
import javax.imageio.ImageIO;

public class Map {
    BufferedImage image1, image2;
    GamePanel gp;

    public Map(GamePanel gp) {
        this.gp = gp;
        getMap(gp);
    }

    public void getMap(GamePanel gp) {
        try {
            image1 =
ImageIO.read(getClass().getResourceAsStream("./6.png"));
            image2 =
ImageIO.read(getClass().getResourceAsStream("./6.png"));
        } catch(IOException e) {
            e.printStackTrace();
        }
    }

    public void draw(Graphics2D g2) {
        g2.drawImage(image1, 0, 0, 1280, 1200, null);
    }
}
```

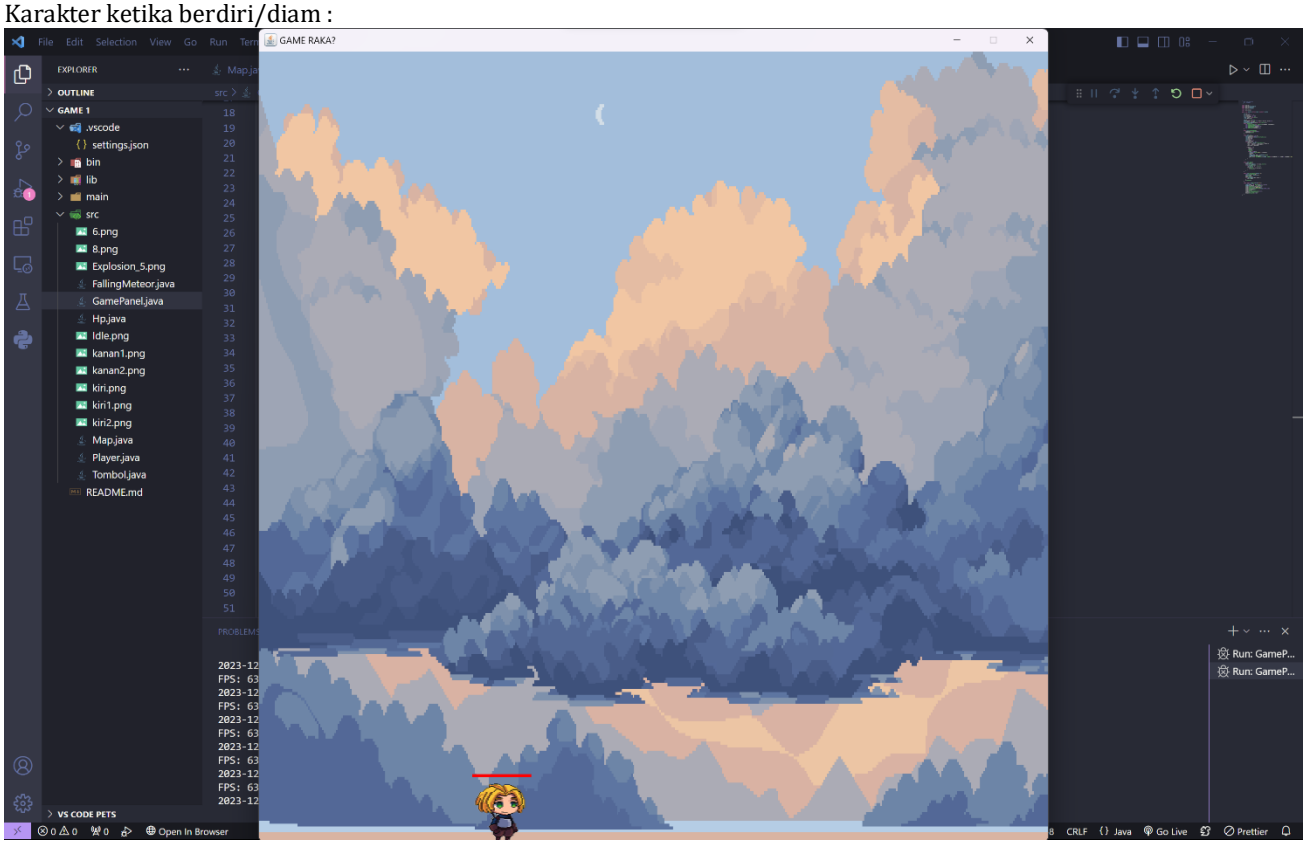
```java
import java.awt.Color;
import java.awt.Graphics;

public class Hp {
    private int maxHp;
    private int currentHp;

    public Hp(int maxHp) {
        this.maxHp = maxHp;
        this.currentHp = maxHp;
    }

    public void decreaseHp(int amount) {
        currentHp -= amount;
        if (currentHp < 0) {
            currentHp = 0;
        }
    }

    public void increaseHp(int amount) {
        currentHp += amount;
        if (currentHp > maxHp) {
            currentHp = maxHp;
        }
    }

    public void draw(Graphics g, int x, int y, int width, int height) {
        g.setColor(Color.RED);
        int healthWidth = (int) ((currentHp / (double) maxHp) * width);
        g.fillRect(x, y, healthWidth, height);
    }
}
```

p-ISSN: 2528-1682
e-ISSN: 2527-9165

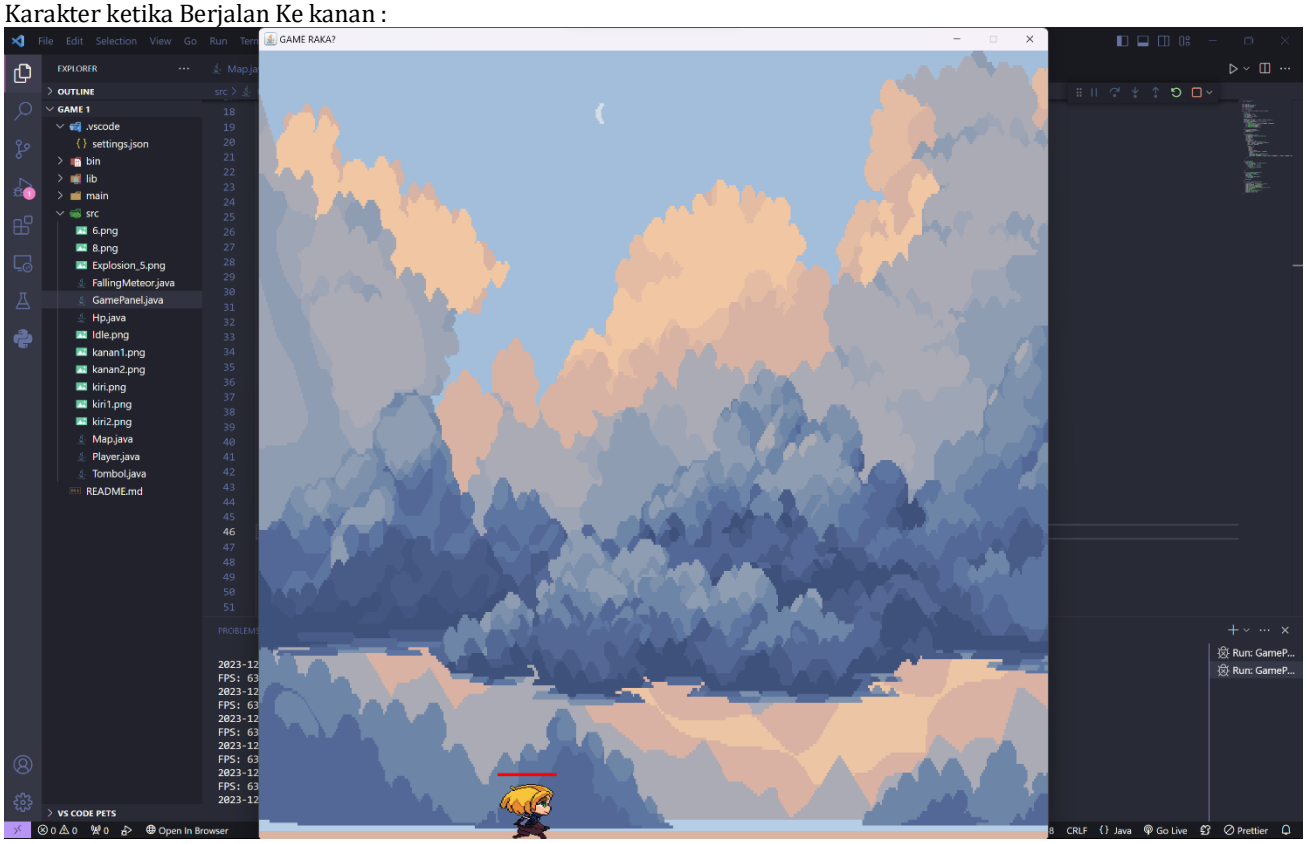
```
import java.awt.*;
import javax.imageio.ImageIO;
import java.io.IOException;

public class FallingMeteor {
    private int x, y;
    private int speed;
    private Image image1;
    public FallingMeteor(int x, int y, int speed, String imagePath) {
        this.x = x;
        this.y = y;
        this.speed = speed;
        try {
            this.image1 =
ImageIO.read(getClass().getResourceAsStream("./Explosion_5.png"));
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    public void update() {
        y += speed;
    }

    public void draw(Graphics2D g2) {
        g2.drawImage(image1, x, y,155, 155, null);
    }

    public int getY() {
        return y;
    }
}
```

Output atau Hasil :

Karakter ketika berdiri/diam :

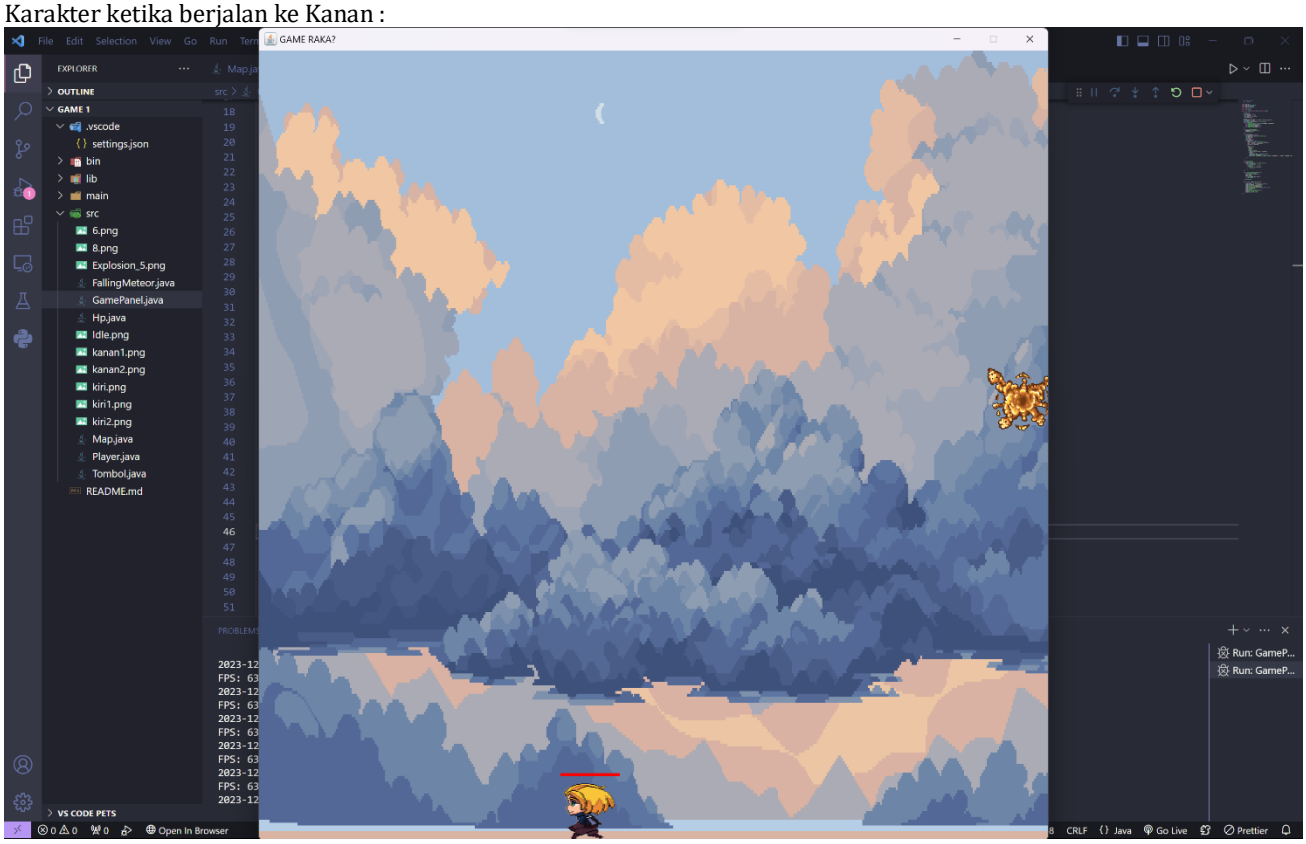


Karakter ketika Berjalan Ke kanan :

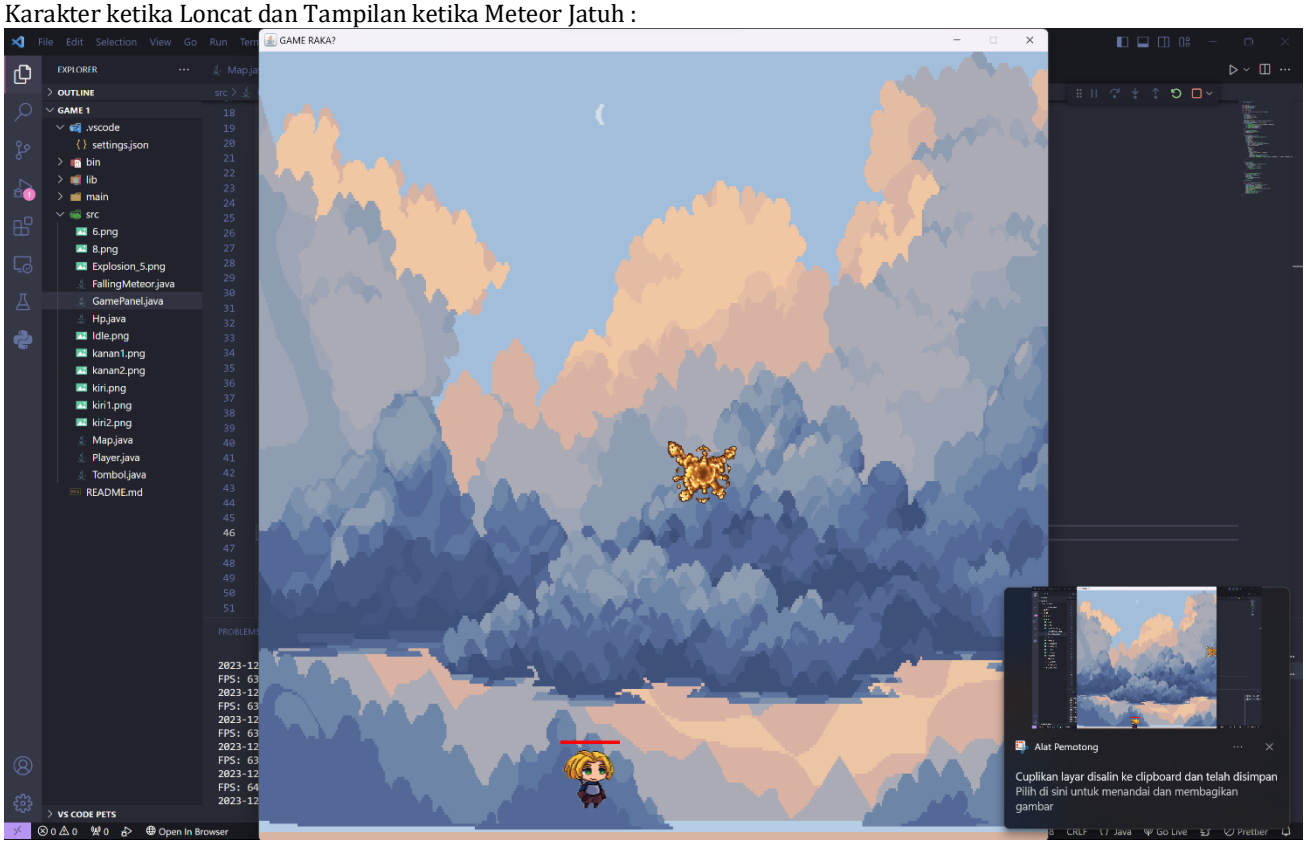


p-ISSN: 2528-1682
e-ISSN: 2527-9165

Karakter ketika berjalan ke Kanan :



Karakter ketika Loncat dan Tampilan ketika Meteor Jatuh :



1. KESIMPULAN

Full Stack Development memainkan peran penting dalam pengembangan game dan aplikasi. Seorang Full Stack Developer memiliki kemampuan untuk bekerja pada berbagai aspek aplikasi, baik itu front-end maupun back-end, yang memungkinkan pembuatan aplikasi menjadi lebih efisien dan efektif.

Full Stack Developer bertanggung jawab untuk menulis kode back-end, melakukan integrasi database, dan menulis HTML atau CSS pada bagian front-end yang terintegrasi dengan kode back-end. Mereka juga perlu memahami teknologi UI dan UX, yang sangat penting dalam pengembangan game dan aplikasi.

Dengan pengetahuan dan keterampilan ini, Full Stack Developer dapat menganalisis dan memecahkan masalah pada aplikasi dari berbagai sisi bahasa pemrograman. Selain itu, mereka juga diharapkan mampu membuat dan mengembangkan berbagai proyek dari klien atau perusahaan.

Secara keseluruhan, dalam konteks mini project Anda, Full Stack Development memungkinkan pembuatan game dan aplikasi yang lebih efisien dan efektif, dan memberikan pemahaman yang lebih baik tentang bagaimana berbagai bagian aplikasi bekerja bersama, yang dapat membantu dalam pengoptimalan dan pemecahan masalah. Ini adalah aspek penting yang perlu diperhatikan dalam pengembangan game dan aplikasi.

REFERENCES

- *https://glints.com/id/lowongan/tools-full-stack-developer/*
- *https://www.codepolitan.com/blog/contoh-referensi-aplikasi-dan-game-berbasis-web-terbaik-yang-menerapkan-pwa-58ea903475162*
- *https://idcloudhost.com/blog/panduan-lengkap-menjadi-full-stack-developer/*
- *https://www.jagoanhosting.com/blog/aplikasi-game-developer/*
- *https://www.niagahoster.co.id/blog/java-adalah/*
- *https://informatika.stei.itb.ac.id/~rinaldi.munir/AljabarGeometri/2020-2021/Pengantar-Pemrograman-dengan-Bahasa-Java-2020.pdf*
- *https://www.gramedia.com/best-seller/buku-pemograman-java/*
- *https://glints.com/id/lowongan/peran-full-stack-developer/*
- *http://www.makalah.my.id/2016/09/makalah-tentang-game-online.html*