LAPORAN TUGAS BESAR
MATA KULIAH KONSEP PEMROGRAMAN
GAMES DENGAN BAHASA C / C++

Disusun oleh :

Nama : Rifqi Makarim

NIM : L0123122

Kelas : D

PROGRAM STUDI INFORMATIKA

FAKULTAS TEKNOLOGI INFORMASI DAN SAINS DATA

UNIVERSITAS SEBELAS MARET

GAME “LUFFY JUMP!”

A. Deskripsi Games

"Luffy Jump!" adalah permainan sederhana yang menawarkan pengalaman bermain yang menyenangkan dengan elemen khas dari karakter anime populer, Monkey D. Luffy dari anime One Piece. Game ini dirancang untuk memberikan tantangan dan kesenangan melalui mekanika lompat yang unik, dan elemen visual yang menarik.

Pemain mengendalikan karakter utama, Monkey D. Luffy, yang berusaha untuk mencapai ketinggian tertinggi dengan melompat dari awan ke awan yang lain karena konon pada titik tertinggi terdapat kekuatan yang tak terkalahkan. Tantangannya adalah menghindari objek jatuh berupa bola api yang dapat menyebabkan game over dan mencapai skor tertinggi. Dengan adanya awan spesial berupa awan yang berwarna pink, pemain dapat meraih loncatan khusus yang tinggi untuk mencapai ketinggian yang lebih sulit dijangkau. Awan khusus ini hanya dapat digunakan sekali dalam permainan dan menimbulkan penambahan skor pemain yang begitu cepat.

Dalam game ini , muncul salah satu villain dari anime One piece yaitu Kaido. Dia akan muncul ketika Luffy berpijak pada awan spesial tersebut. Ada beberapa posisi awan spesial tersebut yang dijadikan jebakan oleh kaido untuk mengalahkan Luffy. Pemain yang menggerakkan luffy harus berhati-hati karena ada saat ketika awan khusus itu dipijak akan menimbulkan efek lompat tinggi dan ada yang menimbulkan gameover karena saat itu Kaido lah yang muncul.

Dengan mekanika permainan yang sederhana namun menantang, serta desain visual yang menarik, permainan ini cocok untuk semua pemain, baik penggemar anime maupun mereka yang mencari pengalaman bermain yang menyenangkan dan seru. Nikmati setiap loncatan dan tantangan yang menghadang, dan lihat sejauh mana bisa mencapai titik tertinggi!

B. Library

Game “Luffy Jump!” dibuat menggunakan bahasa pemrograman C++ dengan mengandalkan SFML dan beberapa library lainnya.

- **SFML** (Simple and Fast Multimedia Library) : Digunakan untuk membuat antarmuka grafis dan menangani input.
- **Random** : Digunakan untuk menghasilkan nilai acak
- **Time** : Digunakan untuk mengukur waktu dan mengatur kecepatan object jatuh.

C. Alur Kerja Game

1. Pertama, Game akan menampilkan layar background, judul game, dan perintah untuk menekan tombol S pada keyboard untuk memulai permainan.
2. Selanjutnya, ketika game sudah dimulai. Kendalikan karakter “Luffy” dengan tombol panah kanan dan tombol panah kiri. Selain itu, dapat juga menggunakan tombol “A” agar karakter ke posisi kiri dan tombol “D” agar karakter ke posisi kanan.
3. Hindari object Bola api yang berjatuhan dan melompatlah ke awan yang lain untuk mencapai titik tertinggi. Jika player terkena object Bola api maka menimbulkan gameover.
4. Terdapat awan special yang terletak dibawah (awan berwarna pink) yang dapat hanya digunakan sekali dalam permainan. Selanjutnya awan tersebut tidak dapat digunakan dan akan memunculkan musuh utama dari game ini. Namun, terdapat posisi dari awan special ini yang akan menimbulkan musuh datang dan menimbulkan gameover.
5. Jika terjadi gameover, maka akan menampilkan layar gameover disertai dengan skor yang didapatkan selama permainan. Ditampilkan juga tombol restart dengan menekan tombol “R” yang berguna untuk memulai kembali permainan.

D. Penjelasan Singkat File Source Code

- **Fungsi showMainMenu**

  Fungsi showMainMenu digunakan untuk menampilkan menu utama game. Fungsi ini menerima parameter window yang merupakan objek window dari SFML.

```
void showMainMenu(sf::RenderWindow& window) {
    sf::Font Start,title;
    Start.loadFromFile("font/RESEARCH AND DEVELOPMENT.otf");
    title.loadFromFile("font/Joorick.ttf");
```

Mendefinisikan dua jenis font yang digunakan untuk judul dan teks pada menu utama

```
    // Declare and load the texture
    sf::Texture startBackgroundTexture;
    startBackgroundTexture.loadFromFile("images/bg-start.jpg");
    sf::Sprite startBackgroundSprite(startBackgroundTexture);
    startBackgroundSprite.setPosition(0, 0);

    sf::Text titleText;
    titleText.setFont(title);
    titleText.setString("Luffy Jump!");
    titleText.setCharacterSize(60);
    titleText.setFillColor(sf::Color::Red);
    titleText.setPosition(100, 200);

    sf::Text startText;
    startText.setFont(Start);
    startText.setString("Press 'S' to Start");
    startText.setCharacterSize(20);
    startText.setFillColor(sf::Color::White);
    startText.setPosition(100, 300);
```

- Membuat object Texture kemudian dimasukkan gambar latar belakang dan membuat object sprite dari gambar latar belakang tersebut.

- Membuat objek teks untuk judul game.
- Membuat objek teks untuk instruksi memulai game.

```
    window.clear(); //membersihkan window
    window.draw(startBackgroundSprite);
    window.draw(titleText);
    window.draw(startText);
    window.display(); // menampilkan object yang sebelumnya di-draw

    // Menekan Tombol "S" untuk memulai game
    while (true) {
        sf::Event event;
        while (window.pollEvent(event)) {
            if (event.type == sf::Event::Closed) {
                window.close();
                exit(0);
            }
            if (event.type == sf::Event::KeyPressed) {
                if (event.key.code == sf::Keyboard::S) {
                    return;
                }
            }
        }
    }
}
```

- Membersihkan window dan menampilkan objek sprite dan teks pada window.
- Pemain menekan tombol ‘S’ untuk memulai game. Jika pemain menekan tombol ‘S’, fungsi akan selesai dan kembali ke fungsi pemanggil

Class SpecialPlatform

```
class SpecialPlatform {
public:
    sf::Vector2f position;
    sf::Texture texture;
    sf::Sprite sprite;
    float jumpHeight;
    bool isUsed; // Menandakan apakah special platform sudah digunakan
```

- Mendefinisikan atribut dari kelas SpecialPlatform.

```
    SpecialPlatform(sf::RenderWindow& window, float height)
        : jumpHeight(height), isUsed(false) {
        texture.loadFromFile("images/special_platform.png");
        sprite.setTexture(texture);

        // Set position at a random location along the X-axis
        std::uniform_int_distribution<unsigned> x(0, window.getSize().x - texture.getSize().x);
        std::default_random_engine e(time(0));
        position.x = static_cast<float>(x(e));

        // Set position above the regular platforms
        position.y = 600.0f;
        sprite.setPosition(position);
    }
```

- Konstruktor kelas SpecialPlatform yang menerima parameter window dan height. Konstruktor ini digunakan untuk menginisialisasi atribut dari kelas SpecialPlatform.

```
    void draw(sf::RenderWindow& window) {
        window.draw(sprite);
    }
```

```
    bool checkCollision(sf::FloatRect playerBounds) {
        return sprite.getGlobalBounds().intersects(playerBounds);
    }

    sf::Vector2f getPosition() const {
        return position;
    }
};
```

- Fungsi draw digunakan untuk menggambar sprite pada window.
- Fungsi checkCollision digunakan untuk memeriksa apakah pemain bersentuhan dengan sprite SpecialPlatform.
- Fungsi getPosition digunakan untuk mengembalikan posisi sprite SpecialPlatform.

- **Class BlockingObject**

```
class BlockingObject {
public:
    sf::Vector2f position;
    sf::Texture texture;
    sf::Sprite sprite;
    bool isActive; // Menandakan blocking object aktif atau tidak aktif
```

Mendefinisikan atribut dari kelas BlockingObject.

```
    BlockingObject(sf::RenderWindow& window)
        : texture(), sprite(), isActive(false) {
        texture.loadFromFile("images/chibi_kaido.png");
        sprite.setTexture(texture);

        // Set the position above the special platform
        position.x = window.getSize().x / 2.0f - texture.getSize().x / 2.0f;
        position.y = 550.0f;
        sprite.setPosition(position);
    }
```

Konstruktor kelas BlockingObject yang menerima parameter window. Konstruktor ini digunakan untuk menginisialisasi atribut dari kelas BlockingObject.

```
    void draw(sf::RenderWindow& window) {
        window.draw(sprite);
    }

    bool checkCollision(sf::FloatRect playerBounds) {
        return sprite.getGlobalBounds().intersects(playerBounds);
    }
```

- Fungsi draw digunakan untuk menggambar sprite pada window.
- Fungsi checkCollision digunakan untuk memeriksa apakah pemain bersentuhan dengan sprite BlockingObject.

```
    void setPosition(float x, float y) {
        position.x = x;
        position.y = y;
        sprite.setPosition(position);
    }

    void Active() {
        isActive = true;
    }

    void deactivate() {
        isActive = false;
    }

    bool getIsActive() const {
        return isActive;
    }
};
```

- Fungsi setPosition digunakan untuk mengatur posisi sprite BlockingObject.
- Fungsi Active digunakan untuk mengaktifkan sprite BlockingObject.
- Fungsi deactivate digunakan untuk menonaktifkan sprite BlockingObject.
- Fungsi getIsActive digunakan untuk mengembalikan status aktivasi sprite BlockingObject.

• Class FallingObject

```
class FallingObject {
public:
    sf::Vector2f position;
    sf::Texture texture;
    sf::Sprite sprite;
    float speed;
```

Kelas FallingObject terdiri dari atribut :

- position: Menyimpan posisi sprite FallingObject.
- texture: Menyimpan tekstur sprite FallingObject.
- sprite: Menyimpan sprite FallingObject.
- speed: Menyimpan kecepatan jatuhnya objek.

```
    FallingObject(float fallSpeed)
        : speed(fallSpeed) {
        texture.loadFromFile("images/fireballs.png");
        sprite.setTexture(texture);
        sprite.setPosition(getRandomX(), 0);
    }
```

Konstruktor FallingObject :

- Menerima parameter fallSpeed yang digunakan untuk mengatur kecepatan jatuhnya objek.
- Memuat tekstur sprite FallingObject dari file gambar fireballs.png.
- Mengatur posisi awal sprite FallingObject pada sumbu X dan Y dengan memanggil fungsi getRandomX() dan mengatur posisi Y ke 0.

```
    void update(float dt) {
        position.y += speed * dt;

        if (position.y > 700) {
            position.y = 0;
            position.x = getRandomX();
        }

        sprite.setPosition(position);
    }
```

Fungsi update :

- Menerima parameter dt yang merupakan waktu delta sejak pembaruan terakhir.
- Mengubah posisi Y sprite FallingObject dengan menambahkan kecepatan jatuhnya dikalikan dengan dt.
- Jika posisi Y sprite FallingObject melebihi 700, maka posisi Y diatur kembali ke 0 dan posisi X diatur ulang dengan memanggil fungsi getRandomX().
- Mengatur posisi sprite FallingObject sesuai dengan posisi yang telah diubah.

```
    float getRandomX() {
        std::uniform_int_distribution<unsigned> x(0, 500 - texture.getSize().x);
        std::default_random_engine e(time(0));
        return static_cast<float>(x(e));
    }
};
```

Fungsi getRandomX yaitu Mengembalikan nilai acak antara 0 dan 500 dikurangi lebar tekstur sprite FallingObject.

- # Fungsi main

```
int main()
{
    sf::RenderWindow window(sf::VideoMode(500, 700), "Luffy Jump!", sf::Style::Close);
    window.setFramerateLimit(60);

mainmenu :

    showMainMenu(window);

    sf::Texture backgroundTexture;
    sf::Texture playerTexture;
    sf::Texture platformTexture;
    backgroundTexture.loadFromFile("images/FloatingIsland.jpeg");
    playerTexture.loadFromFile("images/Luffy_Gear5th.png");
    platformTexture.loadFromFile("images/Awan.png");
    sf::Sprite background(backgroundTexture);
    sf::Sprite player(playerTexture);
    sf::Sprite platform(platformTexture);
    std::vector<SpecialPlatform> specialPlatforms;
    bool displaySpecialPlatform = false;

    // Jumlah loncatan khusus yang diperoleh setelah menggunakan special platform
    int specialPlatformJumps = 2;

    // Counter untuk menghitung berapa kali player telah melakukan loncatan khusus
    int jumpsRemaining = 0;
```

- Membuat objek window dengan ukuran 500 x 700 dan judul “Luffy Jump!”.
- Mengatur window agar tidak dapat diubah ukurannya.
- Mengatur batas frame rate menjadi 60 fps.
- Fungsi showMainMenu digunakan untuk menampilkan menu utama game.
- Memuat gambar latar belakang, karakter pemain, dan platform dari file gambar.
- Membuat objek sprite dari gambar latar belakang.
- Membuat objek sprite dari gambar karakter pemain.
- Membuat objek sprite dari gambar platform.
- Membuat vektor untuk menyimpan objek SpecialPlatform.
- Mengatur variabel displaySpecialPlatform menjadi false.
- Mendefinisikan variabel specialPlatformJumps dan jumpsRemaining.

```
BlockingObject blockingObject(window);
bool playerOnSpecialPlatform = false;

// Inisialisasi objek jatuh
FallingObject fallingObject(300.0f);
sf::Clock clock;
bool gameOver = false;
```

- Membuat objek BlockingObject.
- Mengatur variabel playerOnSpecialPlatform menjadi false.
- Mendefinisikan ukuran gambar yang digunakan dalam game.
- Membuat objek FallingObject dengan kecepatan jatuh sebesar 300 piksel per detik.
- Membuat objek Clock untuk menghitung waktu.
- Mengatur variabel gameOver menjadi false.

```
//Mengganti Jenis Font dalam Game
sf::Font super_peach,watermark_font,Restart_Font;
super_peach.loadFromFile("font/Super Peach.ttf");
watermark_font.loadFromFile("font/Darwin Smith.otf");
Restart_Font.loadFromFile("font/RESEARCH AND DEVELOPMENT.otf");

sf::Text scoreText;
scoreText.setFont(super_peach);
scoreText.setCharacterSize(40);
scoreText.setFillColor(sf::Color::Red);

sf::RectangleShape gameoverBackground(sf::Vector2f(500, 700));
gameoverBackground.setFillColor(sf::Color::White);
sf::Text gameoverText;
gameoverText.setFont(super_peach);
gameoverText.setString("Game Over!");
gameoverText.setCharacterSize(75);
gameoverText.setFillColor(sf::Color::Red);
```

- Memuat font yang digunakan dalam game.
- Membuat objek Text untuk menampilkan skor.
- Membuat objek RectangleShape untuk menampilkan latar belakang game over.
- Membuat objek Text untuk menampilkan teks “Game Over!”.

```
  // initialize platforms
  sf::Vector2u platformPosition[10];
  std::uniform_int_distribution<unsigned> x(0, 500 - platformTexture.getSize().x);
  std::uniform_int_distribution<unsigned> y(100, 700);
  std::default_random_engine e(time(0));
  for (size_t i = 0; i < 10; ++i)
  {
      platformPosition[i].x = x(e);
      platformPosition[i].y = y(e);
  }

  // player's positon and down velocity
  int playerX = 250;
  int playerY = 151;
  float dy = 0;
  int height = 150;
  int score = 0;

  // player's bounding box. It should modify according to the image size
  const int PLAYER_LEFT_BOUNDING_BOX = 20;
  const int PLAYER_RIGHT_BOUNDING_BOX = 60;
  const int PLAYER_BOTTOM_BOUNDING_BOX = 70;
```

- Membuat array platformPosition yang berisi posisi platform.
- Mendefinisikan distribusi acak untuk posisi X dan Y platform.
- Membuat objek default_random_engine dengan seed waktu saat ini.
- Mengisi array platformPosition dengan posisi acak.
- Mendefinisikan posisi awal pemain dan kecepatan jatuhnya.
- Mendefinisikan tinggi pemain, skor awal, bounding box pemain.

```
//restart
sf::Text restartText;
restartText.setFont(Restart_Font);
restartText.setString("Press 'R' to Restart");
restartText.setCharacterSize(10);
restartText.setFillColor(sf::Color::Black);
restartText.setPosition(150, 500);
```

Mendefinisikan teks “Press ‘R’ to Restart”

```
while (window.isOpen() && !gameOver)
{

    sf::Event event;
    while (window.pollEvent(event))
    {
        if (event.type == sf::Event::Closed)
            window.close();
    }

    if (sf::Keyboard::isKeyPressed(sf::Keyboard::A) || sf::Keyboard::isKeyPressed(sf::Keyboard::Left))
        playerX -= 4;
    if (sf::Keyboard::isKeyPressed(sf::Keyboard::D) || sf::Keyboard::isKeyPressed(sf::Keyboard::Right))
        playerX += 4;

    //Mengatur agar player keluar kiri layar maka akan masuk ke kanan layar begitupun sebalinya
    if (playerX > 500)
        playerX = 0;
    if (playerX < -40)
        playerX = window.getSize().x - playerTexture.getSize().x;

    if (playerY == height && dy < (-1.62))
    {
        score += 1;
        scoreText.setString("Score: " + std::to_string(score));
    }
```

- Memulai loop utama game.
- Memeriksa event yang terjadi pada window.
- Menggerakkan pemain ke kiri atau kanan jika tombol A atau D atau kiri atau kanan ditekan.
- Memeriksa apakah pemain berada di luar batas window dan mengubah posisi pemain jika iya.
- Memeriksa apakah pemain berada pada platform dan mengubah kecepatan jatuhnya jika iya.
- Mengubah posisi pemain berdasarkan kecepatan jatuhnya.
- Memeriksa apakah pemain berada di atas platform dan mengubah skor jika iya.

```
// Mengatur Lompatan Karakter player
dy += 0.2;
playerY += dy;

if (playerY < height)
    for (size_t i = 0; i < 10; ++i)
    {
        playerY = height;
        platformPosition[i].y -= dy;  // vertical translation
        if (platformPosition[i].y > 700) // set new platform on the top
        {
            platformPosition[i].y = 0;
            platformPosition[i].x = x(e);
        }
    }

for (size_t i = 0; i < 10; ++i)
{
    if ((playerX + PLAYER_RIGHT_BOUNDING_BOX > platformPosition[i].x) && (playerX + PLAY
        && (playerY + PLAYER_BOTTOM_BOUNDING_BOX > platformPosition[i].y) && (playerY +
        && (dy > 0)) // player is falling
    {
        //sound.play();
        dy = -10;
    }
}
player.setPosition(playerX, playerY);

window.draw(background);
window.draw(player);
```

- **Mengatur Lompatan Karakter player**: Pada bagian ini, posisi player akan diatur agar player dapat melompat ke atas dan ke bawah.
- **Platform movement**: Pada bagian ini, posisi platform akan diatur agar platform dapat bergerak ke atas dan ke bawah.
- **Collision detection**: Pada bagian ini, akan dilakukan deteksi apakah player bersentuhan dengan platform. Jika player bersentuhan dengan platform, maka player akan melompat ke atas.
- **Rendering**: Pada bagian ini, game akan dirender pada window

```
    // set and draw platforms
    for (size_t i = 0; i < 10; ++i)
    {
        platform.setPosition(platformPosition[i].x, platformPosition[i].y);
        window.draw(platform);
    }

    //Special Platform

    if (!displaySpecialPlatform) {
        displaySpecialPlatform = true;
        specialPlatforms.emplace_back(window, 100.0f);
    }

    // Update and draw special platforms
    for (auto& specialPlatform : specialPlatforms) {
        specialPlatform.draw(window);
        if (specialPlatform.checkCollision(player.getGlobalBounds()) && dy > 0 && !specialPlatform.isU
            // Efek loncatan khusus
            dy = -100.0;
            jumpsRemaining = specialPlatformJumps;  // Setel jumlah loncatan khusus yang tersisa

            // Set special platform sudah digunakan
            specialPlatform.isUsed = true;

            blockingObject.Active();
        }
    }
```

- **Mengatur posisi platform pada window**: Pada bagian ini, posisi platform akan diatur agar platform dapat digambar pada window.

- **Membuat platform khusus**: Pada bagian ini, akan dibuat platform khusus yang dapat memberikan efek loncatan khusus pada player jika player menyentuh platform tersebut. Platform khusus ini akan muncul secara acak pada game.
- **Update dan menggambar platform khusus**: Pada bagian ini, platform khusus akan diupdate dan digambar pada window. Jika player menyentuh platform khusus tersebut dan player sedang jatuh, maka player akan melompat ke atas dengan efek loncatan khusus.

```
// Kurangi jumlah loncatan khusus yang tersisa jika player sedang dalam efek loncatan khusus
if (jumpsRemaining > 0) {
    jumpsRemaining--;
}

// Draw blocking object only when it is active
if (blockingObject.getIsActive()) {
    blockingObject.draw(window);

    // Check for collision with blocking object
    if (blockingObject.checkCollision(player.getGlobalBounds())) {
        // Game over if player touches the blocking object
        gameOver = true;
    }
}

// update objek jatuh
sf::Time elapsed = clock.restart();
float dt = elapsed.asSeconds();
fallingObject.update(dt);
window.draw(fallingObject.sprite);

// Jika pemain bertabrakan dengan objek jatuh
if ((playerX + PLAYER_RIGHT_BOUNDING_BOX > fallingObject.position.x) &&
    (playerX + PLAYER_LEFT_BOUNDING_BOX < fallingObject.position.x + fallingObject.texture.getSize(
    (playerY + PLAYER_BOTTOM_BOUNDING_BOX > fallingObject.position.y) &&
    (playerY + PLAYER_BOTTOM_BOUNDING_BOX < fallingObject.position.y + fallingObject.texture.getSiz
    // Game over
    gameOver = true;
}
```

- Mengurangi jumlah loncatan khusus yang tersisa jika pemain sedang dalam efek loncatan khusus.
- Menggambar objek BlockingObject jika aktif dan memeriksa apakah pemain bersentuhan dengan objek tersebut. Jika iya, maka mengatur status game over.
- Memperbarui posisi objek jatuh dan menggambar objek tersebut.
- Memeriksa apakah pemain bersentuhan dengan objek jatuh dan mengatur status game over jika iya.

```
    // menentukan kapan objek jatuh muncul
    if (fallingObject.position.y > 700) {
        fallingObject.position.y = 0;
        fallingObject.position.x = fallingObject.getRandomX();
    }

    // game over
    if (playerY > 700)
    {
        scoreText.setPosition(150, 400);
        goto gameover;
    }
    window.draw(scoreText);
    window.display();
}
```

- Menentukan kapan objek jatuh muncul.
- Memeriksa apakah pemain berada di luar batas window dan mengatur status game over jika iya.
- Menggambar skor.dalam berlangsungnya permainan.

```
    // Game Over
gameover:
    while (window.isOpen())
    {
        sf::Event event;
        while (window.pollEvent(event))
        {
            if (event.type == sf::Event::Closed)
                window.close();
        }

        gameoverText.setPosition(50, 200);
        scoreText.setPosition(150, 300);

        window.draw(gameoverBackground);
        window.draw(gameoverText);
        window.draw(scoreText);

        if (sf::Keyboard::isKeyPressed(sf::Keyboard::R))
        {
            // Reset game variables
            playerX = 250;
            playerY = 151;
            dy = 0;
            height = 150;
            score = 0;
            jumpsRemaining = 0;
```

```
            // Reset special platforms
            for (auto& specialPlatform : specialPlatforms) {
                specialPlatform.isUsed = false;
            }

            // Reset blocking object
            blockingObject.deactivate();

            // Generate new special platforms
            displaySpecialPlatform = false;

            //generate new platform positions
            for (size_t i = 0; i < 10; ++i)
            {
                platformPosition[i].x = x(e);
                platformPosition[i].y = y(e);
            }

            gameOver = false;
            goto mainmenu;
        }

        window.draw(restartText);
        window.draw(watermark);

        window.display();
    }
    return 0;
}
```

- Jika ingin memulai kembali permainan bisa menekan tombol ‘R’. Tombol ini akan mengatur ulang variabel permainan dan mengembalikanmu ke menu utama. Variabel yang diatur ulang meliputi posisi pemain, kecepatan jatuhnya, tinggi pemain, skor, dan jumlah loncatan khusus yang tersisa. Objek SpecialPlatform akan diatur ulang dan objek BlockingObject akan dinonaktifkan. Posisi platform juga akan diatur ulang

E. Cara Menjalankan Games

1) Pastikan SFML telah diinstal dan dapat diakses oleh compiler. Disini saya menggunakan Visual Studio 2022 Community. Untuk setup SFML dan Visual Studio dapat dilihat disini https://youtu.be/lFzpkvrscs4?si=awPY-3g58HmrVJ5k
2) Sertakan semua file gambar, font, dan file sumber ke dalam proyek. Disini saya menyediakan semua assets yang dapat didownload melalui link berikut : https://drive.google.com/drive/folders/1hreIV1pRLUnMGcnN1EMNpTpcQOgCeapo?usp=sharing
3) Pastikan semua file gambar, font, dan file sumber tersedia sesuai dengan direktori yang ditentukan dalam kode.
4) Kompilasi dan jalankan kode sumber menggunakan compiler C++ dengan dukungan SFML. Jika menggunakan Visual Studio jalankan dengan menekan *Local Windows Debugger* dan jangan lupa untuk mengubah ke mode *Debug*.
5) Pada layar pertama, tekan 'S' untuk memulai permainan.
6) Selama permainan, kendalikan pemain dengan tombol panah kiri dan kanan.atau bisa menggunakan keyboard A (untuk kontrol ke kiri) dan D (untuk kontrol ke kanan).
7) Hindari Object yang jatuh dan teruslah melompat untuk mencapai titik tertinggi.
8) Jika bertabrakan dengan object yang jatuh atau bertabrakan dengan object musuh atau jatuh kebawah layar maka akan game over.
9) Tekan tombol 'R' untuk merestart permainan setelah game over.

F. Sumber Sourcecode dan Kontribusi Modifikasi

Game “Luffy Jump!” memiliki konsep yang sama dengan game sederhana dan popular yakni “Doodle Jump!”. Disini saya mengambil sourcecode milik Asbolus (https://github.com/Asbolus/Doodle-Jump-cpp-SFML/blob/master/main.cpp) kemudian memodifikasinya menjadi game baru yang diberi nama yakni “Luffy Jump!”.

Sourcecode milik Asbolus tersebut menampilkan game “Doodle Jump”, dimana pemain dapat mengontrol karakter untuk melompat dari platform ke platform lain dan skor akan meningkat seiring dengan lompatan – lompatan tersebut. Ketika karakter jatuh dibawah layar, maka akan menampilkan layar gameover dan skor yang telah diperoleh.

Modifikasi yang saya lakukan pada sourcecode tersebut,antara lain :

- Menambahkan tampilan awal sebelum memulai game yang menampilkan title dari game dan perintah menekan tombol S untuk memulai game.
- Menambahkan object yang jatuh kebawah dan ketika player menyentuh object tersebut maka akan menimbulkan gameover.
- Menambahkan object special berupa awan yang berwarna pink yang berguna agar karakter dapat lompat lebih tinggi dari sebelumnya
- Menambahkan object karakter villain sebagai musuh dari karakter utama dalam game yang akan muncul ketika player menggunakan object special.
- Menambahkan fitur restart pada tampilan gameover agar dapat memulai kembali game.
- Mengganti font yang berbeda setiap segmen dalam game
- Mengganti images player, platform, dan background dalam game.