JUMPY KIRBY: GAME PLATFORMER BERBASIS PYTHON PYGAME

Mata Kuliah

Pemrograman Berorientasi Objek

Oleh:

Alifah Nurul Fitriyani

24091397136

2024D

PROGRAM STUDI MANAJEMEN INFORMATIKA

FAKULTAS VOKASI

UNIVERSITAS NEGERI SURABAYA

2025

DAFTAR ISI

BAB I

PENDAHULUAN

1.1 Latar Belakang

Pemrograman Berorientasi Objek (Object-Oriented Programming/OOP) merupakan salah satu paradigma pengembangan perangkat lunak yang paling banyak digunakan dalam industri teknologi informasi. Paradigma ini menawarkan pendekatan yang terstruktur dan terorganisir dengan menjadikan entitas di dalam sistem sebagai objek yang memiliki atribut, perilaku, serta hubungan satu sama lain. Melalui konsep-konsep seperti enkapsulasi, pewarisan, dan polimorfisme, OOP memfasilitasi pengembangan perangkat lunak yang modular, mudah dikelola, serta mampu beradaptasi terhadap perubahan kebutuhan.

Dalam dunia pendidikan tinggi, pemahaman mengenai OOP menjadi kompetensi fundamental, khususnya bagi mahasiswa program studi yang berfokus pada pengembangan perangkat lunak. Penerapan OOP tidak hanya penting secara teoritis, tetapi juga secara praktis karena memberikan dasar untuk membangun aplikasi yang kompleks, termasuk aplikasi berbasis antarmuka grafis dan game interaktif. Keterampilan tersebut selaras dengan tuntutan teknologi masa kini, di mana kemampuan untuk membangun aplikasi terstruktur menjadi kebutuhan utama dalam dunia kerja.

Sebagai bagian dari evaluasi pembelajaran pada mata kuliah Pemrograman Berorientasi Objek, mahasiswa ditugaskan untuk mengembangkan sebuah aplikasi yang menerapkan konsep OOP secara nyata. Kebebasan diberikan untuk menentukan jenis aplikasi yang dikembangkan, termasuk game 2D, aplikasi pengelolaan data, maupun aplikasi berbasis antarmuka grafis. Game 2D dipandang sangat relevan sebagai media pembelajaran karena terdiri dari berbagai objek yang berinteraksi, seperti karakter pemain, musuh, platform, sistem skor, latar belakang, dan elemen interaktif lainnya. Setiap objek tersebut

dapat direpresentasikan sebagai kelas dalam OOP, sehingga sangat mendukung penerapan prinsip kelas, objek, pewarisan, dan polimorfisme secara langsung.

Berdasarkan pertimbangan tersebut, dikembangkan sebuah game berjudul "Jumpy Kirby" menggunakan bahasa Python dan library Pygame. Game ini termasuk jenis vertical platformer, di mana pemain mengendalikan karakter Kirby untuk melompat ke platform-platform yang tersedia sambil menghindari musuh yang bergerak horizontal. Selama permainan berlangsung, skor akan meningkat seiring dengan ketinggian yang berhasil dicapai. Selain itu, game dilengkapi dengan fitur-fitur tambahan seperti animasi berbasis spritesheet, pergantian latar belakang menurut skor yang dicapai, efek suara, tampilan menu utama, tombol interaktif, serta sistem penyimpanan skor tertinggi (highscore).

Jumpy Kirby tidak hanya bertujuan sebagai aplikasi hiburan, namun juga sebagai demonstrasi implementasi OOP yang terstruktur. Setiap komponen game diorganisasikan ke dalam kelas-kelas mandiri yang saling berinteraksi melalui mekanisme game loop. Dengan merancang game ini menggunakan pendekatan OOP, aplikasi menjadi lebih mudah dikembangkan, diuji, dan dibenahi. Struktur kelas yang jelas juga memungkinkan pengembangan fitur tambahan di masa depan, seperti penambahan level, variasi musuh, atau sistem poin yang lebih kompleks.

Dengan demikian, pengembangan game "Jumpy Kirby" menjadi sarana pembelajaran yang relevan untuk memahami dan mempraktikkan prinsip-prinsip OOP dalam konteks nyata. Melalui integrasi logika permainan, manajemen objek, animasi, serta desain antarmuka, aplikasi ini merepresentasikan implementasi langsung dari paradigma OOP yang dapat digunakan sebagai dasar pembelajaran dalam pengembangan sistem interaktif lainnya.

1.2 Rumusan Masalah

Berdasarkan latar belakang tersebut, beberapa rumusan masalah yang diidentifikasi adalah sebagai berikut:

1. Bagaimana menerapkan konsep OOP seperti enkapsulasi, pewarisan, dan polimorfisme pada pengembangan game 2D menggunakan Python dan Pygame?
2. Bagaimana merancang struktur kelas dan hubungan antarobjek agar game memiliki arsitektur yang modular dan mudah dikembangkan?
3. Bagaimana mengimplementasikan mekanisme gameplay, termasuk pergerakan karakter, platform dinamis, musuh animatif, scrolling, efek suara, dan sistem skor?
4. Bagaimana memastikan seluruh fitur game berjalan dengan stabil melalui proses pengujian fungsional?

1.3 Tujuan

Pengembangan aplikasi "Jumpy Kirby" memiliki tujuan sebagai berikut:

1. Menerapkan prinsip utama OOP secara menyeluruh dalam pembangunan aplikasi game 2D.
2. Menghasilkan game yang berfungsi stabil, memiliki antarmuka yang interaktif, dan mencerminkan konsep pemrograman terstruktur.
3. Menyusun dokumentasi perancangan yang mencakup class diagram, alur permainan, struktur program, serta deskripsi fitur-fitur yang diimplementasikan.
4. Memperkuat kemampuan dalam mengelola objek game, logika permainan, sistem animasi, dan manajemen event menggunakan Pygame.
5. Memvalidasi penerapan konsep OOP melalui implementasi nyata dalam bentuk game.

1.4 Batasan Masalah

Batasan masalah ditetapkan agar proses pengembangan tetap fokus dan sesuai ruang lingkup akademik, yaitu:

1. Pengembangan hanya menggunakan bahasa Python dan library Pygame tanpa integrasi framework lain.

2. Game hanya mencakup satu karakter utama, satu jenis musuh, dan platform sebagai objek lingkungan.
3. Mekanisme permainan terbatas pada mode tunggal (single endless mode) tanpa sistem level.
4. Penyimpanan data hanya mencakup skor tertinggi yang disimpan menggunakan file teks.
5. Antarmuka pengguna hanya mencakup menu utama, gameplay, dan tampilan game over.
6. Efek suara dan animasi disesuaikan dengan kebutuhan gameplay tanpa penggunaan aset berlisensi tinggi.

1.5 Metodologi

Pengembangan aplikasi dilakukan melalui beberapa tahapan metodologis sebagai berikut:

1. Analisis Kebutuhan

   Tahap ini mengidentifikasi kebutuhan fungsional dan non-fungsional yang diperlukan dalam pengembangan game, meliputi pergerakan karakter, animasi, sistem skor, dan interaksi objek.
2. Perancangan sistem

   Pada tahap ini dilakukan perancangan struktur kelas, hubungan antarobjek, alur permainan, dan desain antarmuka dasar. Perancangan juga mencakup pembuatan class diagram dan struktur folder aplikasi.
3. Implementasi Program

   Tahap ini merupakan proses penerjemahan desain ke dalam kode Python menggunakan pendekatan OOP. Implementasi mencakup pembuatan kelas Player, Platform, Enemy, tombol interaktif, background dinamis, efek suara, serta logika game loop.
4. Pengujian Fungsional

   Pengujian dilakukan untuk memastikan setiap fitur berjalan sesuai kebutuhan. Pengujian mencakup mekanisme pergerakan, collision

detection, animasi musuh, scrolling background, sistem skor, dan input pengguna.

5. Evaluasi dan Dokumentasi

   Tahap akhir mencakup evaluasi hasil implementasi, pengecekan stabilitas sistem, dan penyusunan laporan akademis yang memuat penjelasan lengkap mengenai perancangan dan implementasi aplikasi.

BAB II
LANDASAN TEORI

2.1 Pemrograman Berorientasi Objek

Pemrograman Berorientasi Objek atau Object-Oriented Programming (OOP) merupakan paradigma pemrograman yang memodelkan perangkat lunak melalui representasi objek di dunia nyata. Objek-objek tersebut berinteraksi melalui metode yang saling memanggil sehingga membentuk alur kontrol program yang terstruktur, modular, dan mudah dikembangkan. OOP didesain untuk mengatasi lemahnya metode pemrograman prosedural dalam menangani sistem yang kompleks, terutama sistem yang mengandung banyak entitas dengan perilaku berbeda.

Secara filosofis, OOP berangkat dari konsep antropomorfisme, yakni upaya memandang komponen perangkat lunak sebagai "entitas hidup" yang memiliki pengetahuan, perilaku, serta batasan interaksi. Dengan mekanisme ini, setiap objek bertanggung jawab untuk menjaga konsistensi data dan menjalankan fungsinya tanpa perlu mengetahui detail seluruh sistem. Paradigma ini sangat sesuai digunakan dalam bidang pengembangan game, karena game terdiri dari banyak entitas yang bergerak, berinteraksi, memiliki perilaku, status, dan siklus hidup masing-masing.

Dalam game "Jumpy Kirby", OOP memungkinkan pemisahan logika karakter, platform, musuh, tombol, sistem animasi, serta manajemen background dalam bentuk kelas-kelas spesifik. Pemisahan ini meningkatkan skalabilitas, memungkinkan pengembangan fitur tambahan di masa depan seperti power-up, level baru, mekanik skor tambahan, atau variasi musuh tanpa harus menulis ulang struktur inti permainan.

2.1.1 Konsep Enkapsulasi

Enkapsulasi merujuk pada metode penyembunyian data milik suatu objek dari akses langsung oleh objek lain. Melalui konsep ini, sebuah objek hanya berinteraksi melalui antarmuka yang telah ditentukan, sehingga integritas data terjaga. Enkapsulasi menjaga agar suatu objek memiliki kontrol penuh terhadap bagaimana state internalnya dimodifikasi.

Dalam konteks game "Jumpy Kirby", atribut seperti:

- kecepatan vertikal pemain (vel_y),
- arah pergerakan (flip),
- posisi koordinat dalam bentuk rect (self.rect),

disembunyikan di dalam kelas Player dan hanya boleh diubah melalui metode seperti move() atau pembaruan internal lain. Dengan demikian, entitas lain seperti musuh atau platform tidak memiliki kewenangan untuk mengubah nilai-nilai tersebut secara langsung. Sistem ini menciptakan pemisahan tanggung jawab yang jelas, sehingga mempermudah proses debugging dan memperkecil risiko gangguan interaksi antarobjek.

2.1.2 Konsep Pewarisan (Inheritance)

Pewarisan adalah kemampuan sebuah kelas untuk mewarisi atribut dan metode dari kelas lain. Pewarisan memungkinkan pembentukan hierarki kelas yang lebih terstruktur, meningkatkan reuse kode, dan mengurangi keberulangan implementasi.

Dalam game ini, kelas Platform dan Enemy mewarisi kelas dasar pygame.sprite.Sprite. Dengan melakukan pewarisan tersebut, objek:

- otomatis terintegrasi ke dalam sistem sprite group,
- mendapatkan kemampuan rendering standar,
- serta mengikuti mekanisme update sprite yang telah ditentukan Pygame.

Penggunaan pewarisan ini memungkinkan penulis game untuk memfokuskan energi pada pengembangan logika spesifik seperti gerakan musuh, prosedur spawn platform, dan animasi, tanpa perlu menulis ulang mekanisme dasar seperti struktur gambar, bounding box, dan integrasi grup.

2.1.3 Konsep Polimorfisme

Polimorfisme berarti metode yang sama dapat menghasilkan output atau perilaku berbeda tergantung objek yang memanggilnya. Konsep ini memperkuat modularisasi, karena memungkinkan sistem untuk memproses berbagai objek menggunakan metode tunggal tanpa perlu memeriksa jenis objek terlebih dahulu.

Pada game ini, polimorfisme tampak pada metode:

- update() milik Platform
- update() milik Enemy

Keduanya memiliki nama yang sama, dipanggil dengan cara yang sama, namun berperilaku berbeda. Platform hanya memperbarui posisi vertikalnya sesuai pergerakan kamera (scroll), sedangkan Enemy melakukan pembaruan frame animasi, pergerakan horizontal, dan regenerasi mask pixel untuk collision detection.

Dengan demikian, sistem game dapat menjalankan seluruh elemen game dalam satu pengulangan update tanpa membedakan sumber objek, yang menghasilkan arsitektur lebih bersih dan ringkas.

2.1.4 Konsep Abstraksi

Abstraksi adalah penyederhanaan objek kompleks dengan cara hanya menampilkan fitur penting kepada pemrogram, sementara implementasi detail disembunyikan. Abstraksi meningkatkan pemahaman sistem dan mengurangi beban kognitif pengembang.

Pada game ini, abstraksi dapat dilihat pada:

- penggunaan kelas SpriteSheet untuk mengambil frame animasi tanpa harus menangani koordinat pixel,
- mekanisme pygame.draw dan blit yang menyembunyikan detail proses rendering GPU,
- sprite group pada Pygame yang menyederhanakan manajemen ratusan objek secara otomatis.

Melalui abstraksi, sistem menjadi lebih mudah dikembangkan, lebih bersih, dan lebih mudah dipelihara.

2.2 Bahasa Pemrograman Python

Python merupakan bahasa pemrograman interpretatif yang dirancang dengan fokus pada keterbacaan kode, kesederhanaan sintaks, serta produktivitas pengembang. Python mendukung berbagai paradigma pemrograman, termasuk prosedural, fungsional, dan berorientasi objek, sehingga sangat fleksibel digunakan dalam berbagai konteks pengembangan.

Python memiliki beberapa karakteristik penting, yaitu:

1. Sintaks yang mudah dipahami

   Python memiliki struktur kode yang sederhana, sehingga sangat ideal untuk pengembangan prototipe cepat maupun aplikasi akademis.
2. Library yang sangat luas

   Python memiliki ekosistem library yang sangat kaya, mendukung berbagai kebutuhan mulai dari data science hingga pembuatan game.
3. Komunitas yang besar

   Python memiliki documentasi lengkap dan komunitas global yang berkontribusi aktif dalam pengembangan library dan tools.
4. Mendukung OOP secara penuh

   Python menyediakan dukungan kelas, objek, pewarisan, dan mekanisme OOP lainnya, sehingga digunakan secara luas dalam pendidikan dan industri.

Python sangat cocok untuk pengembangan game 2D berskala kecil hingga menengah, karena sifatnya yang fleksibel serta dukungan library grafis seperti Pygame.

2.3 Pygame

Pygame adalah library Python yang digunakan untuk membuat game dan aplikasi multimedia berbasis 2D. Library ini dibangun di atas SDL (Simple DirectMedia Layer), yang merupakan framework lintas platform untuk pengelolaan grafik, suara, input, dan event.

Beberapa fitur utama yang membuat Pygame relevan dalam pengembangan game “Jumpy Kirby” yaitu:

1. Sistem grafis

   Pygame menyediakan fungsi untuk memuat, menampilkan, dan memanipulasi gambar dalam berbagai format. Fitur ini memungkinkan pembuatan karakter, platform, musuh, dan background melalui fungsi pygame.image.load() serta transformasi dengan pygame.transform.scale() dan flip().

2. Manajemen event

   Pygame menangani input pengguna seperti keyboard dan mouse melalui pygame.event.get(). Sistem ini memungkinkan karakter merespons gerakan pemain secara real-time.

3. Sprite dan group

   Pygame menyediakan modul sprite untuk mempermudah manajemen objek game. Dengan sprite group, objek dapat dirender dan diperbarui secara terorganisir.

4. Audio dan efek suara

   Library ini menyediakan modul pygame.mixer untuk memutar musik latar (BGM) dan efek suara seperti lompatan dan kematian.

5. Pengaturan kecepatan frame

   Pygame menyediakan objek Clock() untuk mengatur FPS sehingga animasi dan pergerakan objek tetap stabil.

2.4 Game Development Dasar

Pengembangan game 2D melibatkan berbagai konsep fundamental yang menjadi fondasi logika permainan. Konsep-konsep ini diterapkan secara langsung dalam implementasi game “Jumpy Kirby” guna memastikan

permainan dapat berjalan stabil, responsif, dan konsisten. Subbab ini menguraikan empat elemen utama dalam pengembangan game, yaitu game loop, event handling, collision detection, dan sprite system. Keempat elemen ini merupakan komponen inti yang membedakan game dari aplikasi konvensional.

1. Game Loop

   Game loop merupakan *inti* dari permainan. Tidak seperti aplikasi biasa yang berjalan berdasarkan event tertentu, game memerlukan pemutakhiran (update) yang berlangsung terus-menerus pada setiap frame. Game loop bertanggung jawab mengatur siklus permainan sehingga setiap elemen—seperti animasi, pergerakan karakter, logika fisika, collision, dan rendering—diperbarui secara berkala dalam interval waktu yang sangat kecil.

   Secara umum, sebuah game loop terdiri dari tiga proses utama:

   a. Processing Input

      Permainan membaca seluruh input dari pengguna, seperti tekan tombol, gerakan mouse, dan klik.

   b. Updating Game State

      Proses ini meliputi pembaruan posisi objek, penerapan gravitasi, scroll background, pembangkitan platform, perhitungan skor, serta deteksi tabrakan.

   c. Rendering

      Setelah seluruh logika diperbarui, sistem menggambar seluruh elemen permainan ke layar secara berurutan. Frame inilah yang terlihat oleh pemain.

   Dalam game “Jumpy Kirby”, game loop bekerja pada 60 FPS stabil berkat penggunaan clock.tick(60). Stabilitas frame rate ini penting untuk menjaga responsivitas permainan—misalnya, jika FPS turun, gerakan karakter menjadi patah-patah dan pengalaman bermain terganggu.

2. Event Haldling

   Event handling merupakan sistem yang bertugas menangani seluruh interaksi pengguna terhadap game. Pygame menyediakan mekanisme event yang harus diproses secara eksplisit oleh program agar permainan dapat merespons tindakan pemain dalam waktu nyata.
   Jenis-jenis event yang umum dalam pengembangan game antara lain:
   - Keyboard event: menekan tombol kiri atau kanan untuk mengerakkan karakter.
   - Mouse event: klik pada tombol start, exit, atau restart.
   - Quit event: menutup jendela game.

   Pada "Jumpy Kirby", event handling memainkan peran krusial karena:
   1. Input horizontal menentukan posisi pemain di layar.
   2. Klik mouse menentukan navigasi pada menu utama dan layar game over.
   3. Event QUIT digunakan untuk menghentikan game loop dan keluar dari permainan.

   Event handling berjalan bersamaan dengan game loop. Setiap frame, permainan membaca daftar event menggunakan pygame.event.get() kemudian memprosesnya sesuai dengan state permainan: apakah berada pada menu, gameplay, atau game over.
3. Collision Detection

   Collision detection adalah proses pendeteksian tabrakan antara dua objek. Dalam game platformer, collision menentukan apakah pemain mendarat di platform, jatuh ke bawah, atau terkena musuh. Deteksi tabrakan harus dilakukan secara konsisten untuk menjaga keadilan dan stabilitas permainan.
   Dalam game 2D, terdapat dua teknik collision umum:
   a. Rectangular Collision (Rect Collision)

      Teknik ini menggunakan kotak pembatas (bounding box) untuk menentukan apakah dua objek saling bertumpukan. Pygame menyediakan fungsi rect.colliderect() untuk melakukan pemeriksaan tersebut.

      Pada "Jumpy Kirby", rect collision digunakan untuk:

      - mendeteksi apakah pemain mendarat di platform,
      - mendeteksi apakah platform melewati batas bawah layar.

      Rect collision dipilih karena sederhana, cepat, dan efisien untuk objek berbentuk persegi panjang.

b. Mask-based Collision (Pixel-perfect Collision)

Teknik ini memeriksa tabrakan berdasarkan bentuk asli sprite pixel demi pixel. Mask collision digunakan ketika akurasi tinggi diperlukan, terutama untuk objek berbentuk tidak beraturan.

Dalam game ini, musuh menggunakan mask collision karena bentuknya lebih kompleks dan membutuhkan deteksi yang lebih presisi.

Mask collision diterapkan melalui:

- pembuatan mask dari surface sprite,
- penggunaan pygame.sprite.collide_mask() untuk deteksi tabrakan.

Dengan kombinasi kedua teknik, permainan tetap akurat namun tetap efisien dari sisi performa.

4. Sprite system

Sprite system merupakan salah satu fitur kunci dalam Pygame yang menyederhanakan manajemen objek dalam game. Sprite adalah objek grafis yang memiliki:

- gambar (image),
- posisi (rect),
- logika (update),
- perilaku tertentu.

Pygame menyediakan kelas dasar pygame.sprite.Sprite yang dapat diwarisi oleh objek seperti platform dan musuh. Sistem ini memungkinkan pengelolaan objek secara terstruktur melalui sprite group (pygame.sprite.Group()).

Manfaat sprite system dalam pengembangan game antara lain:

1. Manajemen objek yang terorganisir

   Seluruh entitas dapat dimasukkan ke dalam grup tertentu sehingga mudah untuk dirender, dihapus, atau diperbarui secara bersamaan.

2. Pemanggilan update secara otomatis

   Fungsi update() akan dijalankan untuk semua sprite dalam grup, sehingga logika game menjadi lebih sederhana.

3. Rendering batch

   Pygame memungkinkan group.draw(screen) untuk menggambar seluruh sprite secara efisien.

4. Pengelolaan daur hidup objek

   Objek yang keluar dari layar dapat dihapus (kill) sehingga tidak memakan memori.

Dalam “Jumpy Kirby”, kelas yang memanfaatkan sprite system adalah:

- Platform → pewarisan dari Sprite
- Enemy → pewarisan dari Sprite

Sementara Player tidak berada dalam sprite group, karena ia memiliki logika unik yang sangat spesifik dan tidak memerlukan manajemen yang sama seperti objek dinamis lainnya.

BAB III
PERANCANGAN APLIKASI

3.1 Deskripsi Umum Aplikasi

Aplikasi Jumpy Kirby merupakan sebuah permainan komputer berbasis *2D vertical platformer* yang dikembangkan menggunakan bahasa pemrograman Python dengan dukungan modul Pygame sebagai *game engine* utama. Aplikasi ini dirancang untuk menunjukkan integrasi konsep *Object-Oriented Programming (OOP)* ke dalam proses pengembangan permainan digital, serta memberikan pengalaman interaktif melalui mekanisme permainan yang sederhana namun menantang.

Jumpy Kirby mengadopsi konsep dasar permainan *endless jumping*, yaitu permainan yang berfokus pada kemampuan pemain untuk mempertahankan posisi karakter setinggi mungkin dalam dunia permainan yang bergulir secara vertikal. Mekanisme ini memberikan tantangan berkelanjutan, karena tingkat kesulitan meningkat seiring pemain melompat lebih tinggi dan menghadapi ancaman dalam bentuk musuh serta platform yang muncul secara prosedural.

Pengembangan aplikasi ini tidak hanya ditujukan untuk menghasilkan sebuah permainan yang dapat dimainkan, tetapi juga untuk mengaplikasikan konsep OOP seperti enkapsulasi, pewarisan, polimorfisme, dan modularisasi dalam konteks nyata. Setiap komponen permainan — mulai dari karakter pemain, platform, musuh, hingga manajemen antarmuka — dipisahkan menjadi kelas-kelas independen yang bekerja secara terkoordinasi melalui mekanisme *game loop*.

3.1.1 Genre Game

Jumpy Kirby dikategorikan sebagai permainan 2D Platformer dengan sub-genre Vertical Endless Platformer. Genre ini menekankan mekanisme permainan berupa pergerakan karakter secara horizontal dan vertikal dengan

fokus utama pada lompatan dari satu platform ke platform lain untuk mencapai ketinggian tertentu.

Beberapa karakteristik genre yang terimplementasi dalam permainan ini meliputi:

1. Lingkungan Dua Dimensi — seluruh objek ditampilkan dalam bidang 2D dengan posisi berbasis koordinat kartesian.
2. Platform sebagai Elemen Utama — pemain harus mendarat pada platform yang tersedia untuk mempertahankan posisi.
3. Progresi Vertikal Tanpa Batas — permainan tidak memiliki batas level; kesulitan meningkat bersamaan dengan ketinggian yang dicapai pemain.
4. Kemunculan Rintangan (Enemy) — muncul musuh secara acak yang berfungsi memperberat tantangan.

Dengan demikian, game ini menggabungkan unsur kelincahan, ketepatan waktu, dan refleks pemain untuk mempertahankan karakter dapat terus bergerak naik.

3.1.2 Tujuan Permainan

Tujuan utama permainan Jumpy Kirby adalah mempertahankan karakter agar dapat terus melompat ke platform di atasnya dan mencapai skor setinggi mungkin. Skor dihitung berdasarkan jarak vertikal yang berhasil ditempuh oleh pemain, yang direpresentasikan melalui sistem *scrolling* pada tampilan layar.

Tujuan-tujuan spesifik dari permainan ini meliputi:

1. Tujuan Gameplay
    - Pemain mempertahankan karakter agar tidak jatuh dari layar.
    - Pemain menghindari musuh yang muncul secara acak.
    - Pemain berupaya mencapai skor tertinggi selama sesi permainan.

2. Tujuan Teknis

    - Mengimplementasikan konsep OOP pada game 2D.
    - Menunjukkan penggunaan *event handling* dan *game loop* dalam Pygame.
    - Menerapkan deteksi tabrakan (collision detection) dengan sprite.

3. Tujuan Pembelajaran

    - Memberikan gambaran praktis mengenai pengelolaan objek, sprite, suara, dan animasi melalui Python dan Pygame.
    - Menunjukkan struktur aplikasi game dari sisi desain, logika, dan antarmuka.

3.1.3 Flow Permainan (Main Menu → Gameplay → Game Over)

Aplikasi Jumpy Kirby dirancang menggunakan pendekatan *finite state machine* dengan tiga state utama yang saling terhubung:

a. Main Menu

State pertama yang tampil ketika aplikasi dijalankan. Pada menu ini pemain disajikan:

- Tombol "Play" untuk memulai permainan.
- Tombol "Exit" untuk keluar dari aplikasi.
- Teks *Best Score* yang menunjukkan skor tertinggi yang tersimpan.

State ini berfungsi sebagai gerbang sebelum gameplay dimulai.

b. Gameplay

State inti dari aplikasi, di mana:

- Karakter pemain (Kirby) dapat bergerak ke kiri dan kanan.
- Sistem fisika berjalan, termasuk gravitasi, lompatan otomatis, dan scroll.

- Platform muncul dan bergerak sesuai scroll vertikal.
- Musuh muncul secara acak.
- Sistem skor terus bertambah seiring pergerakan vertikal karakter.

State ini berakhir apabila terjadi salah satu kondisi:

1. Pemain jatuh keluar dari layar.
2. Pemain menabrak musuh.

Kedua kondisi ini akan membawa permainan ke state "Game Over".

c. Game Over

State yang muncul setelah pemain kalah.

Pada tampilan ini disajikan:

- Skor akhir yang dicapai.
- Best Score yang diperbarui jika skor lebih tinggi dari sebelumnya.
- Tombol "Restart" untuk memulai ulang permainan.

Dari state ini pemain dapat memilih untuk mencoba kembali atau menutup aplikasi.

3.2 Fitur–Fitur Aplikasi

Pada tahap perancangan, setiap fitur permainan dirumuskan dengan mempertimbangkan aspek interaktivitas, fungsi, dan pengalaman pengguna. Adapun fitur-fitur utama dalam Jumpy Kirby yaitu:

1. Pergerakan Pemain

Pemain dapat menggerakkan karakter ke kiri dan kanan menggunakan tombol panah atau tombol A dan D pada keyboard. Lompatan dilakukan secara otomatis ketika karakter mendarat pada platform. Sistem ini menerapkan:

- Gravitasi
- Batas pergerakan horizontal
- Sistem mask untuk collision presisi

2. Kemunculan Musuh Secara Acak

Musuh akan muncul dengan probabilitas tertentu di atas platform baru. Musuh bergerak secara horizontal ke kiri dan kanan dengan kecepatan konstan. Sprite musuh menggunakan animasi berbasis sprite sheet.

3. Sistem Platform Acak

Platform baru muncul secara prosedural untuk menjaga kelanjutan permainan. Lebar, posisi horizontal, dan jarak antar platform diatur dengan random generator.

4. Sistem Skor dan Highscore

Skor bertambah berdasarkan tinggi vertikal yang dicapai pemain. Highscore disimpan dalam file eksternal sehingga tetap dapat diakses pada sesi permainan berikutnya.

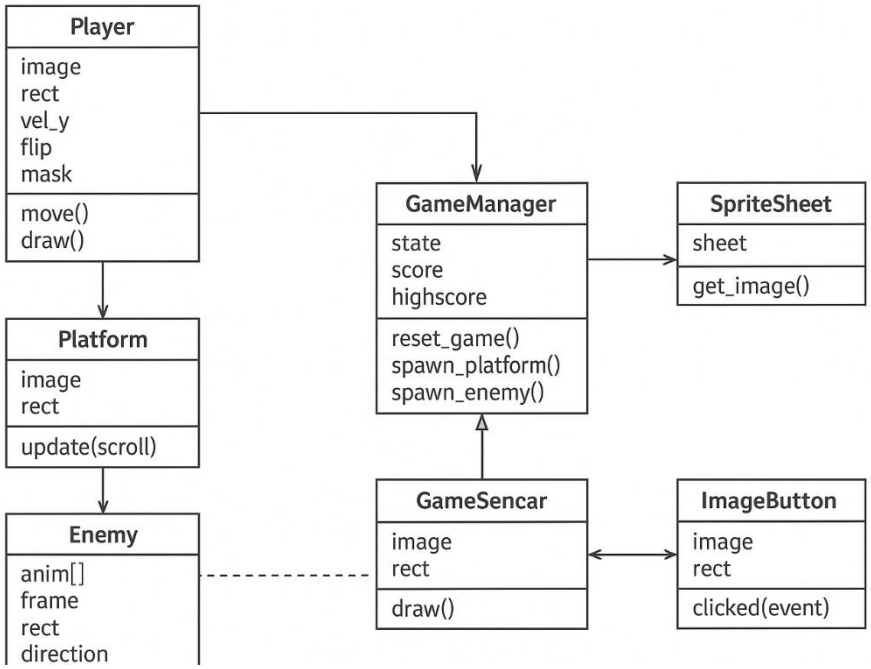
5. Musik dan Efek Suara

Music background (BGM) diputar sepanjang permainan serta efek suara akan berbunyi ketika karakter melompat dan ketika kalah.

6. Menu Utama dan Game Over

Menu utama berisi tombol Play dan Exit, sedangkan tampilan Game Over dilengkapi tombol Restart. Keduanya menggunakan objek tombol berbasis gambar (*ImageButton*).

3.3 Class Diagram



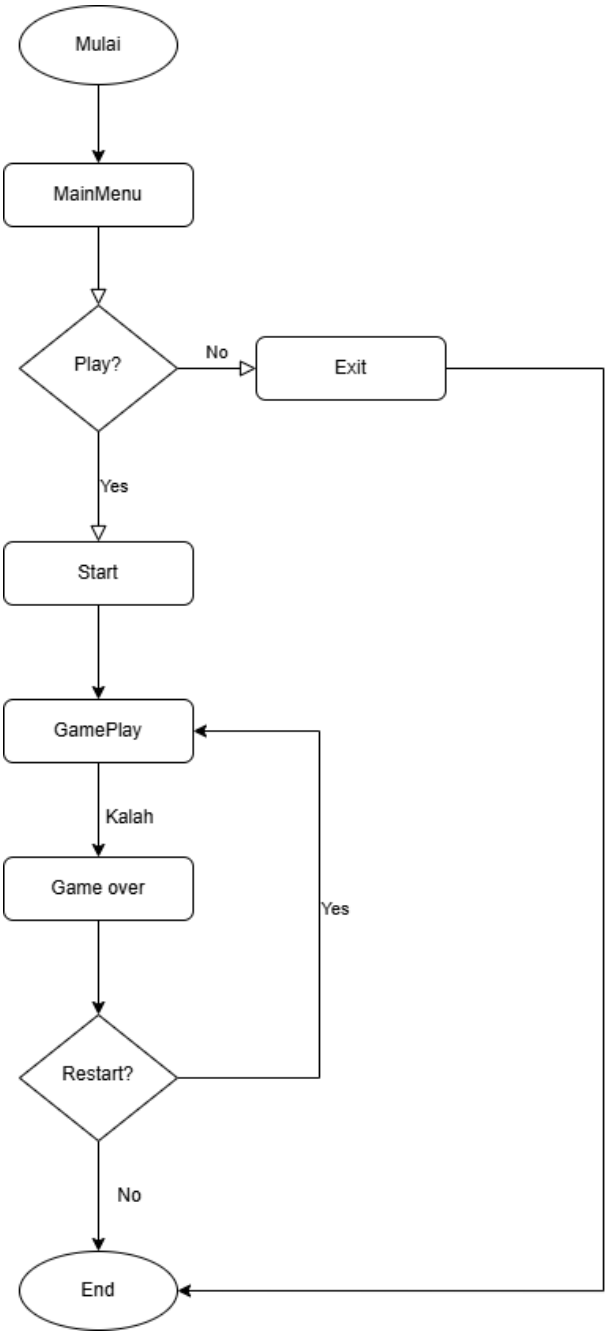
Class diagram tersebut menggambarkan struktur arsitektur utama pada game *Jumpy Kirby* yang dikembangkan menggunakan Pygame. Kelas Player berperan sebagai representasi karakter utama yang dikendalikan oleh pengguna. Di dalamnya terdapat atribut yang mengatur tampilan, posisi, orientasi, serta mask untuk mendukung pendeteksian tabrakan. Metode seperti *move()* dan *draw()* mengatur mekanisme pergerakan dan proses penggambaran karakter pada layar. Kelas ini berinteraksi langsung dengan GameManager, yang bertindak sebagai komponen pengendali utama permainan. GameManager mengelola *state* permainan, skor, serta proses penting seperti *reset game* dan *spawn* objek platform maupun musuh. Hubungan ini menunjukkan bahwa Player bergantung pada GameManager untuk mengetahui status permainan dan menginisiasi ulang perilakunya saat terjadi perubahan keadaan.

Kelas Platform merepresentasikan elemen pijakan yang digunakan pemain untuk melompat. Platform memiliki atribut gambar dan posisi, serta metode *update(scroll)* untuk menyesuaikan pergerakan layar secara vertikal. Kelas ini berfungsi sebagai objek statis namun dinamis dalam konteks permainan karena

dapat bergeser mengikuti kondisi permainan. Sementara itu, kelas Enemy berperan sebagai hambatan yang mengancam keberlangsungan pemain. Dengan atribut animasi, frame, serta arah pergerakan, Enemy memiliki metode *update()* untuk mengatur animasi dan pergerakan musuh. Garis putus-putus antara Enemy dan GameSencar menunjukkan ketergantungan Enemy pada komponen tertentu untuk menentukan posisi atau kondisi lingkungan tanpa adanya hubungan kepemilikan langsung.

Kelas SpriteSheet berfungsi sebagai utilitas yang menyediakan kemampuan pemotongan gambar dari lembar sprite yang lebih besar. GameManager menggunakan kelas ini untuk mempersiapkan aset animasi, khususnya musuh. Pada sisi antarmuka pengguna, kelas ImageButton bertanggung jawab memvisualisasikan tombol menu menggunakan gambar. Dengan atribut *image* dan *rect*, serta metode *clicked(event)*, kelas ini memudahkan pengelolaan interaksi pengguna pada menu maupun layar akhir permainan. Kelas GameSencar pada diagram menunjukkan hubungan generalisasi ke GameManager, yang menandakan adanya turunan atau perluasan fungsi tertentu, meskipun peran spesifiknya lebih mengarah sebagai komponen tampilan atau layar permainan.

3.4 Flowchart



Flowchart dimulai dari simpul Mulai yang merepresentasikan tahap inisialisasi aplikasi. Pada langkah ini sistem melakukan persiapan awal seperti inisialisasi Pygame, pemuatan aset (gambar, suara, font), pembacaan highscore, dan pengaturan parameter game (resolusi layar, FPS, nilai konstanta fisika). Setelah inisialisasi selesai, kontrol dialihkan ke MainMenu, yaitu antarmuka awal yang

menampilkan opsi utama (Play dan Exit) serta informasi pendukung seperti best score.

Pada percabangan keputusan bertanda “Play?” diagram memodelkan interaksi pengguna terhadap menu. Jika pengguna memilih *Play* (jawab *Yes*), alur berpindah ke blok Start yang berfungsi untuk mereset atau mempersiapkan kondisi permainan (reset skor, spawn platform awal, posisi player, mulai BGM). Jika pengguna memilih *Exit* (jawab *No*), alur menuju blok Exit yang mewakili terminasi aplikasi—pada implementasi nyata blok ini akan memicu prosedur pembersihan sumber daya (stop audio, simpan highscore bila perlu, pygame.quit()) dan kemudian berakhir.

Bagian inti flowchart adalah blok GamePlay yang merepresentasikan game loop. Di sini terjadi siklus berulang: pemrosesan event input, pembaruan logika (pergerakan pemain, spawn platform/enemy, collision detection), rendering, dan kontrol frame rate. Diagram menandai transisi ke blok Game over apabila kondisi kegagalan terpenuhi (label “Kalah”), misalnya ketika pemain jatuh keluar layar atau mengalami tabrakan fatal dengan musuh. Blok Game over menampilkan hasil permainan (skor akhir, highscore update) dan memberi pilihan untuk mengulang.

Keputusan Restart? pada layar Game Over menentukan apakah siklus permainan kembali dimulai atau dihentikan. Jawaban *Yes* mengarahkan kembali ke blok GamePlay melalui jalur “Start” sehingga world di-reset dan game loop berjalan ulang; jawaban *No* mengarahkan ke simpul End, menandai penghentian aplikasi. Struktur ini membentuk mesin status sederhana (finite state machine) dengan state utama: Menu → Play → GameOver → (Restart/Exit). Desain tersebut memastikan pemisahan tanggung jawab: antarmuka (menu), orkestrasi (reset/start), loop permainan (GamePlay), dan finalisasi (End).

BAB IV

IMPLEMENTASI

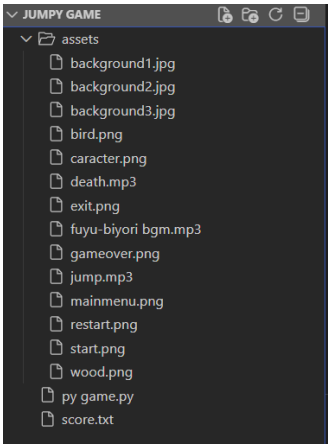
4.1 Lingkungan Pengembangan

Implementasi aplikasi *Jumpy Kirby* dilakukan menggunakan perangkat lunak yang mendukung pengembangan berbasis Python dan library multimedia untuk pembuatan game. Bahasa pemrograman yang digunakan adalah Python 3.11, yang dipilih karena memiliki sintaks yang ringkas, sistem modul yang fleksibel, serta dukungan luas terhadap pemrograman berorientasi objek. Untuk menangani proses grafis, audio, event handling, dan pengelolaan loop permainan, digunakan library Pygame versi 2.6.1. Versi ini dipilih karena stabilitasnya, kompatibilitas dengan Python versi terbaru, serta ketersediaan fitur pendukung seperti manipulasi gambar, mixer audio, sprite management, dan sistem event yang efisien.

Lingkungan pengembangan diimplementasikan melalui Visual Studio Code (VS Code) sebagai IDE utama, karena menyediakan ekosistem ekstensi yang lengkap seperti Python LSP, pengelolaan virtual environment, dan integrasi debugging. Sistem operasi yang digunakan adalah Windows 10/11, yang kompatibel dengan modul grafis SDL milik Pygame sehingga memudahkan eksekusi aplikasi tanpa konfigurasi tambahan. Dengan kombinasi ini, proses implementasi dapat dilakukan secara terstruktur, efisien, dan kompatibel dengan seluruh komponen perangkat lunak yang terlibat.

4.2 Struktur Folder Program

Struktur direktori pada aplikasi *Jumpy Kirby* disusun untuk memisahkan aset multimedia dari kode program, serta mempermudah pemeliharaan dan pengembangan lanjutan. Secara umum, struktur folder tersusun sebagai berikut:



Folder assets/ menyimpan seluruh elemen multimedia seperti gambar background, sprite karakter, sprite musuh, platform, ikon menu, serta file musik dan efek suara. File py_game.py berfungsi sebagai pusat eksekusi yang memuat seluruh kelas, logika gameplay, pengaturan state, dan inisialisasi aset. Sementara itu, file score.txt digunakan untuk menyimpan highscore sehingga data dapat dipertahankan antar sesi permainan.

Pemisahan struktur ini memudahkan proses pembaruan aset tanpa harus memodifikasi kode utama, sekaligus memberikan standar organisasi direktori yang lebih profesional dan mudah dikembangkan.

4.3 Implementasi Kode dan Penjelasan

Pada tahap implementasi, pemrograman berorientasi objek (Object-Oriented Programming) digunakan untuk memodelkan entitas penting dalam permainan seperti pemain, musuh, platform, dan elemen interaktif lainnya.

Berikut kode dan penjelasannya:

```
1   import pygame
2   import random
3   import os
4   from pygame import mixer
5   import sys
6
```

Potongan kode ini merupakan bagian awal dari sebuah game berbasis Pygame. Baris import pygame digunakan untuk memanggil library utama yang menangani grafik, animasi, input keyboard/mouse, dan tampilan game. Modul random diimpor untuk menghasilkan nilai acak, misalnya posisi platform atau musuh dalam permainan. Selanjutnya, os digunakan untuk mengelola path file seperti gambar dan audio agar program dapat memuat asset dari folder yang benar. Bagian from pygame import mixer memanggil fitur pengolahan suara, sehingga game bisa memainkan background music dan efek suara saat event tertentu terjadi. Terakhir, import sys digunakan untuk menghentikan program secara aman, terutama ketika terjadi kesalahan fatal seperti asset tidak ditemukan.

```
6
7   # ================== SAFE PATH HELPERS ==================
8   BASE_DIR = os.path.dirname(os.path.abspath(__file__))
9   ASSETS_DIR = os.path.join(BASE_DIR, "assets")
10
11  def asset(file):
12      full = os.path.join(ASSETS_DIR, file)
13      if not os.path.exists(full):
14          print("Asset missing:", full)
15          sys.exit(1)
16      return full
17
```

Potongan kode ini digunakan untuk memastikan game dapat menemukan semua file asset seperti gambar, musik, dan suara. Variabel BASE_DIR mengambil lokasi folder tempat file Python berada menggunakan os.path.abspath(__file__), sedangkan ASSETS_DIR menggabungkannya dengan folder "assets" sehingga program mengetahui lokasi penyimpanan file pendukung game.

Fungsi asset(file) berfungsi sebagai helper untuk menghasilkan path lengkap ke setiap file asset. Di dalamnya, os.path.exists() mengecek apakah file benar-benar ada. Jika tidak ditemukan, program menampilkan pesan error Asset missing dan menghentikan permainan dengan sys.exit(1) agar tidak terjadi crash saat game berjalan.

```
17
18  # ================== LOAD / SAVE HIGHSCORE ==================
19  def load_highscore():
20      if not os.path.exists("score.txt"):
21          return 0
22      with open("score.txt", "r") as f:
23          return int(f.read())
24
25  def save_highscore(new_score):
26      with open("score.txt", "w") as f:
27          f.write(str(new_score))
28
29  highscore = load_highscore()
```

Kode ini digunakan untuk mengelola data skor tertinggi (high score) pada sebuah game. Fungsi load_highscore() berfungsi membaca nilai skor yang tersimpan di file bernama *score.txt*. Pertama, fungsi ini memeriksa apakah file tersebut sudah ada menggunakan os.path.exists(). Jika file tidak ditemukan, fungsi langsung mengembalikan nilai 0, menandakan belum ada high score sebelumnya. Namun jika file tersedia, program membuka file tersebut dalam mode baca ("r") dan mengembalikan nilai yang terdapat di dalamnya sebagai tipe integer.

Sebaliknya, fungsi save_highscore(new_score) digunakan untuk menyimpan skor baru ketika pemain mendapatkan rekor tertinggi. Fungsi ini membuka file score.txt dalam mode tulis ("w") dan menulis nilai skor terbaru ke dalamnya. Variabel highscore = load_highscore() di akhir digunakan untuk memanggil fungsi pembaca skor dan menyimpan nilainya ke dalam variabel highscore agar dapat ditampilkan pada tampilan awal atau selama permainan berlangsung.

```
31  # ================== INITIAL SETUP ==================
32  pygame.init()
33  mixer.init()
34
35  SCREEN_WIDTH = 400
36  SCREEN_HEIGHT = 600
37  screen = pygame.display.set_mode((SCREEN_WIDTH, SCREEN_HEIGHT))
38  pygame.display.set_caption("Jumpy Game OOP")
39
40  clock = pygame.time.Clock()
41  FPS = 60
42
43  WHITE = (255,255,255)
44  BLACK = (0,0,0)
45
46  font_small = pygame.font.SysFont("Lucida Sans", 20)
47  font_big = pygame.font.SysFont("Lucida Sans", 28)
48  font_title = pygame.font.SysFont("Lucida Sans", 40)
49
```

Bagian kode ini merupakan proses **inisialisasi awal** untuk menjalankan game menggunakan library Pygame. Perintah pygame.init() dan mixer.init() menjalankan modul inti grafis dan modul audio, sehingga game dapat menampilkan gambar, animasi, serta memutar musik atau efek suara. Selanjutnya, SCREEN_WIDTH dan SCREEN_HEIGHT menentukan ukuran jendela permainan, yaitu 400×600 piksel. Nilai tersebut kemudian digunakan oleh pygame.display.set_mode() untuk membuat layar game yang disimpan dalam variabel screen. Perintah pygame.display.set_caption("Jumpy Game OOP") memberikan judul pada jendela permainan agar tampil profesional dan mudah dikenali.

Objek clock = pygame.time.Clock() digunakan untuk mengatur kecepatan jalannya frame game, dan nilai FPS = 60 menandakan bahwa game akan berusaha berjalan pada 60 frame per detik agar animasi terlihat halus. Selain itu, terdapat pula definisi warna dasar seperti WHITE dan BLACK dalam format RGB yang akan digunakan untuk tampilan teks atau elemen grafis lainnya. Bagian terakhir adalah pembuatan tiga ukuran font berbeda menggunakan pygame.font.SysFont(), sehingga game dapat menampilkan teks dengan ukuran kecil, besar, maupun judul sesuai kebutuhan tampilan.

```
# ================= LOAD MEDIA =================
pygame.mixer.music.load(asset("fuyu-biyori bgm.mp3"))
pygame.mixer.music.play(-1)
jump_fx = pygame.mixer.Sound(asset("jump.mp3"))
death_fx = pygame.mixer.Sound(asset("death.mp3"))

jumpy_image    = pygame.image.load(asset("caracter.png")).convert_alpha()
platform_image = pygame.image.load(asset("wood.png")).convert_alpha()
bird_sheet_img = pygame.image.load(asset("bird.png")).convert_alpha()

menu_bg = pygame.transform.scale(pygame.image.load(asset("mainmenu.png")), (SCREEN_WIDTH, SCREEN_HEIGHT))
game_over_bg = pygame.transform.scale(pygame.image.load(asset("gameover.png")), (SCREEN_WIDTH, SCREEN_HEIGHT))

background_day     = pygame.transform.scale(pygame.image.load(asset("background1.jpg")), (SCREEN_WIDTH, SCREEN_HEIGHT))
background_evening = pygame.transform.scale(pygame.image.load(asset("background2.jpg")), (SCREEN_WIDTH, SCREEN_HEIGHT))
background_night   = pygame.transform.scale(pygame.image.load(asset("background3.jpg")), (SCREEN_WIDTH, SCREEN_HEIGHT))
```

```
# ================= BUTTON IMAGES =================
start_img   = pygame.transform.scale(pygame.image.load(asset("start.png")).convert_alpha(), (220, 80))
exit_img    = pygame.transform.scale(pygame.image.load(asset("exit.png")).convert_alpha(), (220, 80))
restart_img = pygame.transform.scale(pygame.image.load(asset("restart.png")).convert_alpha(), (220, 80))
```

Potongan kode ini berfungsi untuk memuat seluruh asset visual dan audio yang diperlukan dalam game. Bagian pertama memuat musik background serta efek suara. Perintah pygame.mixer.music.load() mengambil file musik utama permainan, dan pygame.mixer.music.play(-1) memutarnya secara berulang tanpa henti selama game berlangsung. Efek suara lompatan dan kematian dimuat menggunakan pygame.mixer.Sound(), sehingga efek tersebut dapat dipanggil secara instan saat event tertentu terjadi.

Selanjutnya, kode memuat gambar karakter, platform, dan sprite musuh menggunakan pygame.image.load(), kemudian diubah menjadi format transparan dengan convert_alpha() agar tampil lebih halus di atas background. File background untuk menu, layar game over, dan tiga kondisi waktu permainan (siang, sore, malam) juga dimuat, lalu diubah ukurannya menggunakan pygame.transform.scale() agar sesuai dengan ukuran layar (SCREEN_WIDTH × SCREEN_HEIGHT). Hal ini memastikan tampilan game tetap proporsional dan tidak terdistorsi.

Bagian terakhir memuat gambar-gambar tombol seperti Start, Exit, dan Restart. Sama seperti asset lain, gambar tombol di-scale ulang ke ukuran (220, 80) agar terlihat konsisten dan mudah diklik pada tampilan menu.

```
71
72  # ================= BUTTON CLASS =================
73  class ImageButton:
74      def __init__(self, x, y, image):
75          self.image = image
76          self.rect = self.image.get_rect(center=(x, y))
77
78      def draw(self):
79          screen.blit(self.image, self.rect)
80
81      def clicked(self, event):
82          return (event.type == pygame.MOUSEBUTTONDOWN and event.button == 1 and self.rect.collidepoint(event.pos))
83
```

Kode di atas mendefinisikan sebuah **class** bernama ImageButton, yang digunakan untuk membuat tombol berbasis gambar pada game. Bagian ini sudah menerapkan konsep OOP (Object-Oriented Programming), karena tombol dibuat sebagai objek dengan atribut dan fungsi (method) sendiri. Pada method __init__, class menerima parameter x, y, dan image. Nilai koordinat digunakan untuk menentukan posisi pusat tombol pada layar, sedangkan gambar tombol disimpan ke dalam atribut self.image. Kemudian, tombol dibuatkan area interaktif berupa self.rect menggunakan get_rect(center=(x, y)), sehingga game dapat mendeteksi area yang bisa diklik pemain. Method draw() bertanggung jawab untuk menampilkan gambar tombol pada layar menggunakan screen.blit(). Sementara itu, method clicked() digunakan untuk mengecek apakah tombol diklik. Fungsi ini mengembalikan nilai True jika event mouse yang terjadi adalah klik kiri (event.button == 1) dan posisi klik berada di dalam area tombol (self.rect.collidepoint(event.pos)). Dengan adanya class ini, setiap tombol seperti Start, Exit, atau Restart dapat dibuat sebagai objek yang berdiri sendiri, sehingga kode menjadi lebih terstruktur, mudah dikelola, dan mengikuti prinsip encapsulation dalam OOP.

```
83
84  # ================= BUTTON INSTANCE =================
85  start_button   = ImageButton(SCREEN_WIDTH//2, 410, start_img)
86  quit_button    = ImageButton(SCREEN_WIDTH//2, 480, exit_img)
87  restart_button = ImageButton(SCREEN_WIDTH//2, 380, restart_img)
88
```

Kode ini membuat tiga objek tombol dengan menggunakan class ImageButton yang telah didefinisikan sebelumnya. Tombol Start, Quit, dan Restart masing-masing dibuat dengan memberikan posisi tengah

secara horizontal (SCREEN_WIDTH//2) dan koordinat vertikal yang berbeda agar tersusun rapi di layar. Parameter terakhir yaitu start_img, exit_img, dan restart_img adalah gambar tombol yang akan ditampilkan.

Karena ImageButton merupakan sebuah class, pembuatan objek ini adalah penerapan konsep Object-Oriented Programming (OOP), di mana setiap tombol menjadi objek mandiri yang memiliki fungsi draw() untuk menampilkan dan clicked() untuk mendeteksi klik. Dengan pendekatan OOP, penanganan tombol lebih rapi dan mudah dikembangkan tanpa mengulang kode secara berulang.

```
89
90  # ================= CONSTANTS =================
91  SCROLL_THRESH = 200
92  GRAVITY = 0.75
93  JUMP_VELOCITY = -14
94  MAX_FALL = 10
95  HORIZ_SPEED = 5
96  MAX_PLATFORMS = 10
```

Kode ini berisi sekumpulan konstanta yang digunakan untuk mengatur perilaku fisik dan mekanik permainan..

```
# ================== GAME STATE ==================
scroll = 0
score = 0.0
bg_scroll = 0
game_over = False
game_state = "menu"

platform_group = pygame.sprite.Group()
enemy_group = pygame.sprite.Group()
```

Bagian kode ini berfungsi untuk menginisialisasi variabel awal yang digunakan dalam permainan.

```python
# ================== SPRITES ====================
class SpriteSheet:
    def __init__(self, sheet):
        self.sheet = sheet

    def get_image(self, frame, w, h, scale, color):
        img = pygame.Surface((w,h), pygame.SRCALPHA)
        img.blit(self.sheet, (0,0), (frame*w,0,w,h))
        img = pygame.transform.scale(img, (int(w*scale), int(h*scale)))
        img.set_colorkey(color)
        return img
```

Kode tersebut mendefinisikan sebuah kelas bernama SpriteSheet yang berfungsi untuk mengelola dan mengambil gambar animasi dari satu file spritesheet. Pada konsep OOP, kelas ini memiliki sebuah constructor __init__() yang menyimpan objek gambar spritesheet ke dalam atribut self.sheet sehingga dapat digunakan berulang kali tanpa perlu memuat ulang gambar. Metode get_image() digunakan untuk memotong bagian tertentu dari spritesheet berdasarkan indeks frame yang ditentukan. Proses ini dilakukan dengan membuat surface baru sebagai tempat gambar, kemudian memotong area pada posisi (frame * w) yang sesuai dengan frame ke-berapa yang ingin digunakan. Setelah itu gambar diperbesar menggunakan pygame.transform.scale sesuai nilai scale, dan set_colorkey digunakan untuk menghapus background tertentu sehingga sprite menjadi transparan. Hasil pemotongan tersebut kemudian dikembalikan sebagai satu frame animasi siap pakai dalam game. Penggunaan class ini membuat proses animasi lebih efisien dan terstruktur, karena setiap frame dapat diambil dengan mudah melalui method get_image() tanpa memerlukan banyak file gambar terpisah.

```python
class Player:
    def __init__(self,x,y):
        self.image = pygame.transform.scale(jumpy_image,(45,45))
        self.rect = self.image.get_rect(center=(x,y))
        self.vel_y = 0
        self.flip = False
        self.mask = pygame.mask.from_surface(self.image)
```

Kode tersebut mendefinisikan sebuah class bernama Player, yang merupakan penerapan konsep Object-Oriented Programming (OOP), di mana karakter pemain direpresentasikan sebagai sebuah objek mandiri dengan atribut dan perilaku sendiri. Method __init__() adalah constructor yang otomatis dijalankan saat objek Player dibuat. Di dalamnya, gambar karakter di-scale ke ukuran 45x45 piksel, lalu dibuat area bounding (self.rect) yang menentukan posisi dan ukuran karakter pada layar, menggunakan koordinat pusat (x, y) yang diberikan. Variabel self.vel_y digunakan untuk menyimpan kecepatan vertikal (untuk efek lompat dan gravitasi), sementara self.flip menentukan apakah gambar perlu dibalik saat bergerak ke kiri. Selain itu, self.mask dibuat dari gambar pemain menggunakan pygame.mask.from_surface, yang berguna untuk deteksi tabrakan yang lebih akurat (pixel-perfect collision). Secara keseluruhan, class ini mempermudah pengelolaan data dan perilaku karakter dalam game melalui pendekatan OOP yang terstruktur dan mudah dikembangkan.

```
def move(self):
    global scroll
    dx, dy = 0, 0
    keys = pygame.key.get_pressed()

    if keys[pygame.K_LEFT] or keys[pygame.K_a]:  dx = -HORIZ_SPEED; self.flip = True
    if keys[pygame.K_RIGHT] or keys[pygame.K_d]: dx = HORIZ_SPEED;  self.flip = False

    self.vel_y += GRAVITY
    if self.vel_y > MAX_FALL: self.vel_y = MAX_FALL
    dy += self.vel_y

    if self.rect.left + dx < 0: dx = -self.rect.left
    if self.rect.right + dx > SCREEN_WIDTH: dx = SCREEN_WIDTH - self.rect.right
```

Kode move(self) merupakan method dalam class Player, yang menunjukkan penerapan konsep OOP karena fungsi ini mengatur perilaku objek pemain secara langsung. Pada awalnya, variabel dx dan dy digunakan untuk menyimpan perubahan posisi pemain secara horizontal dan vertikal. Input keyboard dibaca menggunakan pygame.key.get_pressed(), dan jika tombol kiri (LEFT atau A) ditekan maka pemain bergerak ke kiri dengan kecepatan HORIZ_SPEED, serta

self.flip diset True agar sprite menghadap ke kiri. Sebaliknya, jika tombol kanan (RIGHT atau D) ditekan, pemain bergerak ke kanan dan arah sprite disesuaikan.

Setiap frame, gravitasi ditambahkan ke self.vel_y, menyebabkan pemain jatuh secara bertahap. Nilai self.vel_y dibatasi oleh MAX_FALL untuk mencegah kecepatan jatuh terlalu besar. Selanjutnya dy ditambahkan sehingga menentukan pergerakan vertikal. Kode juga memastikan pemain tidak keluar batas layar secara horizontal, dengan mengecek jika posisi kiri atau kanan melewati batas window lalu mengatur ulang dx agar tetap di dalam area permainan.

```
    for p in platform_group:
        if p.rect.colliderect(self.rect.x, self.rect.y + dy, self.rect.width, self.rect.height):
            if self.vel_y > 0 and self.rect.bottom <= p.rect.centery:
                self.rect.bottom = p.rect.top
                dy = 0
                self.vel_y = JUMP_VELOCITY
                jump_fx.play()

    if self.rect.top <= SCROLL_THRESH and self.vel_y < 0:
        scroll = -dy
    else:
        scroll = 0

    self.rect.x += dx
    self.rect.y += dy + scroll
    self.mask = pygame.mask.from_surface(self.image)
    return scroll

def draw(self):
    screen.blit(pygame.transform.flip(self.image,self.flip,False),self.rect)
```

Bagian for p in platform_group: melakukan pengecekan tabrakan antara pemain dan setiap platform pada grup platform_group. Fungsi p.rect.colliderect() digunakan untuk memeriksa apakah posisi pemain setelah bergerak ke bawah (dy) akan bersentuhan dengan platform. Kondisi self.vel_y > 0 memastikan tabrakan hanya dihitung saat pemain sedang jatuh, bukan saat melompat ke atas. Sementara itu, self.rect.bottom <= p.rect.centery memastikan tabrakan terjadi tepat di atas platform. Jika tabrakan valid, pemain diposisikan tepat di atas platform (self.rect.bottom = p.rect.top), gerakan vertikal dihentikan (dy = 0), dan pemain melakukan lompatan baru dengan mengatur self.vel_y menjadi nilai JUMP_VELOCITY, disertai efek suara lompatan

jump_fx.play(). Selanjutnya, logika pengguliran layar (scroll) diatur agar layar bergerak ke bawah hanya ketika pemain melompat melewati batas bagian atas layar (self.rect.top <= SCROLL_THRESH) dan sedang bergerak ke atas. Posisi pemain diperbarui menggunakan self.rect.x += dx dan self.rect.y += dy + scroll. Mask diperbarui untuk mendukung deteksi tabrakan pixel-perfect. Method draw(self) bertanggung jawab menggambar sprite pemain ke layar dan membalik gambarnya jika menghadap kiri sesuai variabel self.flip.

```
class Platform(pygame.sprite.Sprite):
    def __init__(self,x,y,width):
        super().__init__()
        self.image = pygame.transform.scale(platform_image,(width,12))
        self.rect = self.image.get_rect(topleft=(x,y))

    def update(self,scroll):
        self.rect.y += scroll
        if self.rect.top > SCREEN_HEIGHT:
            self.kill()
```

Potongan kode ini menerapkan konsep OOP (Object-Oriented Programming) melalui penggunaan class Platform yang mewakili objek platform di dalam game. Class ini mewarisi (extends) dari pygame.sprite.Sprite menggunakan super().__init__(), yang berarti platform menjadi bagian dari sistem sprite Pygame sehingga dapat dikelola dalam group sprite seperti platform_group().

Pada method __init__(), platform dibuat sebagai gambar (self.image) yang diambil dari asset platform_image kemudian diubah ukurannya sesuai parameter width menggunakan pygame.transform.scale(). Posisi platform ditentukan menggunakan get_rect(topleft=(x,y)), sehingga objek mengetahui posisi dan area tabrakan (rect) di layar.

Method update(self, scroll) mengatur pergerakan platform mengikuti nilai scroll, sehingga platform bergerak turun ketika pemain naik. Jika posisi platform melewati bagian bawah layar (self.rect.top > SCREEN_HEIGHT), platform dihancurkan menggunakan self.kill(), yang merupakan method bawaan dari sprite group untuk menghapus objek agar tidak memenuhi memori dan layar.

```python
class Enemy(pygame.sprite.Sprite):
    def __init__(self,x,y,direction,sheet):
        super().__init__()
        self.anim = [SpriteSheet(sheet).get_image(i,32,32,1.5,(0,0,0)) for i in range(8)]
        self.frame=0
        self.image=self.anim[self.frame]
        self.rect=self.image.get_rect(center=(x,y))
        self.direction=direction
        self.anim_timer=0
        self.mask = pygame.mask.from_surface(self.image)

    def update(self,scroll):
        self.anim_timer+=1
        if self.anim_timer>6:
            self.anim_timer=0
            self.frame=(self.frame+1)%len(self.anim)
            self.image=self.anim[self.frame]
            self.mask=pygame.mask.from_surface(self.image)

        self.rect.x += self.direction*2
        self.rect.y += scroll
        if self.rect.right<0 or self.rect.left>SCREEN_WIDTH or self.rect.top>SCREEN_HEIGHT:
            self.kill()
```

Kode di atas menggunakan konsep OOP melalui pembuatan class Enemy yang merupakan turunan dari pygame.sprite.Sprite. Pewarisan ini digunakan agar objek musuh dapat dimasukkan dalam enemy_group dan otomatis mendapat fitur sprite seperti update dan penghapusan (kill()). Pada method __init__(), musuh diinisialisasi dengan membuat daftar animasi (self.anim) yang berisi 8 frame gambar yang diambil dari sprite sheet menggunakan method get_image() milik class SpriteSheet. Variabel self.frame menentukan frame animasi yang sedang dipakai, dan self.image adalah gambar yang akan ditampilkan. Objek kemudian mendapatkan posisi (rect) berdasarkan koordinat (x,y) serta arah gerak (self.direction) untuk bergerak ke kanan atau ke kiri. Method update() menjalankan animasi dan pergerakan musuh. self.anim_timer berfungsi sebagai kontrol kecepatan animasi; bila melewati batas tertentu, frame akan berubah sehingga gambar terlihat hidup (bergerak sayap). Musuh bergerak horizontal dengan self.rect.x += self.direction*2 dan ikut bergeser vertikal sesuai scroll layar. Jika musuh keluar dari area layar, method bawaan kill() dipanggil untuk menghapus objek dari game agar tidak membebani memori.

```python
def get_game_bg():
    if score < 200: return background_day
    if score < 800: return background_evening
    return background_night
```

Fungsi get_game_bg() digunakan untuk menentukan gambar latar (background) yang akan ditampilkan selama permainan berlangsung. Fungsi ini mengevaluasi nilai score pemain, kemudian mengembalikan background yang sesuai dengan tingkat pencapaian skor. Jika skor masih di bawah 200, permainan menampilkan background siang (background_day). Ketika skor mencapai antara 200 hingga 799, background berubah menjadi senja (background_evening). Setelah melewati 800, background berubah menjadi malam (background_night).

```python
def reset_game():
    global platform_group, enemy_group, player, score, bg_scroll, game_over
    score = 0.0
    bg_scroll = 0
    game_over = False
    platform_group = pygame.sprite.Group()
    enemy_group = pygame.sprite.Group()

    first_y = SCREEN_HEIGHT - 80
    first = Platform(SCREEN_WIDTH//2 - 50, first_y, 100)
    platform_group.add(first)

    y = first_y - 65
    prev_x = SCREEN_WIDTH//2
    for i in range(8):
        pw = random.randint(50,90)
        x = max(40, min(prev_x + random.randint(-120,120), SCREEN_WIDTH - pw - 40))
        platform_group.add(Platform(x, y, pw))
        prev_x = x
        y -= 65

    player = Player(SCREEN_WIDTH//2, first_y - 20)

reset_game()
```

Fungsi reset_game() digunakan untuk mengatur ulang seluruh kondisi permainan setiap kali pemain memulai baru atau menekan tombol restart. Di awal fungsi, beberapa variabel global seperti score, bg_scroll, dan game_over dikembalikan ke nilai awal. Selain itu, grup sprite platform_group dan enemy_group dibuat kembali kosong

menggunakan pygame.sprite.Group(), sehingga semua objek musuh dan platform sebelumnya dihapus.

Selanjutnya, fungsi ini membuat platform pertama yang diletakkan di bagian bawah layar (first_y) sebagai pijakan awal pemain. Platform tersebut dimasukkan ke dalam grup platform dengan memanggil constructor kelas Platform, yang merupakan penerapan konsep OOP, karena objek dibuat dari kelas dan disimpan dalam grup sprite sebagai entitas mandiri.

Setelah platform utama dibuat, fungsi menghasilkan beberapa platform tambahan di atasnya menggunakan perulangan. Posisi horizontal platform dibuat dinamis dan tidak terlalu jauh dari platform sebelumnya untuk menjaga agar pemain tetap dapat melompat ke platform berikutnya. Nilai prev_x menyimpan posisi platform terakhir untuk membantu menentukan jarak yang aman.

Terakhir, objek pemain (player) dibuat dari kelas Player dan ditempatkan tepat di atas platform awal. Pemanggilan reset_game() di akhir memastikan game langsung memulai dengan kondisi yang sudah disiapkan.

```
# ================= GAME LOOP =================
run = True
while run:
    clock.tick(FPS)

    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            run=False

        if game_state=="menu":
            if start_button.clicked(event):
                game_state="play"
                reset_game()
            if quit_button.clicked(event):
                run=False

        if game_over:
            if restart_button.clicked(event):
                reset_game()
                game_over=False
                game_state="play"
```

Potongan kode tersebut merupakan bagian utama dari loop game (while run:) yang terus berjalan selama variabel run bernilai True. Di dalam loop, clock.tick(FPS) digunakan untuk menjaga kecepatan frame agar permainan tetap stabil sesuai nilai FPS yang ditentukan. Kode ini juga

menangani berbagai event dari pemain. Menggunakan for event in pygame.event.get(), program memeriksa setiap event yang terjadi, seperti menutup jendela (event pygame.QUIT) yang akan mengubah run menjadi False, sehingga permainan berhenti. Selanjutnya terdapat logika pengaturan status game berdasarkan variabel game_state. Ketika berada dalam menu (game_state == "menu"), program memeriksa apakah tombol Start atau Quit ditekan melalui fungsi clicked() pada objek ImageButton. Jika tombol start diklik, game berpindah ke mode bermain (game_state="play") dan memanggil reset_game() untuk menyiapkan ulang permainan. Jika tombol quit ditekan, permainan dihentikan. Pada kondisi game_over, tombol Restart diperiksa. Jika diklik, permainan di-reset dan status permainan berubah kembali menjadi mode main (game_state="play"). Kode ini berhubungan dengan konsep OOP, khususnya pada penggunaan objek tombol (start_button, quit_button, restart_button) yang masing-masing merupakan instance dari kelas ImageButton. Setiap objek memiliki fungsi clicked() sendiri untuk memproses input pemain, sehingga logika menjadi lebih terstruktur dan modular.

Potongan kode ini menangani tampilan **menu utama** permainan ketika variabel game_state bernilai "menu". Pertama, gambar latar menu (menu_bg) ditampilkan pada layar menggunakan screen.blit(menu_bg, (0,0)), memastikan background menu muncul penuh di layar. Kemudian, teks Best Score, yaitu skor tertinggi yang telah disimpan sebelumnya, ditampilkan di tengah layar. Proses ini menggunakan font_big.render() untuk membuat teks dan screen.blit() untuk menempatkannya di posisi tertentu. Posisi (SCREEN_WIDTH//2 - 100, 270) memastikan tulisan berada mendekati tengah secara

horizontal. Setelah itu, dua tombol yaitu start_button dan quit_button dipanggil menggunakan metode draw() untuk menggambar tombol di layar. Ini merupakan penerapan OOP karena tombol tersebut adalah objek dari kelas ImageButton, sehingga proses menggambar tombol cukup memanggil fungsi milik objek, bukan memproses gambar secara manual. Akhirnya, pygame.display.update() memperbarui tampilan layar dan continue digunakan untuk kembali ke awal loop tanpa menjalankan logika permainan lainnya. Kode ini mengatur antarmuka menu dengan sederhana namun rapi, memisahkan logika menu dari logika game utama.

```
if not game_over:
    screen.blit(get_game_bg(),(0,bg_scroll))
    screen.blit(get_game_bg(),(0,-SCREEN_HEIGHT+bg_scroll))

    sc = player.move()
    bg_scroll+=sc
    if bg_scroll>=SCREEN_HEIGHT: bg_scroll=0

    if len(platform_group)<MAX_PLATFORMS and random.random()<0.4:
        w=random.randint(50,90)
        x=random.randint(40,SCREEN_WIDTH-w-40)
        y=random.randint(-100,-20)
        platform_group.add(Platform(x,y,w))
        if random.random()<0.12:
            enemy_group.add(Enemy(x+w//2,y-40,random.choice([-1,1]),bird_sheet_img))

    platform_group.update(sc)
    enemy_group.update(sc)
```

Potongan kode ini menangani logika utama permainan saat kondisi game belum berakhir (if not game_over:). Pertama, background permainan digambar dua kali menggunakan screen.blit(get_game_bg(), ...), satu di posisi normal dan satu di atas layar. Ini menciptakan efek scrolling vertikal yang mulus mengikuti gerakan pemain. Selanjutnya, pemain bergerak melalui pemanggilan player.move(), yang mengembalikan nilai scroll (sc). Nilai ini digunakan untuk menggeser background (bg_scroll += sc). Jika posisi scroll melewati tinggi layar, maka bg_scroll di-reset ke nol agar latar terus berputar. Kode kemudian mengatur pembuatan platform baru. Jika jumlah platform kurang dari batas maksimum (MAX_PLATFORMS) dan probabilitas acak sesuai, platform baru dihasilkan pada posisi acak dan dimasukkan ke dalam platform_group. Terkadang musuh (Enemy) juga dibuat di atas

platform baru berdasarkan peluang (0.12 atau 12%). Terakhir, seluruh platform dan musuh diperbarui posisinya melalui pemanggilan platform_group.update(sc) dan enemy_group.update(sc).

```
278         platform_group.draw(screen)
279         enemy_group.draw(screen)
280         player.draw()
281
282         if sc>0: score+=1
283
284         hits = pygame.sprite.spritecollide(player,enemy_group,False)
285         for e in hits:
286             if pygame.sprite.collide_mask(player,e):
287                 game_over=True
288                 death_fx.play()
289                 if score>highscore:
290                     highscore=score
291                     save_highscore(int(highscore))
292
293         if player.rect.top>SCREEN_HEIGHT:
294             game_over=True
295             death_fx.play()
296             if score>highscore:
297                 highscore=score
298                 save_highscore(int(highscore))
299
300         screen.blit(font_small.render(f"Score: {int(score)}",True,WHITE),(10,10))
```

Potongan kode ini bertanggung jawab untuk menampilkan objek permainan dan memproses logika skor serta kondisi kalah. Pertama, semua elemen visual utama digambar ke layar: platform melalui platform_group.draw(screen), musuh enemy_group.draw(screen), dan pemain menggunakan metode player.draw(). Pemanggilan draw() pada player menunjukkan penggunaan konsep OOP karena posisi gambar pemain ditangani oleh metode milik objek. Jika nilai scroll sc lebih besar dari 0 (artinya pemain bergerak naik), maka skor (score) bertambah satu poin. Ini menciptakan gameplay berbasis ketinggian yang berhasil dicapai pemain. Selanjutnya, kode memeriksa tabrakan pemain dengan musuh memakai pygame.sprite.spritecollide(). Jika terjadi tabrakan, dicek menggunakan pygame.sprite.collide_mask() untuk memastikan tabrakan pixel-perfect. Bila benar terjadi, status permainan diubah menjadi kalah (game_over=True), efek suara kematian diputar, dan skor dibandingkan dengan skor tertinggi (highscore). Jika skor baru lebih tinggi, maka skor tertinggi diperbarui dan disimpan ke file menggunakan save_highscore(). Pemeriksaan

kondisi kalah juga dilakukan saat pemain jatuh melewati batas bawah layar (player.rect.top > SCREEN_HEIGHT), dengan logika yang sama seperti tabrakan musuh. Skor saat ini ditampilkan di pojok kiri atas layar menggunakan screen.blit() dan font kecil.

```
    else:
        screen.blit(game_over_bg,(0,0))
        score_text = font_big.render(f"Score: {int(score)}", True, WHITE)
        screen.blit(score_text, (SCREEN_WIDTH//2 - score_text.get_width()//2, 200))

        best_text = font_big.render(f"Best Score: {int(highscore)}", True, WHITE)
        screen.blit(best_text, (SCREEN_WIDTH//2 - best_text.get_width()//2, 240))

        restart_button.draw()

    pygame.display.update()

pygame.quit()
```

Potongan kode ini dijalankan ketika permainan berada pada kondisi game over, yaitu bagian else: dari logika utama. Pertama, background khusus layar game over (game_over_bg) ditampilkan untuk memberi tampilan berbeda dari gameplay. Kemudian skor terakhir pemain dirender menggunakan font_big.render() dan ditampilkan di tengah layar dengan menghitung posisi berdasarkan SCREEN_WIDTH dikurangi setengah dari lebar teks, sehingga teks benar-benar berada di tengah horizontal.

Hal yang sama dilakukan untuk menampilkan Best Score atau skor tertinggi (highscore), memberikan informasi pencapaian maksimal kepada pemain. Setelah itu, tombol restart digambar di layar melalui restart_button.draw(). Ini mencerminkan penggunaan konsep OOP karena tombol merupakan objek yang memiliki metode sendiri untuk menampilkan dirinya.

Pada akhir loop, pygame.display.update() memperbarui seluruh tampilan pada layar. Setelah permainan berakhir dan loop berhenti, pygame.quit() digunakan untuk menutup aplikasi Pygame dan membersihkan semua resource.

4.4 Integrasi Fitur Tambahan

Aplikasi *Jumpy Kirby* menerapkan beberapa fitur tambahan yang mendukung kenyamanan pengguna dan kualitas keseluruhan permainan.

1. Sound Effect dan Musik Latar

   Efek suara seperti *jump.mp3* untuk aksi melompat dan *death.mp3* saat game over meningkatkan feedback audio. Musik latar *fuyu-biyori bgm.mp3* diputar secara looping untuk menciptakan atmosfer permainan yang konsisten.

2. Leaderboard / Highscore

   Pencatatan skor tertinggi disimpan pada file *score.txt*. Sistem ini memungkinkan permainan mempertahankan skor terbaik dari sesi sebelumnya, meningkatkan motivasi pemain untuk bermain kembali.

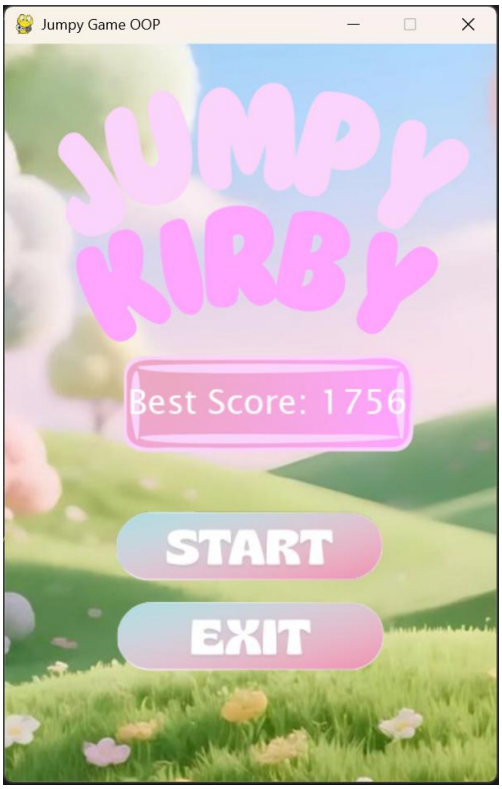
3. Animasi Sprite Sheet

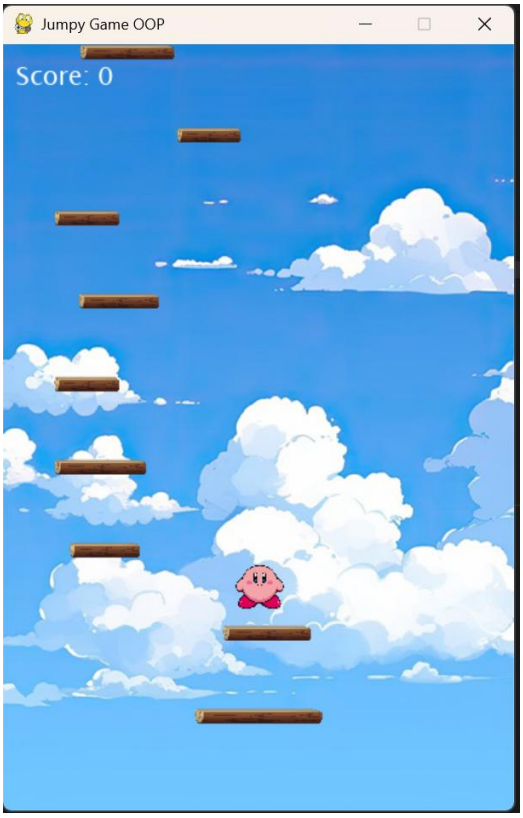
   Musuh menggunakan sprite sheet yang dipotong per frame menggunakan class *SpriteSheet*, menghasilkan animasi yang lebih halus dan dinamis.

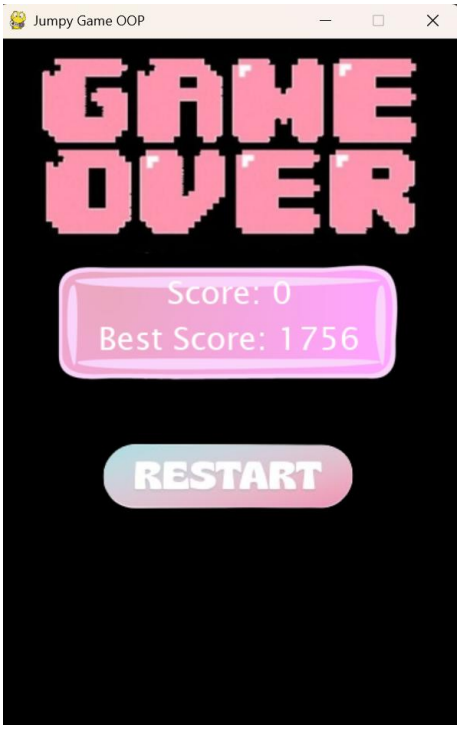
4. Menu Interaktif

   Menu utama dilengkapi tombol grafis (*ImageButton*) yang memberikan antarmuka intuitif, mencakup tombol Play, Exit, dan Restart.

4.5 Screenshot Game






Jumpy Game OOP
GAME
OVER
Score: 0
Best Score: 1756
RESTART

BAB V
PENGUJIAN DAN HASIL

5.1 Pengujian Fungsionalitas

Pengujian dilakukan menggunakan metode *black-box testing*, yaitu pengujian yang berfokus pada validasi keluaran berdasarkan masukan tanpa memeriksa struktur internal kode. Pendekatan ini sesuai untuk aplikasi permainan, karena setiap fitur dapat dievaluasi berdasarkan respons sistem terhadap interaksi pengguna. Seluruh pengujian dilakukan di lingkungan yang sama dengan proses pengembangan, yaitu Python 3.11 dengan Pygame 2.6.1 pada Windows 10.

Pengujian mencakup seluruh fitur inti permainan, mulai dari tampilan menu, pergerakan pemain, sistem deteksi tabrakan, logika game over, hingga fungsi tombol interaktif. Tabel berikut merangkum hasil pengujian secara sistematis:

No	Fitur yang Diuji	Skenario Pengujian	Hasil yang Diharapkan	Hasil Aktual	Status
1	Tampilan main menu	Aplikasi dijalankan	Menu tampil dengan tombol start & exit	Sesuai	Lulus
2	Tombol start	Tombol start diklik	Game berpindah ke gameplay	Berhasil	Lulus
3	Tombol exit	Tombol exit diklik	Aplikasi tertutup	Berhasil	Lulus
4	Pergerakan player	Arah kiri/kanan ditekan	Player bergerak sesuai input	Berfungsi	Lulus
5	Gravitasi & lompatan	Player mendarat	Player melompat otomatis	Sesuai	Lulus

		di platfrom			
6	Spawn platform	Game berjalan 5 detik	Platform baru muncul acak	Sesuai	Lulus
7	Spawn musuh	Game berjalan	Musuh muncul pada interval acak	Sesuai	Lulus
8	Deteksi tabrakan musuh	Player menabrak musuh	Game over	Sesuai	Lulus
9	Player jatuh	Player jatuh di bawah layar	Game over	Sesuai	Lulus
10	Sistem skor	Player naik ke atas	Skor bertambah	Berfungsi	Lulus
11	Highscore	Skore melebihi nilai disimpan	Highscore diperbarui	Sesuai	Lulus
12	Tombol restart	Restart diklik	Permainan dimulai kembali	Berhasil	Lulus

Hasil pengujian menunjukkan bahwa seluruh fitur utama dan tambahan berjalan sesuai spesifikasi.

5.2 Hasil Pengujian

Secara keseluruhan, *Jumpy Kirby* dapat dijalankan dengan stabil tanpa ditemukannya bug yang mengganggu jalannya permainan. Setiap elemen

grafis, audio, serta interaksi pengguna merespons secara konsisten dengan skenario gameplay yang dirancang.

1. Stabilitas Game

   Permainan mampu berjalan pada frame rate konstan (±60 FPS), tanpa lag atau kendala rendering walaupun memiliki animasi sprite, scrolling background, dan aktor bergerak. Tidak ditemukan crash selama proses pengujian.

2. Validasi Implementasi OOP

   Konsep OOP yang diterapkan dinilai telah berjalan efektif:

   - Enkapsulasi tampak pada pemisahan atribut dan metode dalam class Player, Enemy, Platform, dan ImageButton.
   - Polimorfisme terlihat pada metode *update()* pada objek musuh dan platform yang dipanggil pada loop utama.
   - Inheritansi diterapkan pada Enemy yang mewarisi struktur dasar *pygame.sprite.Sprite*.
   - Abstraksi tercermin pada penggunaan class GameManager sebagai pengontrol seluruh alur permainan.

   Cara objek saling berinteraksi juga menunjukkan bahwa struktur class dirancang secara modular, sehingga mudah diuji dan dikembangkan.

3. Respons Input dan Deteksi Tabrakan

   Sistem input keyboard merespons secara real-time, dan mekanisme collision detection berbasis mask berjalan akurat, sehingga tabrakan dengan musuh maupun platform terdeteksi secara tepat.

   Dengan demikian, hasil pengujian menyimpulkan bahwa permainan berfungsi secara optimal dan selaras dengan desain awal.

5.3 Evaluasi

Berikut adalah kelebihan, kekurangan dan potensi pengembangan dari game Jumpy kirby.

Kelebihan

Aplikasi *Jumpy Kirby* memiliki beberapa keunggulan yang mendukung pengalaman bermain, yaitu:

- Antarmuka sederhana dan intuitif, memudahkan pemain memahami cara bermain tanpa instruksi tambahan.
- Animasi dan aset visual dinamis, dengan sprite sheet musuh dan perubahan background sesuai skor.
- Penggunaan OOP yang terstruktur, sehingga memudahkan pemeliharaan dan pengembangan ke depannya.
- Gameplay responsif, dengan sistem fisika sederhana yang terasa natural.
- Sistem highscore persisten, memberikan motivasi bagi pemain untuk meningkatkan skor.

Kekurangan

Beberapa aspek masih dapat diperbaiki:

- Belum terdapat level atau variasi rintangan, sehingga tingkat kesulitan tidak berubah secara signifikan.
- Tidak ada sistem pause, sehingga permainan harus diselesaikan dalam satu sesi.
- Musuh hanya memiliki satu pola gerakan, membuat gameplay berulang setelah cukup lama.
- Pengelolaan asset masih satu file Python, belum dipisah ke beberapa modul.

Potensi Pengembangan

Permainan ini memiliki ruang pengembangan yang cukup luas, antara lain:

- Penambahan level dan mekanisme progresi (misal: speed meningkat, platform bergerak, obstacle baru).

- Pembuatan sistem power-up, seperti double-jump, shield, atau slow motion.
- Implementasi modulisasi kode, misalnya player.py, enemy.py, platform.py, game_manager.py.
- Integrasi database untuk leaderboard online.
- Penambahan main menu animasi dan pengaturan audio.
- Pembuatan versi mobile menggunakan Pygame Subset for Android (PGS4A).

BAB VI

KESIMPULAN DAN SARAN

6.1 Kesimpulan

Berdasarkan proses perancangan, implementasi, dan pengujian yang telah dilakukan, dapat disimpulkan bahwa pengembangan game *Jumpy Kirby* berhasil menunjukkan penerapan konsep *Object-Oriented Programming (OOP)* secara efektif. Setiap komponen permainan seperti pemain, musuh, platform, tombol interaktif, dan pengelola permainan, diimplementasikan dalam bentuk kelas yang terpisah, sehingga struktur program menjadi lebih modular, terorganisasi, dan mudah dipelihara. Penerapan prinsip OOP seperti *enkapsulasi, inheritance, abstraksi,* dan *polimorfisme* terbukti meningkatkan fleksibilitas dan keterbacaan kode.

Selain itu, game ini mampu berjalan sesuai dengan rancangan awal, baik dari sisi alur permainan, interaksi pengguna, maupun mekanisme gameplay seperti deteksi tabrakan, sistem skor, dan manajemen state permainan. Pengujian menunjukkan bahwa seluruh fitur inti telah berfungsi dengan baik tanpa menimbulkan kesalahan yang mengganggu jalannya aplikasi. Pemanfaatan Pygame memberi dukungan kuat untuk pengolahan grafis, audio, dan animasi sehingga pengalaman bermain terasa dinamis dan responsif.

Game *Jumpy Kirby* berhasil dikembangkan sebagai sebuah aplikasi permainan 2D sederhana yang interaktif, stabil, dan sesuai dengan tujuan pembelajaran, khususnya dalam memahami penerapan konsep OOP dalam konteks pengembangan game.

6.2 Saran

Walaupun aplikasi telah berjalan dengan baik, masih terdapat beberapa aspek yang dapat dikembangkan untuk meningkatkan kualitas permainan serta memperluas pengalaman pengguna.

1. Penambahan sistem level dapat memberikan tantangan yang lebih variatif, misalnya dengan meningkatkan kecepatan musuh, memperkenalkan jenis platform baru, atau desain arena yang berbeda. Hal ini akan memberikan pengalaman bermain yang lebih progresif dan tidak monoton.
2. Penerapan sistem inventory dapat memperkaya mekanisme permainan. Pemain dapat mengumpulkan item seperti power-up, perisai, tambahan nyawa, atau kemampuan khusus. Sistem ini tidak hanya menambah

kompleksitas gameplay, tetapi juga membuka peluang desain strategi yang lebih menarik.

3. Penambahan NPC interaktif seperti panduan, musuh dengan kecerdasan buatan yang lebih adaptif, atau karakter yang memberikan misi tertentu dapat memperluas kedalaman permainan. NPC yang interaktif dapat meningkatkan imersi dan variasi dalam permainan.
4. Pengembangan mode multiplayer dapat menjadi nilai tambah yang signifikan, baik dalam bentuk *co-op* maupun *versus*. Implementasi multiplayer akan membutuhkan sinkronisasi data, manajemen input ganda, serta logika game baru, namun memberikan pengalaman bermain yang lebih sosial dan kompetitif.

Dengan melakukan pengembangan pada aspek-aspek tersebut, game *Jumpy Kirby* memiliki potensi untuk berkembang menjadi aplikasi permainan yang lebih kompleks, menarik, dan kompetitif, sekaligus memberikan tantangan lanjutan bagi pengembang dalam menerapkan konsep-konsep lanjutan dalam pemrograman game.

DAFTAR PUSTAKA

Beazley, D., & Jones, B. (2013). *Python cookbook* (3rd ed.). O'Reilly Media.

Lutz, M. (2013). *Learning Python* (5th ed.). O'Reilly Media.

McGugan, W. (2007). *Beginning game development with Python and Pygame: From novice to professional*. Apress.

Pygame Community.(2023). Pygame documentation. https://www.pygame.org/docs/

Silva, A. (2015). *Game development using Python*. Packt Publishing.

Van Rossum, G., & Drake, F. L. (2022). *The Python language reference manual*. Python Software Foundation.

Nystrom, R. (2014). *Game programming patterns*. Genever Benning.

Suryani, S., & Pratama, A. (2020). Penerapan object-oriented programming dalam pengembangan aplikasi berbasis desktop. *Jurnal Teknologi dan Sistem Komputer*, 8(2), 112–120.

Putra, D. A. (2019). Analisis implementasi mask-based collision detection pada permainan 2D. *Jurnal Ilmu Komputer Indonesia*, 4(1), 22–30.

LAMPIRAN

Link Youtube:

https://youtu.be/ZTeBsJE8mDM

Link GitHub:

Alifahh6/Jumpy-Kirby