GAME FRUITS NINJA SEDERHANA

LAPORAN AKHIR PROYEK PEMROGRAMAN DASAR

Disusun Oleh:

Ibnu A. Zahran Latua (25031554053)

Muhammad Alvian Alpha Romeo (25031554080)

Desminica Mahasani M. (25031554190)

Dosen Pengampu:

Hasanuddin Al-Habib, S.Si., M.Si

PROGRAM STUDI SAINS DATA

FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM

UNIVERSITAS NEGERI INDONESIA

TAHUN 2025

Daftar Pustaka

Contents

BAB 1
PENDAHULUAN

1.1 Latar belakang

Pada era digital saat ini, interaksi berbasis gerakan menjadi semakin populer dalam berbagai aplikasi dan game, terutama yang memanfaatkan kamera dan kecerdasan buatan untuk mendeteksi pergerakan tubuh atau tangan pengguna (Pun & Zhu, 2011). Game sederhana dengan teknologi deteksi tangan memberikan kesempatan bagi pengembang pemula untuk memahami konsep pengolahan citra, pelacakan objek, serta integrasi antara perangkat keras kamera dengan perangkat lunak secara real-time (Pun & Zhu, 2011).

Berdasarkan hal tersebut, penulis mengembangkan sebuah game Fruit Ninja sederhana yang dimainkan hanya dengan gerakan tangan di depan kamera, tanpa memerlukan kontrol fisik seperti mouse atau joystick (Game et al., 2025). Proyek ini tidak hanya ditujukan sebagai media hiburan interaktif, tetapi juga sebagai sarana pembelajaran bagi pelajar maupun masyarakat umum untuk mengenal penerapan teknologi deteksi tangan dalam pengembangan game. Dengan adanya game ini, diharapkan pengguna dapat merasakan pengalaman bermain yang lebih imersif sekaligus memahami bahwa teknologi visi komputer dapat diimplementasikan pada aplikasi yang dekat dengan kehidupan sehari-hari.

1.2 Rumusan masalah

Penggunaan kontrol fisik seperti mouse atau joystick pada game pemotongan buah umumnya membatasi pengalaman bermain yang lebih natural dan interaktif (Fitri Barokah et al., 2024). Di sisi lain, teknologi deteksi tangan dengan bantuan kamera sudah tersedia, tetapi belum banyak dimanfaatkan dalam game sederhana sebagai media pembelajaran bagi pelajar untuk memahami penerapan visi komputer. Selain itu, belum terdapat aplikasi game Fruit Ninja sederhana yang memanfaatkan gerakan jari di depan kamera sebagai pengganti kontrol fisik, dengan tampilan antarmuka yang mudah dipahami oleh pengguna. Oleh karena itu, diperlukan sebuah aplikasi game Fruit Ninja berbasis deteksi tangan yang mampu membaca gerakan jari secara real-time, menerjemahkannya menjadi aksi memotong buah, serta menyajikan tampilan permainan yang sederhana dan nyaman

sehingga dapat memberikan hiburan sekaligus pengalaman belajar interaktif bagi pengguna.

1.3 Tujuan

1.3.1 Mengembangkan game Fruit Ninja yang dapat dimainkan menggunakan Gerakan jari di depan kamera tanpa control fisik.

1.3.2 Merancang tampilan permainan yang sederhana dan mudah dipahami agar lebih nyaman digunakan

1.3.3 Menerapkan teknologi deteksi tangan secara real-time pada game sebagai media pembelajaran interaktif.

BAB 2
ANALISIS DAN PERANCANGAN

2.1 Analisis Kebutuhan Aplikasi

Dalam pengembangan Game Fruit Ninja Berbasis Deteksi Tangan dengan Python, diperlukan analisis kebutuhan untuk menjelaskan mengapa aplikasi ini dibutuhkan serta fungsi apa saja yang harus tersedia agar pengalaman bermain menjadi interaktif dan nyaman. Analisis kebutuhan aplikasi dibagi menjadi dua bagian, yaitu kebutuhan fungsional dan kebutuhan non-fungsional, sebagaimana lazim digunakan pada pengembangan game dan aplikasi interaktif.

2.1.1 Kebutuhan Fungsional Aplikasi

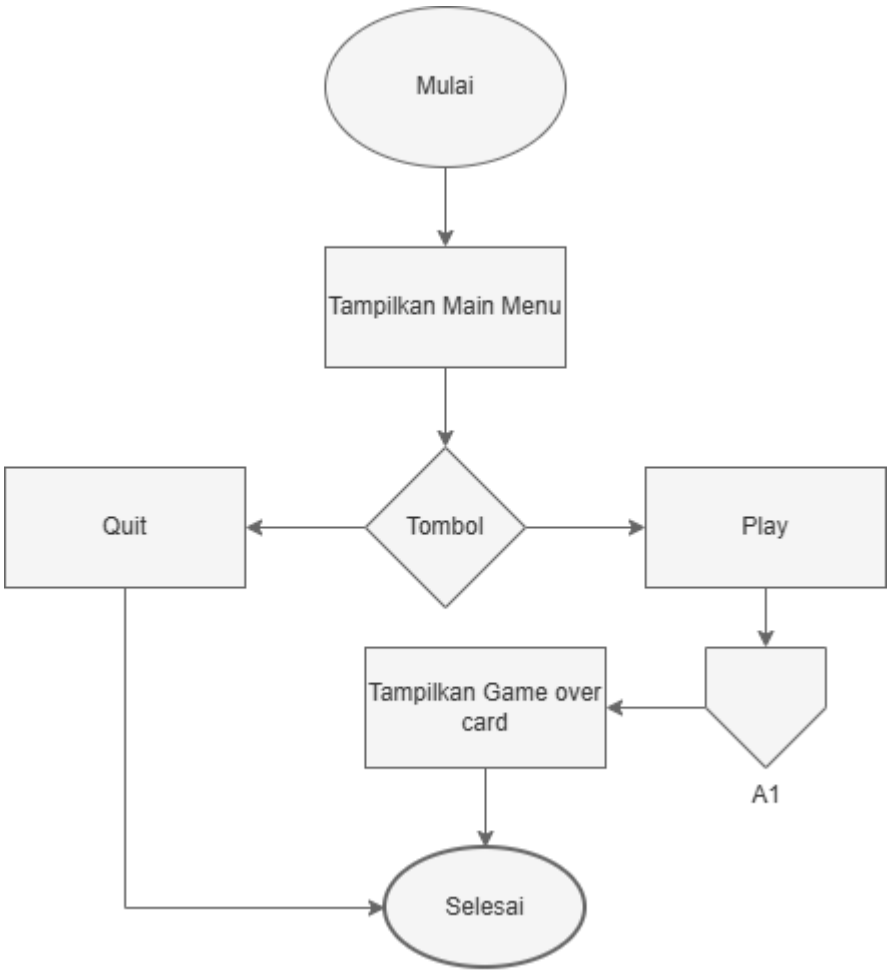
- Aplikasi mampu menangkap gambar tangan pemain melalui kamera dan mendeteksi pergerakan jari sebagai alat pemotong buah.Aplikasi menyediakan fitur input, edit, dan hapus nilai data siswa.
- Aplikasi menampilkan menu utama berisi pilihan bermain, pengaturan tema/tampilan, dan keluar dari game.
- Aplikasi menyediakan mode permainan Solo dan Multiplayer, masing-masing dengan tampilan skor dan level permainan.
- Aplikasi dapat menampilkan objek buah yang bergerak dan menghitung skor ketika buah berhasil dipotong, serta menambah jumlah “miss” ketika buah terlewat.
- Aplikasi mengatur peningkatan level atau kecepatan buah berdasarkan skor atau waktu bermain tertentu.
- Aplikasi menyediakan fitur keluar/berhenti permainan dan kembali ke menu sesuai input pemain.

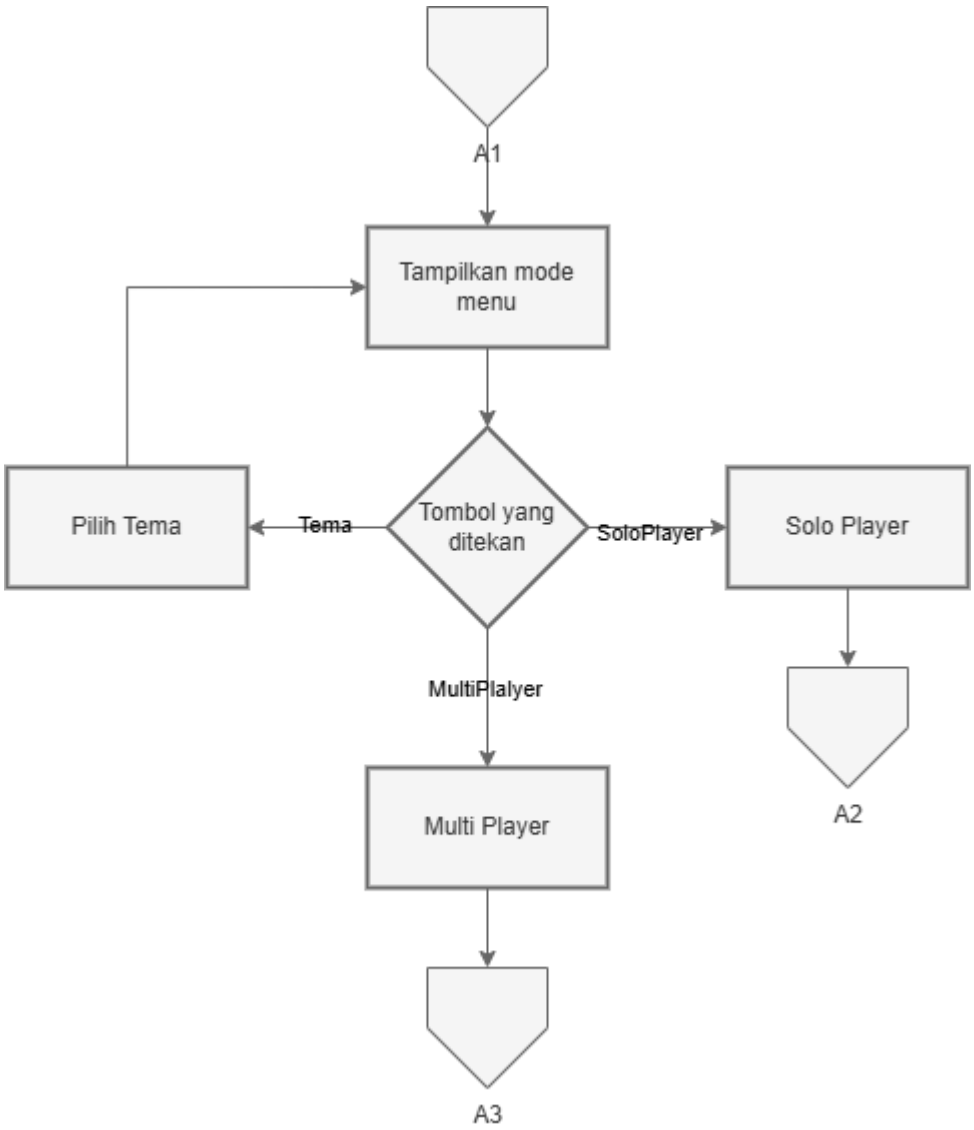
2.1.2 Kebutuhan Non-Fungsional Aplikasi

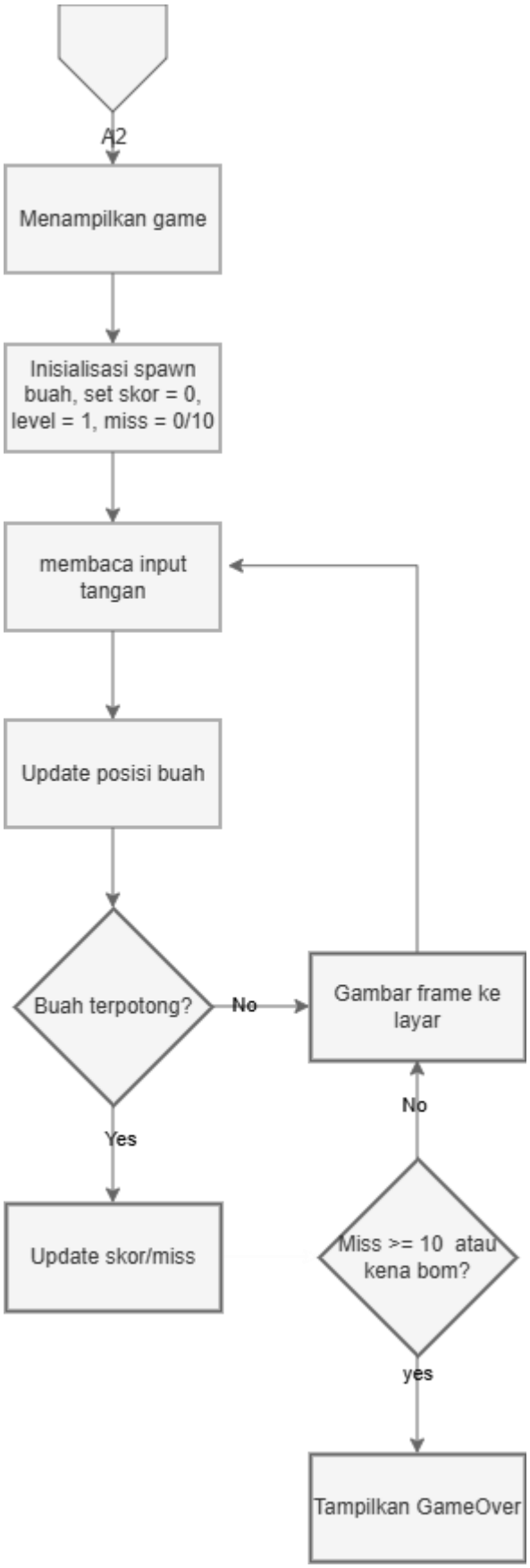
- Aplikasi memiliki tampilan antarmuka yang sederhana dan mudah dipahami pengguna, dengan teks skor dan level yang jelas terbaca.

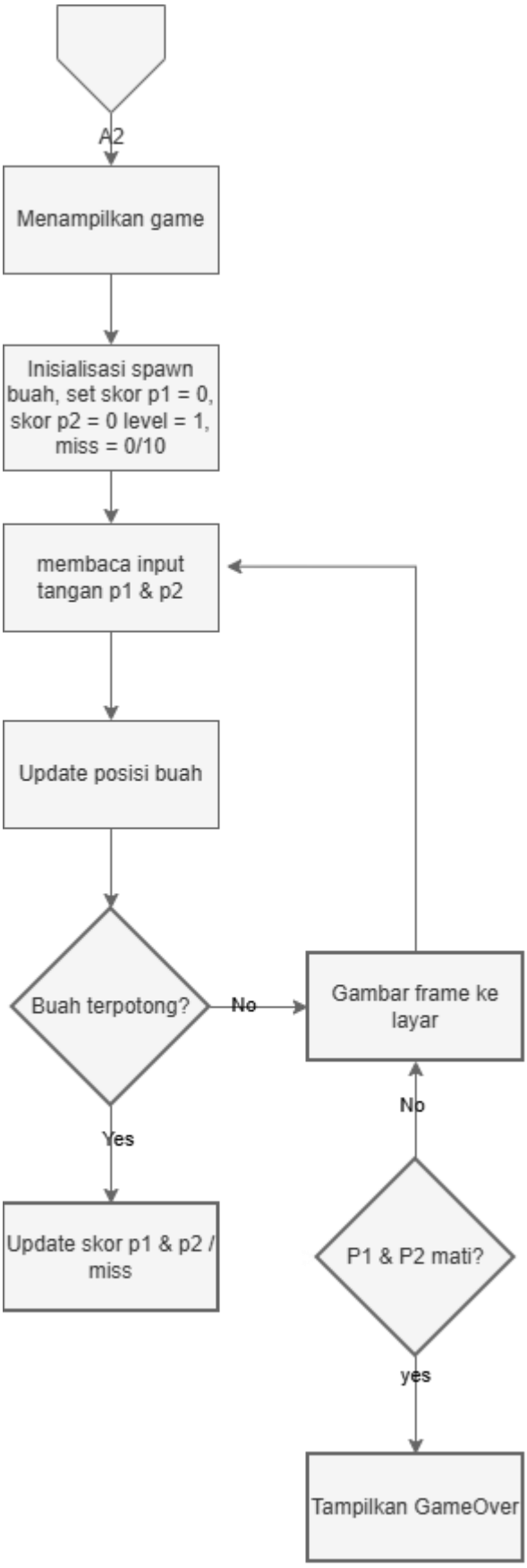
- Aplikasi dapat berjalan dengan lancar pada perangkat dengan spesifikasi standar yang mendukung Python dan Library Pygame.
- Aplikasi mampu memproses deteksi tangan secara real-time agar gerakan pemain terasa responsif.
- Aplikasi dapat dijalankan secara lokal tanpa koneksi internet setelah seluruh dependensi terpasang.

2.2 Diagram Alur




A1
Tampilkan mode menu
Pilih Tema
Tema
Tombol yang ditekan
SoloPlayer
Solo Player
MultiPlalyer
A2
Multi Player
A3


A2
Menampilkan game
Inisialisasi spawn buah, set skor = 0, level = 1, miss = 0/10
membaca input tangan
Update posisi buah
Buah terpotong?
No
Gambar frame ke layar
Yes
Update skor/miss
No
Miss >= 10 atau kena bom?
yes
Tampilkan GameOver


A2
Menampilkan game
Inisialisasi spawn buah, set skor p1 = 0, skor p2 = 0 level = 1, miss = 0/10
membaca input tangan p1 & p2
Update posisi buah
Buah terpotong?
No
Gambar frame ke layar
Yes
Update skor p1 & p2 / miss
No
P1 & P2 mati?
yes
Tampilkan GameOver

2.3 Sketsa Desain Antarmuka

2.3.1 Main Menu

Komponen:

- Tittle: “Fruit Ninja”
- Buttons:
  - Play
  - Exit

2.3.2 Mode Menu

Komponen:

- Tittle: “Mode Menu”

  Buttons:
  - SoloPlayer
  - MultiPlayer
  - Thema

2.3.3 Solo Player

Komponen:

- Tittle:
  - Score: 0
  - Level: 1
  - Miss: 0/10
- Area Permainan:
  - Background:

    default: kayu
  - Objek buah & bomb

2.3.4 MultiPlayer

Komponen:

- Tittle:
  - Player1: 0
  - Player2: 0
  - Level: 1
- Area Permainan:
  - Background:

    Default: Kayu
  - Objek buah & bomb

2.3.5 Theme Menu

Komponen:

- Tittle: “Pilih tema”
- Buttons:
  - Kaktus
  - Kayu
  - Salju
  - Sakura

2.3.6 Game Over

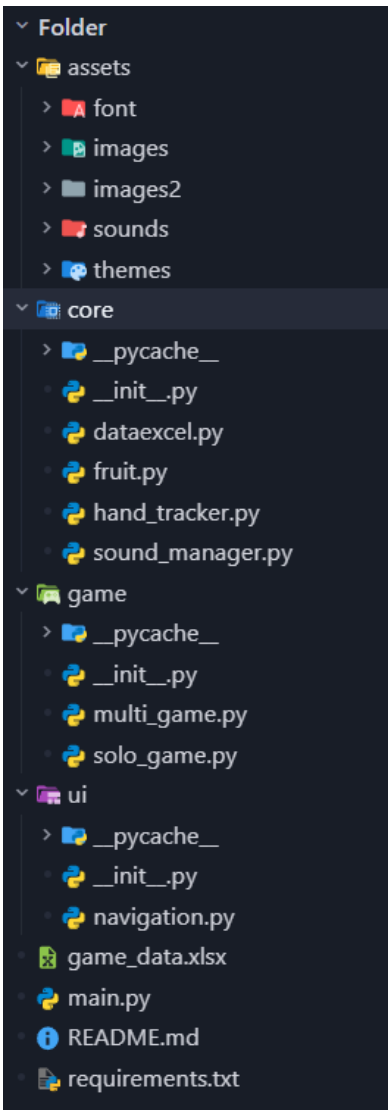
Komponen:

- Tittle:
  - Score
  - Level
- Buttons:
  - Kembali Menu
  - Main Ulang
  - Export

BAB 3
IMPLEMENTASI

3.1 Penjelasan Kode Program

Struktur file FRUIT_NINJA_GAME:



3.1.1 core/dataexcel.py

```python
class GameDataSaver:
    def __init__(self, filename="game_data.xlsx"):
        self.filename = filename

        folder = os.path.dirname(self.filename)
        if folder:
            os.makedirs(folder, exist_ok=True)

    def _get_sheet(self, sheet_name, header):
        if not os.path.exists(self.filename):
            wb = Workbook()
            ws = wb.active
            ws.title = sheet_name
            ws.append(header)
            wb.save(self.filename)

        wb = load_workbook(self.filename)
        if sheet_name not in wb.sheetnames:
            ws = wb.create_sheet(sheet_name)
            ws.append(header)
        else:
            ws = wb[sheet_name]

        return wb, ws

    def save_multiplayer(self, player1, player2, level):
        sheet_name = "Multiplayer"
        header = ["Tanggal", "Player1", "Player2", "Level", "Pemenang", "HighScore"]

        wb, ws = self._get_sheet(sheet_name, header)

        if player1 > player2:
            winner = "Player1"
        elif player2 > player1:
            winner = "Player2"
        else:
            winner = "Draw"

        highscore = max(player1, player2)

        ws.append([
            datetime.now().strftime("%Y-%m-%d %H:%M"),
            player1,
            player2,
            level,
            winner,
            highscore
        ])

        rows = list(ws.iter_rows(values_only=True))[1:]
        rows.sort(key=lambda r: r[5], reverse=True)

        ws.delete_rows(2, ws.max_row)
        for r in rows:
            ws.append(r)

        wb.save(self.filename)

    def save_solo(self, score, missed, level):
        sheet_name = "Soloplayer"
        header = ["Tanggal", "Score", "Missed", "Level"]

        wb, ws = self._get_sheet(sheet_name, header)

        ws.append([
            datetime.now().strftime("%Y-%m-%d %H:%M"),
            score,
            missed,
            level
        ])

        rows = list(ws.iter_rows(values_only=True))[1:]
        rows.sort(key=lambda r: r[1], reverse=True)

        ws.delete_rows(2, ws.max_row)
        for r in rows:
            ws.append(r)

        wb.save(self.filename)
```

Improt Library:

- Os Library ini digunakan untuk berinteraksi dengan sistem operasi, khususnya untuk melakukan pengecekan keberadaan file serta pembuatan direktori secara otomatis jika direktori penyimpanan belum tersedia.
- openpyxl Library *openpyxl* digunakan untuk mengelola file Excel dengan format .xlsx.Workbook berfungsi untuk membuat file Excel baru, sedangkan load_workbook digunakan untuk membuka dan memodifikasi file Excel yang sudah ada.
- Datetime Library ini digunakan untuk mengambil informasi waktu dan tanggal saat data permainan disimpan. Informasi ini penting untuk mencatat waktu pencapaian skor atau hasil permainan.

Fungsi __init__:

- Menentukan nama file Excel (game_data.xlsx) sebagai media penyimpanan.
- Memastikan folder penyimpanan tersedia sebelum file dibuat.

Fungsi _get_sheet:

- Mengecek keberadaan file Excel.
- Membuat file dan sheet baru jika belum tersedia.
- Menambahkan header kolom pada sheet.
- Mengembalikan objek **Workbook** dan **Worksheet** untuk digunakan kembali.

Fungsi save_multiplayer:

Fungsi ini digunakan untuk menyimpan data permainan **mode multiplayer**, dengan data yang dicatat meliputi:

- Tanggal dan waktu permainan
- Skor Player 1 dan Player 2
- Level permainan
- Pemenang permainan
- High score

Data akan diurutkan berdasarkan **HighScore** secara menurun untuk membentuk
papan peringkat (*leaderboard*).

Fungsi save_solo:

Fungsi ini digunakan untuk menyimpan data permainan **mode solo**, yang meliputi:

- Tanggal dan waktu permainan
- Skor pemain
- Jumlah kesalahan (*missed*)
- Level permainan

Data diurutkan berdasarkan **Score** tertinggi agar hasil permainan tersimpan
secara rapi dan terstruktur.

3.1.2 core/sound_manager.py

```python
import os
import pygame

class SoundManager:
    def __init__(self, folder_candidates):
        self.folder = ""
        for f in folder_candidates:
            if os.path.exists(f):
                self.folder = f
                break

        self.sounds = {}

    def load(self, key, filename):
        if not self.folder:
            self.sounds[key] = None
            return
        path = os.path.join(self.folder, filename)
        if os.path.exists(path):
            self.sounds[key] = pygame.mixer.Sound(path)
        else:
            self.sounds[key] = None

    def play(self, key):
        s = self.sounds.get(key)
        print(f"Playing: {key}")
        if s:
            s.play()
        else:
            print(f"Sound {key} not found!")
```

Class **SoundManager** digunakan untuk mengatur manajemen suara dalam permainan, termasuk pencarian folder suara, pemuatan file audio, dan pemutaran efek suara saat permainan berlangsung.

Import Library:

- Os Digunakan untuk mengelola direktori dan file sistem, seperti mengecek
  keberadaan folder dan file suara.
- Pygame Digunakan untuk memuat dan memutar efek suara serta musik latar permainan melalui modul *pygame.mixer*.

Fungsi __init__ :

Fungsi **init** digunakan untuk menginisialisasi sistem suara dan menentukan
folder yang berisi file audio.

- folder_candidates
  Merupakan daftar kemungkinan folder yang berisi file suara.
- os.path.exists(f)
  Digunakan untuk mengecek apakah folder suara tersedia.
- self.folder
  Digunakan untuk menyimpan path folder suara yang valid.
- self.sounds = {} Digunakan sebagai media penyimpanan objek suara dalam bentuk dictionary.

Fungsi load:

Fungsi **load** digunakan untuk memuat file suara berdasarkan nama file yang
diberikan dan menyimpannya ke dalam sistem suara.

- os.path.join(self.folder, filename) Digunakan untuk menggabungkan path folder dengan nama file suara.
- pygame.mixer.Sound(path) Digunakan untuk memuat file suara ke dalam memori pygame.
- self.sounds[key] Digunakan untuk menyimpan suara dengan key tertentu agar mudah dipanggil.
- Exception Handling Digunakan untuk menangani kondisi ketika folder atau file suara tidak ditemukan.

Fungsi play

Fungsi **play** digunakan untuk memutar efek suara yang telah dimuat sebelumnya.

- self.sounds.get(key) Digunakan untuk mengambil objek suara berdasarkan key.
- s.play() Digunakan untuk memutar efek suara.
- print("Sound not found") Digunakan sebagai notifikasi apabila suara tidak tersedia.

3.1.3 core/hand_tracker.py

```
import cv2
import mediapipe as mp

class HandTracker:
    def __init__(self, max_hands=1, det_conf=0.4, track_conf=0.4):
        self.mp_hands = mp.solutions.hands
        self.hands = self.mp_hands.Hands(
            max_num_hands=max_hands,
            min_detection_confidence=det_conf,
            min_tracking_confidence=track_conf
        )

    def process(self, frame_bgr):
        rgb = cv2.cvtColor(frame_bgr, cv2.COLOR_BGR2RGB)
        return self.hands.process(rgb)

    def close(self):
        self.hands.close()

```

Class **HandTracker** digunakan untuk mendeteksi dan melacak tangan pemain secara real-time dengan memanfaatkan library MediaPipe dan OpenCV.

Import Library:

- cv2 (OpenCV) Digunakan untuk mengelola pengolahan citra dan video dari kamera, termasuk konversi format warna dan pemrosesan frame.

- mediapipe as mp Digunakan sebagai library utama untuk mendeteksi dan melacak tangan secara real-time menggunakan model *hand tracking* dari MediaPipe.

Fungsi __init__ :

Fungsi **init** digunakan untuk menginisialisasi sistem pendeteksian tangan
beserta parameter pendukungnya.

- max_hands Digunakan untuk menentukan jumlah maksimum tangan yang dapat dideteksi.
- det_conf Digunakan untuk menentukan tingkat kepercayaan minimum pada proses deteksi tangan.
- track_conf Digunakan untuk menentukan tingkat kepercayaan minimum pada proses pelacakan tangan.
- mp.solutions.hands Digunakan untuk mengakses modul pendeteksian tangan dari MediaPipe.
- self.mp_hands.Hands() Digunakan untuk menginisialisasi objek pendeteksi tangan dengan parameter yang telah ditentukan.

Fungsi process:

Fungsi **process** digunakan untuk memproses frame kamera dan mendeteksi tangan pada setiap frame.

- cv2.cvtColor(frame_bgr, cv2.COLOR_BGR2RGB) Digunakan untuk mengubah format warna frame dari BGR ke RGB agar sesuai dengan kebutuhan MediaPipe.
- self.hands.process(rgb) Digunakan untuk melakukan proses deteksi dan pelacakan tangan.

Fungsi close:

Fungsi **close** digunakan untuk menghentikan dan membersihkan resource yang
digunakan oleh sistem pendeteksian tangan.

- self.hands.close() Digunakan untuk menutup objek MediaPipe Hands secara aman.

3.1.4 core/Fruit.py

```python
import random
import cv2
import os

class Fruit:
    def __init__(self, img, is_bomb, x, y, vy):
        self.img = img
        self.is_bomb = is_bomb
        self.x = x
        self.y = y
        self.vy = vy
        self.alive = True
        self.cut = False
        self.counted = False

class FruitManager:
    def __init__(self, width, height, image_folder="assets/images", min_size=70, max_size=120):
        self.WIDTH = width
        self.HEIGHT = height
        self.min_size = min_size
        self.max_size = max_size

        self.fruit_images = []
        for f in os.listdir(image_folder):
            if f.endswith(".png"):
                self.fruit_images.append({
                    "img": cv2.imread(os.path.join(image_folder, f), cv2.IMREAD_UNCHANGED),
                    "is_bomb": "bomb" in f.lower()
                })

        if not self.fruit_images:
            raise Exception("Tidak ada file PNG di folder 'images'!")

        self.cache = {}

    def _resize_cached(self, base, size, is_bomb):
        key = (size, is_bomb)
        if key not in self.cache:
            try:
                self.cache[key] = cv2.resize(base, (size, size))
            except Exception:
                self.cache[key] = base.copy()
        return self.cache[key]

    def spawn_fruit_solo(self, level):
        f = random.choice(self.fruit_images)
        base = f["img"]
        size = random.randint(self.min_size, self.max_size)
        img = self._resize_cached(base, size, f["is_bomb"]).copy()

        x = random.randint(100, self.WIDTH - 150)
        y = -random.randint(100, 800)
        vy = random.uniform(6 + level, 9 + level)

        return Fruit(img, f["is_bomb"], x, y, vy)

    def spawn_fruit_multi(self, level, p1_alive=True, p2_alive=True):
        import random
        f = random.choice(self.fruit_images)
        base = f["img"]
        size = random.randint(self.min_size, self.max_size)
        img = self._resize_cached(base, size, f["is_bomb"]).copy()

        left_min = 60
        left_max = max(120, self.WIDTH // 2 - 120)
        right_min = min(self.WIDTH // 2 + 40, self.WIDTH - 300)
        right_max = self.WIDTH - 120

        if p1_alive and p2_alive:
            x = random.randint(left_min, left_max) if random.random() < 0.5 else random.randint(right_min, right_max)
        elif p1_alive and not p2_alive:
            x = random.randint(left_min, left_max)
        elif not p1_alive and p2_alive:
            x = random.randint(right_min, right_max)
        else:
            x = random.randint(left_min, right_max)

        y = -random.randint(80, 600)
        vy = random.uniform(5 + level * 0.6, 9 + level * 1.0)

        return Fruit(img, f["is_bomb"], x, y, vy)

```

Import Library:

- random digunakan untuk menghasilkan nilai acak pada pemilihan buah, ukuran, posisi, dan kecepatan buah.
- cv2 Digunakan untuk membaca dan memproses gambar buah.
- Os Digunakan untuk mengakses folder aset gambar buah.

Class Fruit:

Class **Fruit** digunakan untuk merepresentasikan objek buah atau bom dalam
permainan.

- Img Menyimpan citra buah yang akan ditampilkan.
- is_bomb Menentukan apakah objek merupakan bom atau buah.
- x, y Menentukan posisi buah pada layar permainan.
- Vy Menentukan kecepatan gerak vertikal buah.
- Alive Menunjukkan status buah masih aktif.
- Cut Menandakan apakah buah sudah terpotong.
- Counted Digunakan untuk mencegah perhitungan skor ganda.

Class FruitManager:

Class **FruitManager** digunakan untuk mengelola pembuatan, pengaturan, dan
kemunculan buah selama permainan.

Fungsi __init__ :

Fungsi **init** digunakan untuk menginisialisasi pengelola buah.

- width, height Menentukan ukuran layar permainan.
- image_folder Menentukan lokasi folder gambar buah.
- cv2.imread() Digunakan untuk membaca file gambar buah.
- "bomb" in f.lower() Digunakan untuk membedakan buah dan bom berdasarkan nama file.
- self.cache = {} Digunakan untuk menyimpan hasil resize gambar agar lebih efisien.

Fungsi _resize_cached:

Fungsi **_resize_cached** digunakan untuk mengubah ukuran gambar buah.

- cv2.resize() Digunakan untuk mengubah ukuran gambar buah.

- Cache Digunakan untuk menyimpan hasil resize agar tidak diproses ulang.

Fungsi spawn_fruit_solo:

Fungsi **spawn_fruit_solo** digunakan untuk menghasilkan buah pada mode solo.

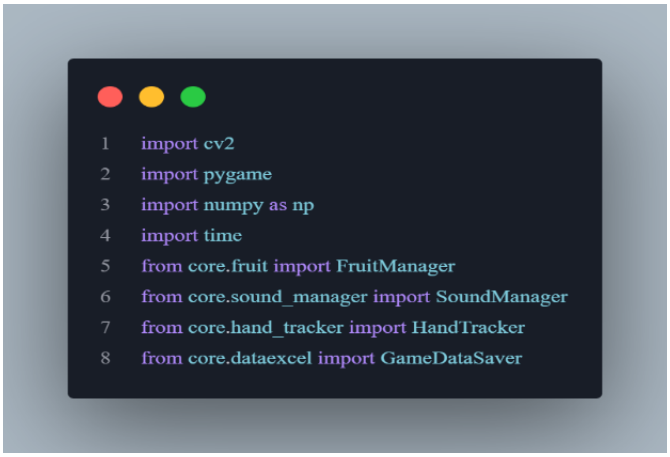
- random.choice(self.fruit_images) Memilih buah secara acak.
- random.randint() Menentukan ukuran dan posisi buah.
- random.uniform() Menentukan kecepatan jatuh buah berdasarkan level.
- return Fruit(...) Mengembalikan objek buah yang siap digunakan.

Fungsi spawn_fruit_multi:

Fungsi **spawn_fruit_multi** digunakan untuk menghasilkan buah pada mode
multiplayer.

- Pembagian area kiri dan kanan Digunakan untuk menyesuaikan area pemain.
- p1_alive, p2_alive Menentukan area kemunculan buah berdasarkan status pemain.
- Kecepatan buah berbasis level Digunakan untuk meningkatkan tingkat kesulitan permainan.
- return Fruit(...) Mengembalikan objek buah untuk mode multiplayer.

3.1.5 /game/solo_game.py



Import library:

- cv2 (OpenCV): Mengakses kamera, membaca frame video, dan menggambar pointer tangan pada gambar.
- pygame: Membuat jendela game, menampilkan gambar ke layar, menangani event, dan memutar efek suara.
- numpy (np): Mengolah data gambar dalam bentuk array, misalnya mengubah ukuran dan menggabungkan gambar.
- time: Mengatur dan membaca waktu, seperti mencatat durasi permainan dan jeda tampilan
- FruitManager: Mengelola buah (spawn, gerak, jenis buah/bom) selama permainan.
- SoundManager: Memuat dan memutar efek suara seperti tebasan, bom, dan naik level.
- HandTracker: Mendeteksi dan melacak posisi tangan/jari pemain sebagai “pedang” pemotong buah
- GameDataSaver: Menyimpan data hasil permainan (skor, miss, level) ke file Excel untuk dokumentasi

Class SoloFruitNinjaGame:

```
class SoloFruitNinjaGame:
    def __init__(self, background_path="assets/themes/2.png"):
        pygame.init()
        pygame.mixer.init()
        self.WIDTH, self.HEIGHT = 1280, 720
        self.screen = pygame.display.set_mode((self.WIDTH, self.HEIGHT), pygame.RESIZABLE)
        pygame.display.set_caption("Ninja Fruits - Solo")

        self.cap = cv2.VideoCapture(0)
        self.cap.set(3, self.WIDTH)
        self.cap.set(4, self.HEIGHT)

        bg_img = cv2.imread(background_path)
        self.background_path = background_path
        self.background = cv2.resize(bg_img, (self.WIDTH, self.HEIGHT)) if bg_img is not None else \
            255 * np.ones((self.HEIGHT, self.WIDTH, 3), dtype=np.uint8)

        self.sounds = SoundManager(["assets/sounds", "ninja_fruit_sounds"])
        self.sounds.load("slash", "slash.mp3")
        self.sounds.load("boom", "boom.mp3")
        self.sounds.load("levelup", "levelup.mp3")

        self.FONT = "assets/font/Gang_of_Three_Regular.ttf"

        self.hand_tracker = HandTracker(max_hands=1)
        self.fruit_manager = FruitManager(self.WIDTH, self.HEIGHT)
        self.clock = pygame.time.Clock()

        self.start_time = time.time()
        self.sliced_fruits = 0
        self.touched_bombs = 0

        self.card_img = pygame.image.load("assets/images2/card.png")
        self.btn1_img = pygame.image.load("assets/images2/1.png")
        self.btn2_img = pygame.image.load("assets/images2/2.png")
        self.btn3_img = pygame.image.load("assets/images2/6.png")

        self.card_img = pygame.transform.scale(self.card_img, (350, 350))
        self.btn1_img = pygame.transform.scale(self.btn1_img, (90, 90))
        self.btn2_img = pygame.transform.scale(self.btn2_img, (90, 90))
        self.btn3_img = pygame.transform.scale(self.btn3_img, (90, 90))
        self.data_saver = GameDataSaver()
        self.save_message = None
        self.save_message_time = 0

        self.reset_game()
```

Class **SoloFruitNinjaGame** digunakan untuk menjalankan permainan Fruit Ninja mode solo, di mana permainan hanya melibatkan satu pemain dengan satu jari.

Fungsi __init__:

Fungsi **init** digunakan untuk menginisialisasi seluruh komponen utama
permainan sebelum permainan dimulai.

- pygame.init() dan pygame.mixer.init() Digunakan untuk menginisialisasi modul pygame dan sistem audio.
- self.WIDTH, self.HEIGHT = 1280, 720 Menentukan resolusi awal layar permainan.
- pygame.display.set_mode(..., pygame.RESIZABLE) Digunakan untuk membuat jendela permainan dengan ukuran yang dapat diubah.
- pygame.display.set_caption("Ninja Fruits - Solo") Digunakan untuk memberikan judul pada jendela permainan.
- cv2.VideoCapture(0) Digunakan untuk mengaktifkan kamera sebagai input utama permainan.
- self.cap.set(3, self.WIDTH) dan self.cap.set(4, self.HEIGHT) Digunakan untuk menyesuaikan resolusi kamera dengan ukuran layar.
- cv2.imread(background_path) Digunakan untuk membaca gambar latar permainan.
- cv2.resize(...) Digunakan untuk menyesuaikan ukuran background dengan resolusi layar.
- np.ones(...) Digunakan sebagai background cadangan apabila gambar latar tidak tersedia.
- SoundManager(...) Digunakan untuk mengelola efek suara permainan.
- self.sounds.load(...) Digunakan untuk memuat suara potong buah, ledakan bom, dan kenaikan level.
- self.FONT Digunakan untuk menentukan jenis font teks dalam permainan.
- HandTracker(max_hands=1) Digunakan untuk mendeteksi satu tangan pemain pada mode solo.
- FruitManager(self.WIDTH, self.HEIGHT) Digunakan untuk mengelola kemunculan dan pergerakan buah.
- pygame.time.Clock() Digunakan untuk mengatur kecepatan frame (*frame rate*) permainan.
- time.time() Digunakan untuk mencatat waktu mulai permainan.
- self.sliced_fruits Digunakan untuk menghitung jumlah buah yang berhasil dipotong.
- self.touched_bombs Digunakan untuk menghitung jumlah bom yang tersentuh pemain.

- pygame.image.load(...) Digunakan untuk memuat elemen antarmuka permainan seperti kartu hasil dan tombol.
- pygame.transform.scale(...)Digunakan untuk menyesuaikan ukuran elemen antarmuka.
- GameDataSaver() Digunakan untuk menyimpan data hasil permainan mode solo ke dalam file Excel.
- self.reset_game()Digunakan untuk mengatur kondisi awal permainan.

Fungsi reset_game:

```
def reset_game(self):
    self.score = 0
    self.missed = 0
    self.level = 1
    self.fruits = [self.fruit_manager.spawn_fruit_solo(1) for _ in range(5)]
    self.x_buffer = []
    self.y_buffer = []
    self.prev_x = None
    self.prev_y = None
    self.pointer_history = []
    self.game_over = False
    self.show_go_screen = False
    self.go_start_ticks = 0
    self.GO_DURATION = 2000
    self.game_over_time = None
    self._setup_gameover_ui()
```

Fungsi **reset_game** digunakan untuk mengatur ulang kondisi permainan ke keadaan awal pada mode solo, baik saat permainan baru dimulai maupun ketika pemain memulai ulang permainan setelah *game over*.

- self.score = 0 Digunakan untuk mengatur ulang skor pemain menjadi nol.
- self.missed = 0 Digunakan untuk mengatur ulang jumlah buah yang terlewat.
- self.level = 1 Digunakan untuk mengatur level permainan ke level awal.
- self.fruits=[self.fruit_manager.spawn_fruit_solo(1) for _ in range(5)] Digunakan untuk membuat sejumlah buah awal pada mode solo.

- self.x_buffer dan self.y_buffer Digunakan sebagai buffer untuk menyimpan riwayat posisi tangan pemain.
- self.prev_x dan self.prev_y Digunakan untuk menyimpan posisi tangan sebelumnya.
- self.pointer_history Digunakan untuk menyimpan jejak pergerakan tangan pemain.
- self.game_over = False Digunakan untuk menandai bahwa permainan masih berjalan.
- self.show_go_screen = False Digunakan untuk mengatur tampilan layar *game over*.
- self.go_start_ticks = 0 Digunakan untuk mencatat waktu awal tampilan *game over*.
- self.GO_DURATION = 2000 Menentukan durasi tampilan teks *game over* dalam satuan milidetik.
- self.game_over_time = None Digunakan untuk menyimpan waktu saat permainan berakhir.
- self._setup_gameover_ui() Digunakan untuk menyiapkan tampilan antarmuka *game over*.

Fungsi _overlay_image:

```
def _overlay_image(self, bg, fg, x, y):
    h, w = fg.shape[:2]
    if len(fg.shape) == 3 and fg.shape[2] == 3:
        fg = np.dstack((fg, np.ones((h, w), dtype=np.uint8) * 255))
    if x < 0 or y < 0 or x + w > self.WIDTH or y + h > self.HEIGHT:
        return bg
    alpha = fg[:, :, 3] / 255.0
    for c in range(3):
        bg[y:y+h, x:x+w, c] = (alpha * fg[:, :, c] + (1 - alpha) * bg[y:y+h, x:x+w, c]).astype(np.uint8)
    return bg
```

Fungsi **_overlay_image** digunakan untuk menampilkan gambar objek (*foreground*) di atas gambar latar (*background*) dengan memanfaatkan nilai transparansi (*alpha blending*).

- h, w = fg.shape[:2] Digunakan untuk mengambil tinggi dan lebar gambar *foreground*.
- fg.shape[2] == 3 Digunakan untuk mengecek apakah gambar *foreground* belum memiliki channel alpha.
- np.dstack(...) Digunakan untuk menambahkan channel alpha pada gambar agar mendukung transparansi.
- if x < 0 or y < 0 or x + w > self.WIDTH or y + h > self.HEIGHT Digunakan untuk memastikan gambar tidak digambar di luar batas layar.

- return bg
  Mengembalikan background tanpa perubahan jika posisi gambar tidak valid.
- alpha = fg[:, :, 3] / 255.0 Digunakan untuk mengambil nilai transparansi dari channel alpha.
- for c in range(3) Digunakan untuk menggabungkan warna RGB foreground dan background menggunakan perhitungan *alpha blending*.
- astype(np.uint8)
  Digunakan untuk memastikan tipe data citra sesuai dengan format OpenCV.
- return bg
  Mengembalikan gambar latar yang telah ditambahkan objek *foreground*.

Fungsi _update_size:

```
def _update_size(self):
    new_w, new_h = self.screen.get_size()
    if new_w != self.WIDTH or new_h != self.HEIGHT:
        self.WIDTH, self.HEIGHT = new_w, new_h
        bg_img = cv2.imread(self.background_path)
        self.background = cv2.resize(bg_img, (self.WIDTH, self.HEIGHT)) if bg_img is not None else \
            255 * np.ones((self.HEIGHT, self.WIDTH, 3), dtype=np.uint8)
        self.fruit_manager.WIDTH = self.WIDTH
        self.fruit_manager.HEIGHT = self.HEIGHT
```

Fungsi **_update_size** digunakan untuk menyesuaikan ukuran layar permainan
ketika terjadi perubahan ukuran jendela (*window resize*).

- self.screen.get_size() Digunakan untuk mengambil ukuran terbaru jendela permainan.
- if new_w != self.WIDTH or new_h != self.HEIGHT Digunakan untuk mengecek apakah ukuran layar mengalami perubahan.
- self.WIDTH, self.HEIGHT = new_w, new_h Digunakan untuk memperbarui ukuran layar permainan.
- cv2.imread(self.background_path) Digunakan untuk membaca ulang gambar latar permainan.
- cv2.resize(...) Digunakan untuk menyesuaikan ukuran background dengan ukuran layar terbaru.
- np.ones(...) Digunakan sebagai background cadangan jika gambar latar tidak ditemukan.

- self.fruit_manager.WIDTH dan self.fruit_manager.HEIGHT Digunakan untuk memperbarui batas area kemunculan buah agar tetap sesuai dengan ukuran layar.

Fungsi _handle_hand:

```
def _handle_hand(self, cam_frame, frame, now):
    result = self.hand_tracker.process(cam_frame)
    if not result.multi_hand_landmarks:
        return frame

    cam_h, cam_w = cam_frame.shape[:2]
    scale_x = self.WIDTH / cam_w
    scale_y = self.HEIGHT / cam_h

    for hand_lms in result.multi_hand_landmarks:
        x = int(hand_lms.landmark[8].x * cam_w * scale_x)
        y = int(hand_lms.landmark[8].y * cam_h * scale_y)

        # Smoothing buffer
        self.x_buffer.append(x)
        self.y_buffer.append(y)
        if len(self.x_buffer) > 4:
            self.x_buffer.pop(0)
            self.y_buffer.pop(0)
        x = int(sum(self.x_buffer) / len(self.x_buffer))
        y = int(sum(self.y_buffer) / len(self.y_buffer))

        cv2.circle(frame, (x, y), 20, (0, 140, 255), -1)
        cv2.circle(frame, (x, y), 10, (0, 255, 255), -1)

        # Anti-jitter
        if self.prev_x is not None:
            x = int(self.prev_x * 0.5 + x * 0.5)
            y = int(self.prev_y * 0.5 + y * 0.5)

        self.pointer_history.append((x, y, now))
        self.pointer_history = [p for p in self.pointer_history if now - p[2] < 0.25]

        if self.prev_x is not None and self.prev_y is not None:
            for fruit in self.fruits:
                if fruit.alive and not fruit.cut:
                    fh, fw = fruit.img.shape[:2]
                    if fruit.x < x < fruit.x + fw and fruit.y < y < fruit.y + fh:
                        fruit.cut = True
                        if fruit.is_bomb and not self.game_over:
                            self.game_over = True
                            self.show_go_screen = True
                            self.go_start_ticks = pygame.time.get_ticks()
                            self.game_over_time = now
                            self.touched_bombs += 1
                            self.sounds.play("boom")
                        else:
                            self.score += 1
                            self.sliced_fruits += 1
                            self.sounds.play("slash")
                            if self.score % 30 == 0:
                                self.level += 1
                                self.sounds.play("levelup")

        self.prev_x, self.prev_y = x, y
    return frame
```

Fungsi **_handle_hand** digunakan untuk memproses pendeteksian tangan pemain, menampilkan penunjuk (*pointer*), serta menangani interaksi pemotonganbuah pada mode permainan solo.

- self.hand_tracker.process(cam_frame) Digunakan untuk mendeteksi tangan pemain dari frame kamera.
- if not result.multi_hand_landmarks Digunakan untuk memastikan bahwa tangan terdeteksi sebelum proses dilanjutkan.
- cam_frame.shape[:2] Digunakan untuk mengambil resolusi frame kamera.
- scale_x dan scale_y Digunakan untuk menyesuaikan koordinat kamera dengan ukuran layar permainan.
- hand_lms.landmark[8] Digunakan untuk mengambil koordinat ujung jari telunjuk sebagai titik penunjuk pemain.
- Smoothing Buffer **(**x_buffer**,** y_buffer**)** Digunakan untuk meratakan pergerakan tangan agar kursor tidak bergetar (*anti-jitter*).
- cv2.circle(...) Digunakan untuk menggambar penunjuk tangan pada layar permainan.
- Anti-jitter tambahan Digunakan untuk menghaluskan pergerakan dengan menggabungkan posisi saat ini dan posisi sebelumnya.
- self.pointer_history Digunakan untuk menyimpan riwayat pergerakan tangan dalam interval waktu tertentu.
- Deteksi Pemotongan Buah Digunakan untuk mengecek apakah posisi tangan mengenai area buah.
    - Jika buah adalah bom, permainan berakhir (*game over*).
    - Jika buah normal, skor pemain bertambah dan suara potong dimainkan.
- self.score % 30 == 0 Digunakan untuk menaikkan level permainan setiap kelipatan skor tertentu.
- self.prev_x, self.prev_y Digunakan untuk menyimpan posisi tangan terakhir.
- return frameMengembalikan frame yang telah diproses untuk ditampilkan.

Fungsi _update_fruits:

```
def _update_fruits(self):
    for fruit in self.fruits:
        if fruit.alive:
            fruit.y += fruit.vy
            if fruit.y > self.HEIGHT and not fruit.counted:
                fruit.alive = False
                fruit.counted = True
                if not fruit.is_bomb:
                    self.missed += 1
                    if self.missed >= 10 and not self.game_over:
                        self.game_over = True
                        self.show_go_screen = True
                        self.go_start_ticks = pygame.time.get_ticks()
                        self.game_over_time = time.time()

    for i in range(len(self.fruits)):
        if not self.fruits[i].alive or self.fruits[i].cut:
            self.fruits[i] = self.fruit_manager.spawn_fruit_solo(self.level)
```

Fungsi **_update_fruits** digunakan untuk memperbarui posisi buah, mendeteksi buah yang terlewat, serta mengatur kondisi *game over* pada mode permainan solo.

- for fruit in self.fruits Digunakan untuk memproses seluruh objek buah yang sedang aktif.
- fruit.y += fruit.vy Digunakan untuk memperbarui posisi vertikal buah sesuai kecepatan geraknya.
- if fruit.y > self.HEIGHT Digunakan untuk mendeteksi buah yang telah melewati batas bawah layar.
- fruit.counted Digunakan untuk mencegah perhitungan buah terlewat lebih dari satu kali.
- self.missed += 1 Digunakan untuk menambah jumlah buah yang terlewat oleh pemain.
- if self.missed >= 10 Digunakan untuk menentukan kondisi *game over* apabila jumlah buah terlewat mencapai batas tertentu.
- self.game_over = True Digunakan untuk menghentikan permainan ketika kondisi *game over* terpenuhi.
- self.show_go_screen = True Digunakan untuk menampilkan layar *game over*.
- pygame.time.get_ticks() dan time.time() Digunakan untuk mencatat waktu saat permainan berakhir.
- if not self.fruits[i].alive or self.fruits[i].cut Digunakan untuk mengecek buah yang sudah tidak aktif atau telah terpotong.
- spawn_fruit_solo(self.level) Digunakan untuk mengganti buah yang sudah hilang dengan buah baru sesuai level permainan.

Fungsi _setup_gameover_ui:

```
def _setup_gameover_ui(self):
    cx = self.WIDTH // 2
    cy = self.HEIGHT // 2

    self.card_rect = self.card_img.get_rect(center=(cx, cy - 80))

    y = cy + 165
    self.btn1_rect = self.btn1_img.get_rect(center=(cx - 150, y))
    self.btn2_rect = self.btn2_img.get_rect(center=(cx, y))
    self.btn3_rect = self.btn3_img.get_rect(center=(cx + 150, y))
```

Fungsi **_setup_gameover_ui** digunakan untuk mengatur tata letak (*layout*)
antarmuka Game Over, termasuk kartu hasil permainan dan tombol menu.

- cx = self.WIDTH // 2 Digunakan untuk menentukan titik tengah horizontal layar.
- cy = self.HEIGHT // 2 Digunakan untuk menentukan titik tengah vertikal layar.
- self.card_img.get_rect(center=(cx, cy - 80)) Digunakan untuk memposisikan kartu hasil permainan di bagian tengah layer dengan sedikit pergeseran ke atas.
- y = cy + 165 Digunakan sebagai posisi vertikal untuk menempatkan tombol-tombol menu.
- self.btn1_img.get_rect(center=(cx - 150, y)) Digunakan untuk menempatkan tombol pertama di sisi kiri.
- self.btn2_img.get_rect(center=(cx,y))Digunakan untuk menempatkan tombol kedua di bagian tengah.
- self.btn3_img.get_rect(center=(cx + 150, y)) Digunakan untuk menempatkan tombol ketiga di sisi kanan.

```python
def run(self):
    running = True
    while running:
        success, cam = self.cap.read()
        if not success:
            time.sleep(0.05)
            continue

        cam = cv2.flip(cam, 1)
        self._update_size()

        frame = self.background.copy()
        now = time.time()
        self._update_fruits()

        for fruit in self.fruits:
            if fruit.alive:
                frame = self._overlay_image(
                    frame, fruit.img, int(fruit.x), int(fruit.y)
                )

        if not self.game_over:
            frame = self._handle_hand(cam, frame, now)

            cv2.putText(frame, f"Score: {self.score}", (30, 60),
                        cv2.FONT_HERSHEY_SIMPLEX, 1.5, (255,255,255), 3)
            cv2.putText(frame, f"Miss: {self.missed}/10", (30, 120),
                        cv2.FONT_HERSHEY_SIMPLEX, 1.5, (0,0,255), 3)
            cv2.putText(frame, f"Level: {self.level}", (30, 180),
                        cv2.FONT_HERSHEY_SIMPLEX, 1.5, (0,255,255), 3)

        frame_rgb = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
        surface = pygame.surfarray.make_surface(frame_rgb.swapaxes(0, 1))
        self.screen.blit(surface, (0, 0))

        if self.game_over:

            overlay = pygame.Surface((self.WIDTH, self.HEIGHT), pygame.SRCALPHA)
            overlay.fill((0, 0, 0, 160))
            self.screen.blit(overlay, (0, 0))

            elapsed = pygame.time.get_ticks() - self.go_start_ticks

            if self.show_go_screen and elapsed < self.GO_DURATION:
                font_go = pygame.font.Font(self.FONT, 96)
                go_text = font_go.render("GAME OVER", True, (255, 60, 60))
                go_rect = go_text.get_rect(center=(self.WIDTH // 2, self.HEIGHT // 2))
                self.screen.blit(go_text, go_rect)
            else:
                self.show_go_screen = False

                self.screen.blit(self.card_img, self.card_rect)
                self.screen.blit(self.btn1_img, self.btn1_rect)
                self.screen.blit(self.btn2_img, self.btn2_rect)
                self.screen.blit(self.btn3_img, self.btn3_rect)

                font_score = pygame.font.Font(self.FONT, 36)
                score_text = font_score.render(f"SCORE : {self.score}", True, (0, 0, 0))
                level_text = font_score.render(f"LEVEL : {self.level}", True, (0, 0, 0))
                score_rect = score_text.get_rect(
                    center=(self.card_rect.centerx, self.card_rect.centery - 30)
                )
                level_rect = level_text.get_rect(
                    center=(self.card_rect.centerx, self.card_rect.centery + 30)
                )
                self.screen.blit(score_text, score_rect)
                self.screen.blit(level_text, level_rect)

        if self.save_message:
            if time.time() - self.save_message_time < 2.5:
                font = pygame.font.Font(self.FONT, 28)
                text = font.render(self.save_message, True, (0, 255, 0))
                text_rect = text.get_rect(center=(self.WIDTH // 2, self.HEIGHT // 2 + 265))
                self.screen.blit(text, text_rect)
            else:
                self.save_message = None

        pygame.display.flip()
        self.clock.tick(20)

        for event in pygame.event.get():
            if event.type == pygame.QUIT:
                self._cleanup()
                return "exit"

            if event.type == pygame.KEYDOWN:
                if event.key == pygame.K_ESCAPE:
                    self._cleanup()
                    return "menu"
                if self.game_over and event.key == pygame.K_r:
                    self.reset_game()

            if event.type == pygame.MOUSEBUTTONDOWN and self.game_over:
                mx, my = event.pos
                if self.btn1_rect.collidepoint(mx, my):
                    self._cleanup()
                    return "menu"
                elif self.btn2_rect.collidepoint(mx, my):
                    self.reset_game()
                elif self.btn3_rect.collidepoint(mx, my):
                    self.data_saver.save_solo(
                        score=self.score,
                        missed=self.missed,
                        level=self.level
                    )
                    self.save_message = "Data solo player berhasil disimpan"
                    self.save_message_time = time.time()

    self._cleanup()
    return "menu"
```

Fungsi run:

fungsi **run** digunakan untuk menjalankan loop utama permainan solo, memproses input pemain, logika permainan, dan menampilkan antarmuka.

- success, cam = self.cap.read() Membaca frame kamera untuk mendeteksi gerakan tangan. Jika gagal, loop menunggu 0,05 detik sebelum mencoba lagi.
- cam = cv2.flip(cam, 1) Membalik frame kamera secara horizontal agar gerakan pemain sesuai arah layar.
- self._update_size() Memperbarui ukuran layar dan background jika jendela di-resize.
- frame = self.background.copy() Membuat frame awal dari background untuk rendering.
- now = time.time() Mencatat waktu saat ini untuk keperluan logika permainan.
- self._update_fruits() Memperbarui posisi buah, mengganti buah yang mati atau terpotong, dan menyesuaikan level.
- for fruit in self.fruits: Menampilkan semua buah hidup di layar menggunakan _overlay_image.
- if not self.game_over:
    - Memproses input tangan pemain dengan self._handle_hand(cam, frame, now).
    - Menampilkan skor, jumlah buah terlewat, dan level permainan.
- frame_rgb = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB) Mengonversi frame ke format RGB untuk Pygame.
- surface=pygame.surfarray.make_surface(frame_rgb.swapaxes(0, 1)) Membuat surface Pygame dari frame kamera.
- Game Over Handling**:**
    - Membuat overlay gelap dengan transparansi.
    - Menampilkan teks GAME OVER selama durasi GO_DURATION.
    - Setelah itu, menampilkan kartu hasil permainan, skor, level, dan tombol interaksi (btn1, btn2, btn3).
- Pesan penyimpanan data **(**save_message**):** Ditampilkan selama 2,5 detik jika pemain menekan tombol simpan data.
- pygame.display.flip() dan self.clock.tick(20) Memperbarui layar dan mengatur frame rate sekitar 20 FPS.
- Event Handling**:**
    - pygame.QUIT → membersihkan sumber daya dan keluar.
    - KEYDOWN K_ESCAPE → kembali ke menu utama.
    - KEYDOWN K_r saat game over → reset permainan.
    - MOUSEBUTTONDOWN saat game over:
        - Klik btn1 → kembali ke menu utama.
        - Klik btn2 → reset permainan.

    - Klik btn3 → menyimpan data solo player dan menampilkan pesan konfirmasi.
- self._cleanup() dan return "menu" Menutup sumber daya dan kembali ke menu utama saat loop permainan selesai.

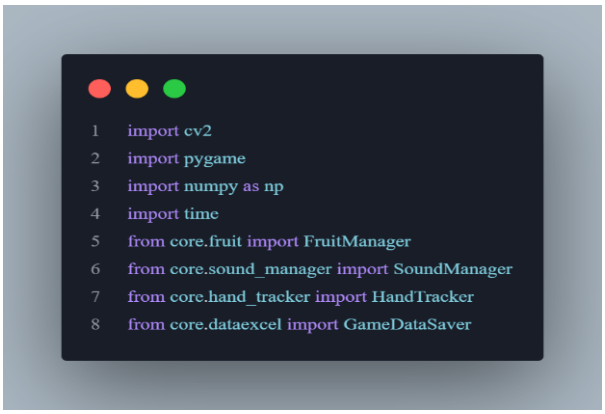
Fungsi _cleanup:

```
def _cleanup(self):
    self.cap.release()
    self.hand_tracker.close()
    print("Solo game closed safely.")
```

Fungsi **_cleanup** digunakan untuk membersihkan sumber daya permainan saat permainan berhenti atau keluar.

- self.cap.release()Melepaskan akses kamera yang digunakan untuk menangkap gerakan tangan pemain.
- self.hand_tracker.close() Menutup dan membersihkan modul HandTracker untuk menghentikan pemrosesan deteksi tangan.
- print("Solo game closed safely.")Menampilkan pesan pada konsol untuk memastikan bahwa game telah ditutup dengan aman.

3.1.6 Game/multi_game.py



```
import cv2
import pygame
import numpy as np
import time
from core.fruit import FruitManager
from core.sound_manager import SoundManager
from core.hand_tracker import HandTracker
from core.dataexcel import GameDataSaver
```

Import Library:

- cv2 (OpenCV): Mengakses kamera, membaca frame video, dan menggambar pointer tangan pada gambar.
- pygame: Membuat jendela game, menampilkan gambar ke layar, menangani event, dan memutar efek suara.
- numpy (np): Mengolah data gambar dalam bentuk array, misalnya mengubah ukuran dan menggabungkan gambar.
- time: Mengatur dan membaca waktu, seperti mencatat durasi permainan dan jeda tampilan
- FruitManager: Mengelola buah (spawn, gerak, jenis buah/bom) selama permainan.
- SoundManager: Memuat dan memutar efek suara seperti tebasan, bom, dan naik level.
- HandTracker: Mendeteksi dan melacak posisi tangan/jari pemain sebagai “pedang” pemotong buah
- GameDataSaver: Menyimpan data hasil permainan (skor, miss, level) ke file Excel untuk dokumentasi

Class MultiFruitNinjaGame:

```
class MultiFruitNinjaGame:
    def __init__(self, background_path="assets/themes/2.png"):
        pygame.init()
        pygame.mixer.init()
        self.WIDTH, self.HEIGHT = 1280, 720

        self.screen = pygame.display.set_mode((self.WIDTH, self.HEIGHT), pygame.RESIZABLE)
        pygame.display.set_caption("Ninja Fruits - Multiplayer")

        self.cap = cv2.VideoCapture(0)
        self.cap.set(3, self.WIDTH)
        self.cap.set(4, self.HEIGHT)

        self.background_path = background_path
        bg_img = cv2.imread(background_path)
        self.background = cv2.resize(bg_img, (self.WIDTH, self.HEIGHT)) if bg_img is not None else \
            255 * np.ones((self.HEIGHT, self.WIDTH, 3), dtype=np.uint8)

        self.sounds = SoundManager(["assets/sounds", "ninja_fruit_sounds"])
        self.sounds.load("slash", "slash.mp3")
        self.sounds.load("boom", "boom.mp3")
        self.sounds.load("levelup", "levelup.mp3")

        self.FONT = "assets/font/Gang_of_Three_Regular.ttf"

        self.hand_tracker = HandTracker(max_hands=2)
        self.fruit_manager = FruitManager(self.WIDTH, self.HEIGHT)
        self.clock = pygame.time.Clock()

        self.card_img = pygame.image.load("assets/images2/card.png")
        self.btn1_img = pygame.image.load("assets/images2/1.png")
        self.btn2_img = pygame.image.load("assets/images2/2.png")
        self.btn3_img = pygame.image.load("assets/images2/6.png")

        self.card_img = pygame.transform.scale(self.card_img, (350, 350))
        self.btn1_img = pygame.transform.scale(self.btn1_img, (90, 90))
        self.btn2_img = pygame.transform.scale(self.btn2_img, (90, 90))
        self.btn3_img = pygame.transform.scale(self.btn3_img, (90, 90))
        self.data_saver = GameDataSaver()
        self.save_message = None
        self.save_message_time = 0
        self.reset_game()
        self.start_time = time.time()
```

Class **MultiFruitNinjaGame** digunakan untuk menjalankan permainan Fruit Ninja mode Multi player, di mana permainan melibatkan dua pemain dengan dua deteksi jari

Fungsi __init__:

Fungsi **init** digunakan untuk menginisialisasi seluruh komponen utama
permainan sebelum permainan dimulai

- pygame.init() dan pygame.mixer.init() Menginisialisasi modul Pygame dan mixer untuk audio.
- self.WIDTH, self.HEIGHT = 1280, 720 Menentukan resolusi awal layar permainan.
- self.screen = pygame.display.set_mode(..., pygame.RESIZABLE) Membuat jendela permainan yang dapat diubah ukurannya.
- pygame.display.set_caption("Ninja Fruits Multiplayer") Menetapkan judul jendela permainan.
- self.cap = cv2.VideoCapture(0) Mengaktifkan kamera untuk mendeteksi gerakan tangan pemain.
- self.cap.set(3, self.WIDTH) dan self.cap.set(4, self.HEIGHT) Menetapkan resolusi kamera sesuai ukuran layar.
- self.background_path dan self.background Memuat dan mengubah ukuran gambar background sesuai resolusi layar. Jika gambar tidak ditemukan, digunakan background putih.
- self.sounds = SoundManager(...) Menginisialisasi pengelola suara dan memuat efek suara: "slash", "boom", "levelup".
- self.FONT = "assets/font/Gang_of_Three_Regular.ttf" Menetapkan font yang digunakan untuk teks dalam permainan.
- self.hand_tracker = HandTracker(max_hands=2) Menginisialisasi detektor tangan untuk dua pemain.
- self.fruit_manager=FruitManager(self.WIDTH,self.HEIGHT) Menginisialisasi pengelola buah dengan ukuran layar.
- self.clock = pygame.time.Clock() Mengatur clock untuk kontrol frame rate permainan.
- Memuat gambar UI: card_img, btn1_img, btn2_img, btn3_img dan mengubah ukurannya sesuai layout.
- self.data_saver = GameDataSaver() Menginisialisasi modul untuk menyimpan data permainan.
- self.save_message = None dan self.save_message_time = 0 Menyiapkan variabel untuk menampilkan pesan penyimpanan data.
- self.reset_game() Mengatur ulang status permainan multiplayer ke kondisi awal.
- self.start_time = time.time() Mencatat waktu mulai permainan.

Fungsi reset_game:

```
def reset_game(self):
    self.player1 = 0
    self.player2 = 0
    self.p1_alive = True
    self.p2_alive = True
    self.p1_prev = None
    self.p2_prev = None
    self.p1_buf = []
    self.p2_buf = []
    self.level = 1
    self.fruits = [self.fruit_manager.spawn_fruit_multi(1, True, True) for _ in range(7)]
    self.last_death_time = None
    self.game_over = False
    self.show_go_screen = False
    self.go_start_ticks = 0
    self.GO_DURATION = 2000
```

Fungsi **reset_game** digunakan untuk mengatur ulang status permainan multiplayer sebelum memulai sesi baru.

- self.player1 = 0 Mengatur skor pemain 1 menjadi 0.
- self.player2 = 0 Mengatur skor pemain 2 menjadi 0.
- self.p1_alive = True Menandai pemain 1 dalam keadaan hidup (*active*).
- self.p2_alive = True Menandai pemain 2 dalam keadaan hidup (*active*).
- self.p1_prev = None dan self.p2_prev = None Menghapus posisi tangan terakhir masing-masing pemain.
- self.p1_buf = [] dan self.p2_buf = [] Mengosongkan buffer untuk smoothing gerakan tangan pemain.
- self.level = 1 Mengatur level permainan kembali ke awal.
- self.fruits = [self.fruit_manager.spawn_fruit_multi(1, True, True) for _ in range(7)] Memunculkan 7 buah baru untuk permainan multiplayer di level 1.
- self.last_death_time = None Menghapus catatan waktu kematian terakhir pemain.
- self.game_over = False Menandai permainan belum berakhir.
- self.show_go_screen = False Menonaktifkan tampilan layar Game Over sementara.
- self.go_start_ticks = 0 Mengatur ulang penghitung waktu untuk animasi Game Over.
- self.GO_DURATION = 2000 Menentukan durasi tampilan layar Game Over dalam milidetik.

Fungsi _overlay_image:

```
def _overlay_image(self, bg, fg, x, y):
    h, w = fg.shape[:2]
    if len(fg.shape) == 3 and fg.shape[2] == 3:
        fg = np.dstack((fg, np.ones((h, w), dtype=np.uint8) * 255))
    if x < 0 or y < 0 or x + w > self.WIDTH or y + h > self.HEIGHT:
        return bg
    alpha = fg[:, :, 3] / 255.0
    for c in range(3):
        bg[y:y+h, x:x+w, c] = (alpha * fg[:, :, c] + (1 - alpha) * bg[y:y+h, x:x+w, c]).astype(np.uint8)
    return bg
```

Fungsi **_overlay_image** digunakan untuk menampilkan gambar objek (*foreground*) di atas gambar latar (*background*) dengan memanfaatkan nilai transparansi(*alpha blending*).

- h, w = fg.shape[:2] Digunakan untuk mengambil tinggi dan lebar gambar *foreground*.
- fg.shape[2] == 3 Digunakan untuk mengecek apakah gambar *foreground* belum memiliki channel alpha.
- np.dstack(...) Digunakan untuk menambahkan channel alpha pada gambar agar mendukung transparansi.
- if x < 0 or y < 0 or x + w > self.WIDTH or y + h > self.HEIGHT Digunakan untuk memastikan gambar tidak digambar di luar batas layar.
- return bg Mengembalikan background tanpa perubahan jika posisi gambar tidak valid.
- alpha = fg[:, :, 3] / 255.0 Digunakan untuk mengambil nilai transparansi dari channel alpha.
- for c in range(3) Digunakan untuk menggabungkan warna RGB foreground dan background menggunakan perhitungan *alpha blending*.
- astype(np.uint8) Digunakan untuk memastikan tipe data citra sesuai dengan format OpenCV.
- return bg Mengembalikan gambar latar yang telah ditambahkan objek *foreground*.

Fungsi _update_size:

```
def _update_size(self):
    new_w, new_h = self.screen.get_size()
    if new_w != self.WIDTH or new_h != self.HEIGHT:
        self.WIDTH, self.HEIGHT = new_w, new_h
        bg_img = cv2.imread(self.background_path)
        self.background = cv2.resize(bg_img, (self.WIDTH, self.HEIGHT)) if bg_img is not None else \
            255 * np.ones((self.HEIGHT, self.WIDTH, 3), dtype=np.uint8)
        self.fruit_manager.WIDTH = self.WIDTH
        self.fruit_manager.HEIGHT = self.HEIGHT
```

Fungsi **_update_size** digunakan untuk menyesuaikan ukuran layar permainan
ketika terjadi perubahan ukuran jendela (*window resize*).

- self.screen.get_size() Digunakan untuk mengambil ukuran terbaru jendela permainan.
- if new_w != self.WIDTH or new_h != self.HEIGHT Digunakan untuk mengecek apakah ukuran layar mengalami perubahan.
- self.WIDTH, self.HEIGHT = new_w, new_h Digunakan untuk memperbarui ukuran layar permainan.
- cv2.imread(self.background_path) Digunakan untuk membaca ulang gambar latar permainan.
- cv2.resize(...) Digunakan untuk menyesuaikan ukuran background dengan ukuran layar terbaru.
- np.ones(...) Digunakan sebagai background cadangan jika gambar latar tidak ditemukan.
- self.fruit_manager.WIDTH dan self.fruit_manager.HEIGHT Digunakan untuk memperbarui batas area kemunculan buah agar tetap sesuai dengan ukuran layar.

Fungsi _assign_hands:

Fungsi **_assign_hands** digunakan untuk menentukan posisi tangan masing-masing pemain berdasarkan hasil deteksi tangan dari kamera.

- p1_hand = None dan p2_hand = None Inisialisasi posisi tangan pemain 1 dan pemain 2.
- scale_x = self.WIDTH / cam_w dan scale_y = self.HEIGHT / cam_h
  Menghitung skala koordinat dari ukuran kamera ke ukuran layar permainan.

- if result.multi_hand_landmarks: Mengecek apakah ada tangan terdeteksi pada frame kamera.
- for hand_landmark in result.multi_hand_landmarks: Iterasi setiap tangan yang terdeteksi.
- lx = int(hand_landmark.landmark[8].x * cam_w) ly = int(hand_landmark.landmark[8].y * cam_h) Menghitung posisi jari telunjuk (landmark ke-8) dalam koordinat kamera.
- gx = int(lx * scale_x) dan gy = int(ly * scale_y) Mengubah koordinat kamera ke koordinat layar permainan.
- if gx < self.WIDTH // 2: Menentukan tangan pemain 1 jika berada di sisi kiri layar, dengan syarat self.p1_alive.
- else: Menentukan tangan pemain 2 jika berada di sisi kanan layar, dengan syarat self.p2_alive.
- return p1_hand, p2_hand Mengembalikan posisi tangan masing-masing pemain dalam koordinat layar.

```
1  def _assign_hands(self, result, cam_w, cam_h):
2      p1_hand = None
3      p2_hand = None
4
5      scale_x = self.WIDTH / cam_w
6      scale_y = self.HEIGHT / cam_h
7
8      if result.multi_hand_landmarks:
9          for hand_landmark in result.multi_hand_landmarks:
10             lx = int(hand_landmark.landmark[8].x * cam_w)
11             ly = int(hand_landmark.landmark[8].y * cam_h)
12             gx = int(lx * scale_x)
13             gy = int(ly * scale_y)
14
15             if gx < self.WIDTH // 2:
16                 if self.p1_alive:
17                     p1_hand = (gx, gy)
18             else:
19                 if self.p2_alive:
20                     p2_hand = (gx, gy)
21
22     return p1_hand, p2_hand
```

Fungsi _smooth_hand:

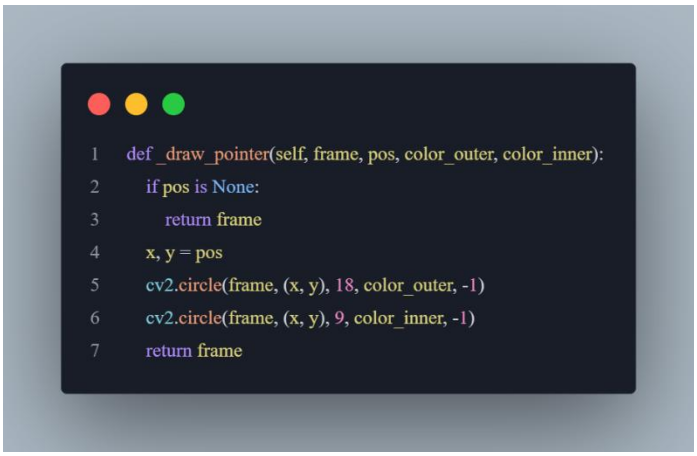
```
def _smooth_hand(self, hand_pos, buf, prev_pos):
    if hand_pos is None:
        return None

    x, y = hand_pos
    buf.append((x, y))
    if len(buf) > 5:
        buf.pop(0)

    sx = int(sum(px for px, py in buf) / len(buf))
    sy = int(sum(py for px, py in buf) / len(buf))

    if prev_pos is not None:
        sx = int(prev_pos[0] * 0.45 + sx * 0.55)
        sy = int(prev_pos[1] * 0.45 + sy * 0.55)

    return (sx, sy)
```

Fungsi **_smooth_hand** digunakan untuk menghaluskan posisi tangan pemain agar gerakan pointer di layar tidak terlalu jitter atau bergetar.

- if hand_pos is None: return None Mengembalikan None jika tidak ada tangan terdeteksi.
- buf.append((x, y)) dan if len(buf) > 5: buf.pop(0) Menyimpan posisi tangan terbaru dalam buffer maksimal 5 posisi untuk smoothing.

- sx = int(sum(px for px, py in buf) / len(buf)) sy = int(sum(py for px, py in buf) / len(buf)) Menghitung rata-rata posisi tangan dari buffer untuk menghasilkan gerakan lebih halus.
- if prev_pos is not None: Menambahkan anti-jitter tambahan dengan menggabungkan posisi sebelumnya dan posisi rata-rata baru: sx = int(prev_pos[0] * 0.45 + sx * 0.55) sy = int(prev_pos[1] * 0.45 + sy * 0.55)
- return (sx, sy) Mengembalikan posisi tangan yang sudah dihaluskan untuk digunakan pada pointer di layar.

Fungsi _draw_pointer:



```
1  def _draw_pointer(self, frame, pos, color_outer, color_inner):
2      if pos is None:
3          return frame
4      x, y = pos
5      cv2.circle(frame, (x, y), 18, color_outer, -1)
6      cv2.circle(frame, (x, y), 9, color_inner, -1)
7      return frame
```

Fungsi **_draw_pointer** digunakan untuk menggambar indikator pointer (misalnya jari pemain) pada frame kamera**,** sehingga posisi tangan lebih mudah dilihat pemain.

- if pos is None: return frame Mengembalikan frame tanpa perubahan jika tidak ada posisi pointer yang tersedia.
- x, y = pos Menentukan koordinat pointer dari posisi yang diberikan.
- cv2.circle(frame, (x, y), 18, color_outer, -1) Menggambar lingkaran luar dengan radius 18 px dan warna color_outer sebagai indikator utama.
- cv2.circle(frame, (x, y), 9, color_inner, -1) Menggambar lingkaran dalam dengan radius 9 px dan warna color_inner untuk menambah efek visual dan membedakan pusat pointer.
- return frame Mengembalikan frame yang sudah digambar pointer untuk ditampilkan di layar permainan.

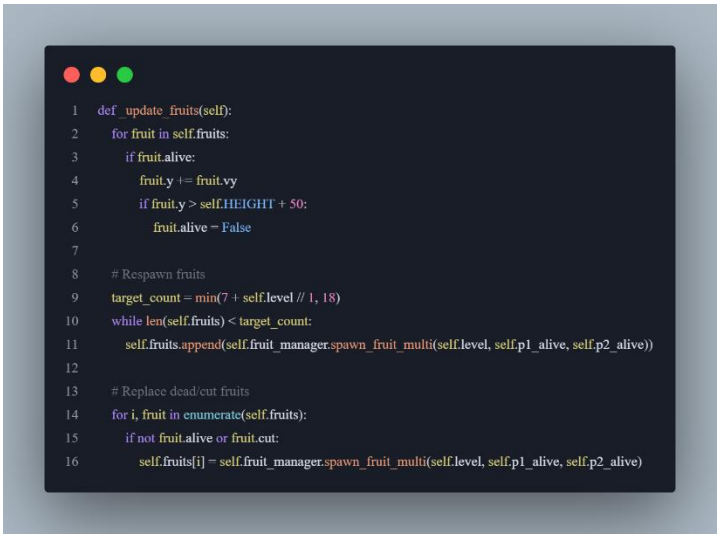
Fungsi _handle_hands_and_slice:

```python
def _handle_hands_and_slice(self, cam_frame, frame, now):
    result = self.hand_tracker.process(cam_frame)
    cam_h, cam_w = cam_frame.shape[:2]

    p1_hand, p2_hand = self._assign_hands(result, cam_w, cam_h)

    # Smooth and draw P1
    if self.p1_alive and p1_hand is not None:
        smooth_p1 = self._smooth_hand(p1_hand, self.p1_buf, self.p1_prev)
        if smooth_p1:
            frame = self._draw_pointer(frame, smooth_p1, (0, 255, 255), (0, 180, 255))
            self.p1_prev = smooth_p1

    # Smooth and draw P2
    if self.p2_alive and p2_hand is not None:
        smooth_p2 = self._smooth_hand(p2_hand, self.p2_buf, self.p2_prev)
        if smooth_p2:
            frame = self._draw_pointer(frame, smooth_p2, (0, 200, 255), (0, 120, 255))
            self.p2_prev = smooth_p2

    # Slice detection
    for fruit in self.fruits:
        if not fruit.alive or fruit.cut:
            continue

        fx, fy = fruit.x, fruit.y
        fh, fw = fruit.img.shape[:2]

        if self.p1_alive and self.p1_prev is not None:
            px, py = self.p1_prev
            if fx < px < fx + fw and fy < py < fy + fh:
                fruit.cut = True
                if fruit.is_bomb and not self.game_over:
                    self.p1_alive = False
                    self.sounds.play("boom")

                    if not self.p2_alive:
                        self.game_over = True
                        self.show_go_screen = True
                        self.go_start_ticks = pygame.time.get_ticks()
                        self.last_death_time = now
                else:
                    self.player1 += 1
                    self.sounds.play("slash")
                    if (self.player1 + self.player2) % 30 == 0:
                        self.level += 1
                        self.sounds.play("levelup")

        if self.p2_alive and self.p2_prev is not None and not fruit.cut:
            px, py = self.p2_prev
            if fx < px < fx + fw and fy < py < fy + fh:
                fruit.cut = True
                if fruit.is_bomb and not self.game_over:
                    self.p2_alive = False
                    self.sounds.play("boom")

                    if not self.p1_alive:
                        self.game_over = True
                        self.show_go_screen = True
                        self.go_start_ticks = pygame.time.get_ticks()
                        self.last_death_time = now
                else:
                    self.player2 += 1
                    self.sounds.play("slash")
                    if (self.player1 + self.player2) % 30 == 0:
                        self.level += 1
                        self.sounds.play("levelup")

    return frame
```

Fungsi **_handle_hands_and_slice** digunakan untuk memproses deteksi tangan pemain, menggambar pointer, dan mendeteksi pemotongan buah dalam permainan multiplayer.

- result=self.hand_tracker.process(cam_frame) Memproses frame kamera untuk mendeteksi posisi tangan pemain.
- p1_hand, p2_hand = self._assign_hands(result, cam_w, cam_h) Menentukan koordinat tangan pemain 1 dan pemain 2 berdasarkan hasil deteksi.
- Smooth dan gambar pointer P1 dan P2
    - _smooth_hand(p1_hand, self.p1_buf, self.p1_prev) Menghaluskan posisi tangan pemain 1 agar gerakan pointer lebih stabil.
    - _draw_pointer(frame, smooth_p1, (0, 255, 255), (0, 180, 255)) Menggambar pointer pemain 1.
    - Proses serupa dilakukan untuk pemain 2 dengan warna pointer berbeda.
- Deteksi pemotongan buah
    - Memeriksa setiap buah yang masih hidup (alive) dan belum terpotong (cut).
    - Jika koordinat pointer pemain berada dalam area buah:
        - fruit.cut = True menandai buah sebagai terpotong.
        - Jika buah adalah **bom**, pemain yang menyentuhnya mati (p1_alive atau p2_alive = False) dan jika kedua pemain mati, permainan berakhir (game_over = True).
        - Jika buah biasa, skor pemain bertambah dan efek suara diputar (slash).
        - Level naik setiap 30 buah yang berhasil dipotong (levelup).
- return frame Mengembalikan frame yang sudah digambar pointer dan diupdate interaksi buah**.**

Fungsi _update_fruits:



```
def _update_fruits(self):
    for fruit in self.fruits:
        if fruit.alive:
            fruit.y += fruit.vy
            if fruit.y > self.HEIGHT + 50:
                fruit.alive = False

    # Respawn fruits
    target_count = min(7 + self.level // 1, 18)
    while len(self.fruits) < target_count:
        self.fruits.append(self.fruit_manager.spawn_fruit_multi(self.level, self.p1_alive, self.p2_alive))

    # Replace dead/cut fruits
    for i, fruit in enumerate(self.fruits):
        if not fruit.alive or fruit.cut:
            self.fruits[i] = self.fruit_manager.spawn_fruit_multi(self.level, self.p1_alive, self.p2_alive)
```

Fungsi **_update_fruits** digunakan untuk memperbarui posisi buah, menangani respawn, dan mengganti buah yang sudah mati atau terpotong dalam permainan multiplayer.

- for fruit in self.fruits: Mengiterasi setiap buah yang ada di layar.
- if fruit.alive: fruit.y += fruit.vy Menggerakkan buah yang masih hidup secara vertikal sesuai kecepatan vy**.**
- if fruit.y > self.HEIGHT + 50: fruit.alive = False Menandai buah sebagai mati jika melewati batas bawah layar dengan buffer 50 px.
- Respawn buah baru
- target_count = min(7 + self.level // 1, 18) Menentukan jumlah buah yang diinginkan berdasarkan level, dengan maksimal 18 buah.
- while len(self.fruits) < target_count: Menambahkan buah baru hingga jumlah buah mencapai target_count.
- Ganti buah mati atau terpotong
- for i, fruit in enumerate(self.fruits): if not fruit.alive or fruit.cut: Mengganti buah yang sudah mati atau terpotong dengan buah baru dari fruit_manager.

Fungsi _setup_gameover_ui:

Fungsi **_setup_gameover_ui** digunakan untuk mengatur tata letak (*layout*) antarmuka Game Over, termasuk kartu hasil permainan dan tombol menu.

- cx = self.WIDTH // 2 Digunakan untuk menentukan titik tengah horizontal layar.
- cy = self.HEIGHT // 2 Digunakan untuk menentukan titik tengah vertikal layar.
- self.card_img.get_rect(center=(cx, cy - 80)) Digunakan untuk memposisikan kartu hasil permainan di bagian tengah layer dengan sedikit pergeseran ke atas.
- y = cy + 165 Digunakan sebagai posisi vertikal untuk menempatkan tombol-tombol menu.
- self.btn1_img.get_rect(center=(cx - 150, y)) Digunakan untuk menempatkan tombol pertama di sisi kiri.
- self.btn2_img.get_rect(center=(cx,y))Digunakan untuk menempatkan tombol kedua di bagian tengah.
- self.btn3_img.get_rect(center=(cx + 150, y)) Digunakan untuk menempatkan tombol ketiga di sisi kanan.

Fungsi run:

fungsi **run** digunakan untuk menjalankan loop utama permainan multiplayer, menangani input pemain, memproses logika permainan, dan menampilkan antarmuka.

- success, cam = self.cap.read() Membaca frame kamera untuk deteksi tangan pemain. Jika gagal, loop menunggu 0,05 detik sebelum mencoba lagi.
- cam = cv2.flip(cam, 1) Membalik frame kamera secara horizontal agar gerakan pemain sesuai dengan arah layar.
- self._update_size() Memperbarui ukuran layar dan background jika jendela di-resize.
- frame = self.background.copy() Membuat frame awal dari background untuk rendering selanjutnya.
- now = time.time() Mencatat waktu saat ini untuk keperluan logika permainan.
- frame = self._handle_hands_and_slice(cam, frame, now) Memproses posisi tangan pemain, smoothing gerakan, dan mendeteksi pemotongan buah.
- self._update_fruits() Memperbarui posisi buah, mengganti buah yang mati atau terpotong, serta menyesuaikan jumlah buah sesuai level.
- for fruit in self.fruits: Me-render semua buah hidup di layar menggunakan _overlay_image.
- if not self.game_over: Menampilkan skor pemain, level, dan garis pemisah tengah layar jika permainan masih berjalan.

- frame_rgb = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB) Mengonversi frame ke format RGB untuk ditampilkan di Pygame.
- surface = pygame.surfarray.make_surface(frame_rgb.swapaxes(0, 1)) Membuat surface Pygame dari frame kamera.
- Game Over Handling**:**
    - Membuat overlay gelap dengan transparansi.
    - Menampilkan teks GAME OVER selama GO_DURATION.
    - Setelah itu, menampilkan kartu hasil permainan, skor masing-masing pemain, level, dan tombol interaksi (btn1, btn2, btn3).
- Pesan penyimpanan data **(**save_message**):**Ditampilkan selama 2,5 detik jika pemain menekan tombol simpan data.
- pygame.display.flip() dan self.clock.tick(20) Memperbarui layar dan mengatur frame rate sekitar 20 FPS.
- Event Handling:
    - pygame.QUIT → membersihkan sumber daya dan keluar.
    - KEYDOWN K_ESCAPE → kembali ke menu utama.
    - KEYDOWN K_r saat game over → reset permainan.
    - MOUSEBUTTONDOWN saat game over:
        - Klik btn1 → kembali ke menu utama.
        - Klik btn2 → reset permainan.
        - Klik btn3 → menyimpan data multiplayer dan menampilkan pesan konfirmasi.
- self._cleanup() dan return "menu" Menutup sumber daya dan kembali ke menu utama saat loop permainan selesai.

```python
def run(self):
    running = True
    while running:
        success, cam = self.cap.read()
        if not success:
            time.sleep(0.05)
            continue
        cam = cv2.flip(cam, 1)

        self._update_size()
        frame = self.background.copy()
        now = time.time()

        frame = self._handle_hands_and_slice(cam, frame, now)
        self._update_fruits()

        for fruit in self.fruits:
            if fruit.alive:
                frame = self._overlay_image(frame, fruit.img, int(fruit.x), int(fruit.y))

        if not self.game_over:
            cv2.putText(frame, f"Player1: {self.player1}", (30, 60),
                        cv2.FONT_HERSHEY_SIMPLEX, 1.3, (0,255,255), 3)
            cv2.putText(frame, f"Player2: {self.player2}", (30, 120),
                        cv2.FONT_HERSHEY_SIMPLEX, 1.3, (0,200,255), 3)
            cv2.putText(frame, f"Level: {self.level}", (30, 180),
                        cv2.FONT_HERSHEY_SIMPLEX, 1.0, (255,255,255), 2)

            cv2.line(frame, (self.WIDTH//2, 0), (self.WIDTH//2, self.HEIGHT), (30,30,30), 2)

        frame_rgb = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
        surface = pygame.surfarray.make_surface(frame_rgb.swapaxes(0, 1))
        self.screen.blit(surface, (0, 0))

        # Game Over
        if self.game_over:
            # overlay gelap
            overlay = pygame.Surface((self.WIDTH, self.HEIGHT), pygame.SRCALPHA)
            overlay.fill((0, 0, 0, 160))
            self.screen.blit(overlay, (0, 0))

            elapsed = pygame.time.get_ticks() - self.go_start_ticks

            if self.show_go_screen and elapsed < self.GO_DURATION:
                font_go = pygame.font.Font(self.FONT, 96)
                go_text = font_go.render("GAME OVER", True, (255, 60, 60))
                go_rect = go_text.get_rect(center=(self.WIDTH // 2, self.HEIGHT // 2))
                self.screen.blit(go_text, go_rect)
            else:
                if self.show_go_screen:        # baru pertama kali masuk fase 2
                    self._setup_gameover_ui()
                self.show_go_screen = False

                self.screen.blit(self.card_img, self.card_rect)

                font_score = pygame.font.Font(self.FONT, 36)

                p1_text = font_score.render(f"P1 : {self.player1}", True, (0, 0, 0))
                p2_text = font_score.render(f"P2 : {self.player2}", True, (0, 0, 0))
                level_text = font_score.render(f"LEVEL : {self.level}", True, (0, 0, 0))

                gap = 35

                p1_rect = p1_text.get_rect(center=(self.card_rect.centerx, self.card_rect.centery - gap))
                p2_rect = p2_text.get_rect(center=(self.card_rect.centerx, self.card_rect.centery))
                level_rect = level_text.get_rect(center=(self.card_rect.centerx, self.card_rect.centery + gap))

                self.screen.blit(p1_text, p1_rect)
                self.screen.blit(p2_text, p2_rect)
                self.screen.blit(level_text, level_rect)

                self.screen.blit(self.btn1_img, self.btn1_rect)
                self.screen.blit(self.btn2_img, self.btn2_rect)
                self.screen.blit(self.btn3_img, self.btn3_rect)

        if self.save_message:
            if time.time() - self.save_message_time < 2.5:
                font = pygame.font.Font(self.FONT, 28)
                text = font.render(self.save_message, True, (0, 255, 0))
                text_rect = text.get_rect(center=(self.WIDTH // 2, self.HEIGHT // 2 + 265))
                self.screen.blit(text, text_rect)
            else:
                self.save_message = None

        pygame.display.flip()
        self.clock.tick(20)

        for event in pygame.event.get():
            if event.type == pygame.QUIT:
                self._cleanup()
                return "exit"

            if event.type == pygame.KEYDOWN:
                if event.key == pygame.K_ESCAPE:
                    self._cleanup()
                    return "menu"
                if self.game_over and event.key == pygame.K_r:
                    self.reset_game()

            if event.type == pygame.MOUSEBUTTONDOWN and self.game_over:
                mx, my = event.pos
                if self.btn1_rect.collidepoint(mx, my):
                    self._cleanup()
                    return "menu"
                elif self.btn2_rect.collidepoint(mx, my):
                    self.reset_game()
                elif self.btn3_rect.collidepoint(mx, my):
                    self.data_saver.save_multiplayer(
                        player1=self.player1,
                        player2=self.player2,
                        level=self.level
                    )
                    self.save_message = "Data multiplayer berhasil disimpan"
                    self.save_message_time = time.time()

    self._cleanup()
    return "menu"
```

Fungsi _cleanup:

Fungsi **cleanup** digunakan untuk membersihkan sumber daya permainan saat permainan berhenti atau keluar.

- self.cap.release() Melepaskan akses kamera yang digunakan untuk menangkap gerakan tangan pemain.
- self.hand_tracker.close() Menutup dan membersihkan modul HandTracker untuk menghentikan pemrosesan deteksi tangan.
- print("Multiplayer game closed safely.") Menampilkan pesan pada konsol untuk memastikan bahwa game multiplayer telah ditutup dengan aman..

3.1.7 ui/navigation.py

Library:

- import pygame Digunakan sebagai library utama untuk pengelolaan grafis, suara, dan interaksi pengguna pada game.
- import sys Digunakan untuk mengakses fungsi sistem, seperti menghentikan program.

- from game.solo_game import SoloFruitNinjaGame Digunakan untuk menjalankan mode permainan single player.
- from game.multi_game import MultiFruitNinjaGame Digunakan untuk menjalankan mode permainan multiplayer.

```
class MenuApp:
    def __init__(self):
        pygame.init()
        pygame.mixer.init()

        self.MENU_MUSIC = "assets/sounds/music.mp3"
        self.WIDTH, self.HEIGHT = 1280, 720
        self.BUTTON_SIZE = (320, 120)
        self.FONT = "assets/font/Gang_of_Three_Regular.ttf"

        self.screen = pygame.display.set_mode((self.WIDTH, self.HEIGHT), pygame.RESIZABLE)
        pygame.display.set_caption("Ninja Fruits Navigation")

        self.font_big = pygame.font.Font(self.FONT, 60)

        self.themes = {
            "kaktus": "assets/themes/1.png",
            "kayu": "assets/themes/2.png",
            "salju": "assets/themes/3.png",
            "sakura": "assets/themes/4.png",
        }
        self.selected_theme = "kayu"

        self.bg_main = pygame.image.load(self.themes[self.selected_theme])
        self.bg_main = pygame.transform.scale(self.bg_main, (self.WIDTH, self.HEIGHT))

        self.button_anim_state = {}

        self.theme_icons = {
            "kaktus": "assets/images2/kaktus.png",
            "kayu": "assets/images2/kayu.png",
            "salju": "assets/images2/salju.png",
            "sakura": "assets/images2/sakura.png",
        }

        self.back_img = pygame.image.load("assets/images2/kembali.png")
        self.back_img = pygame.transform.scale(self.back_img, (200, 90))
```

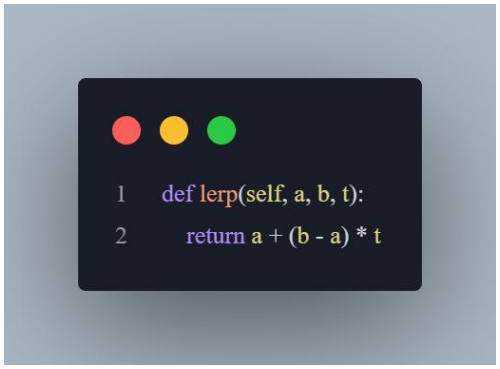
Class MenuApp:
fungsi __init__ :

Fungsi **__init__** pada kelas MenuApp digunakan untuk menginisialisasi bagian menu permainan, termasuk tampilan, musik, tema, dan ikon tombol.

- pygame.init() dan pygame.mixer.init() menginisialisasi Pygame dan modul audio untuk digunakan dalam menu.
- self.MENU_MUSIC = "assets/sounds/music.mp3" menyimpan path musik latar untuk menu utama.
- self.WIDTH, self.HEIGHT = 1280, 720 Menentukan resolusi jendela menu.
- self.BUTTON_SIZE = (320, 120) menyimpan ukuran standar tombol menu.

- self.FONT = "assets/font/Gang_of_Three_Regular.ttf" Menentukan font yang digunakan untuk teks pada menu.
- self.screen = pygame.display.set_mode((self.WIDTH, self.HEIGHT), pygame.RESIZABLE) Membuat layar Pygame dengan ukuran yang dapat diubah (*resizable*).
- pygame.display.set_caption("Ninja Fruits Navigation") Memberikan judul jendela aplikasi.
- self.font_big = pygame.font.Font(self.FONT, 60) Mengatur font besar untuk judul menu.
- self.themes = {...} dan self.selected_theme = "kayu" Menyimpan daftar tema dan memilih tema default.
- self.bg_main = pygame.image.load(self.themes[self.selected_theme]) Memuat gambar latar dari tema yang dipilih.
- self.bg_main = pygame.transform.scale(self.bg_main, (self.WIDTH, self.HEIGHT)) Menskalakan latar agar sesuai ukuran layar.
- self.button_anim_state = {} Menyimpan status animasi tombol untuk efek hover.
- self.theme_icons = {...} Menyimpan ikon visual untuk masing-masing tema.
- self.back_img = pygame.image.load("assets/images2/kembali.png") Memuat gambar tombol kembali.
- self.back_img = pygame.transform.scale(self.back_img, (200, 90)) Menskalakan tombol kembali sesuai ukuran yang diinginkan.

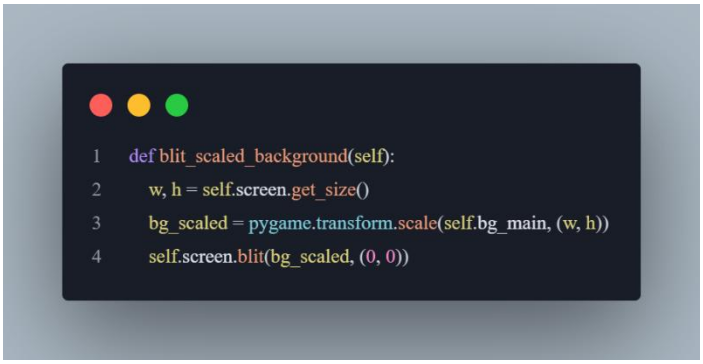
Fungsi lerp:



Fungsi lerp digunakan untuk menghitung interpolasi linier antara dua nilai, berguna untuk efek animasi yang halus.

- return a + (b - a) * t Menghasilkan nilai antara a dan b berdasarkan faktor t ($0 \leq t \leq 1$).

Fungsi blit_scaled_background:



Fungsi ini digunakan untuk menampilkan latar belakang yang disesuaikan dengan ukuran layar saat ini**.**

- w, h = self.screen.get_size() Mendapatkan ukuran layar saat ini.
- bg_scaled = pygame.transform.scale(self.bg_main, (w, h)) menskalakan gambar latar belakang agar sesuai ukuran layar.
- self.screen.blit(bg_scaled, (0, 0)) menampilkan latar belakang pada koordinat (0,0) layar.

Fungsi reset_layar:

Fungsi ini digunakan untuk mereset tampilan layar Pygame ketika menu dimuat ulang atau resolusi berubah.

- pygame.display.quit() dan pygame.display.init() Menutup dan menginisialisasi ulang tampilan Pygame.
- self.screen = pygame.display.set_mode((self.WIDTH, self.HEIGHT), pygame.RESIZABLE) membuat layar baru dengan ukuran tetap.
- self.bg_main = pygame.image.load(self.themes[self.selected_theme]) memuat ulang gambar latar belakang sesuai tema yang dipilih.
- self.bg_main = pygame.transform.scale(self.bg_main, (self.WIDTH, self.HEIGHT)) menskalakan latar belakang agar sesuai ukuran layar.

Fungsi fade_transition:

```
def fade_transition(self):
    width, height = self.screen.get_size()
    fade = pygame.Surface((width, height))
    fade.fill((0, 0, 0))
    for alpha in range(0, 255, 12):
        fade.set_alpha(alpha)
        self.screen.blit(fade, (0, 0))
        pygame.display.update()
        pygame.time.delay(18)
```

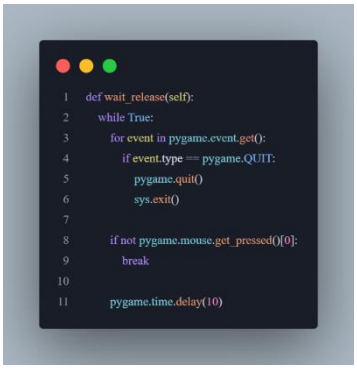
Fungsi ini digunakan untuk menampilkan efek transisi fade-in atau fade-out ketika berpindah antar menu atau mode permainan.

- fade = pygame.Surface((width, height)) membuat permukaan hitam seluas layar.
- fade.set_alpha(alpha) mengatur tingkat transparansi (alpha) secara bertahap dari 0 hingga 255.
- self.screen.blit(fade, (0, 0)) dan pygame.display.update() menampilkan permukaan transparan di atas layar untuk efek transisi.
- pygame.time.delay(18) memberikan jeda antar frame untuk menghasilkan efek transisi yang halus.

Fungsi wait_release:

Funsgi wait_release digunakan untuk menunggu hingga tombol mouse dilepas sebelum melanjutkan eksekusi, mencegah deteksi klik ganda.

- for event in pygame.event.get(): if event.type == pygame.QUIT: ...
  Menangani event keluar dari aplikasi, menutup Pygame dan menghentikan program.
- if not pygame.mouse.get_pressed()[0]: break keluar dari loop saat tombol kiri mouse dilepas.
- pygame.time.delay(10) memberikan jeda singkat untuk mencegah penggunaan CPU berlebihan selama menunggu.



```
def wait_release(self):
    while True:
        for event in pygame.event.get():
            if event.type == pygame.QUIT:
                pygame.quit()
                sys.exit()

        if not pygame.mouse.get_pressed()[0]:
            break

        pygame.time.delay(10)
```

Fungsi draw_image_button:

```
def draw_image_button(self, image_path, pos, scale=None, anim_speed=0.15):
    if scale is None:
        scale = self.BUTTON_SIZE

    img = pygame.image.load(image_path)
    btn_id = (image_path, pos)
    if btn_id not in self.button_anim_state:
        self.button_anim_state[btn_id] = 1.0

    base_rect = pygame.Rect(pos[0], pos[1], scale[0], scale[1])
    mouse = pygame.mouse.get_pos()
    click = pygame.mouse.get_pressed()

    is_hover = base_rect.collidepoint(mouse)
    target = 1.12 if is_hover else 1.0
    self.button_anim_state[btn_id] = self.lerp(self.button_anim_state[btn_id], target, anim_speed)

    s = self.button_anim_state[btn_id]
    new_w = int(scale[0] * s)
    new_h = int(scale[1] * s)

    img = pygame.transform.smoothscale(img, (new_w, new_h))
    rect = img.get_rect(center=base_rect.center)

    self.screen.blit(img, rect.topleft)

    if is_hover and click[0]:
        pygame.time.wait(150)
        return True
    return False
```

Fungsi ini digunakan untuk menampilkan tombol berbasis gambar dengan efek hover dan mendeteksi klik**.**

- img = pygame.image.load(image_path) memuat gambar tombol dari path yang diberikan.
- btn_id = (image_path, pos) dan self.button_anim_state[btn_id] = 1.0 menyimpan status animasi tombol untuk efek hover per tombol.
- base_rect = pygame.Rect(pos[0], pos[1], scale[0], scale[1]) membuat rect dasar untuk area tombol.
- is_hover = base_rect.collidepoint(mouse) mengecek apakah mouse berada di atas tombol.
- self.button_anim_state[btn_id] = self.lerp(..., target, anim_speed) mengatur skala tombol secara halus untuk efek hover.
- img = pygame.transform.smoothscale(img, (new_w, new_h)) menskalakan gambar tombol sesuai animasi.
- self.screen.blit(img, rect.topleft) menampilkan tombol di layar.
- if is_hover and click[0]: pygame.time.wait(150); return True Mengembalikan True jika tombol diklik, dengan jeda 150 ms untuk mencegah klik ganda.
- return False mengembalikan False jika tombol tidak diklik.

Funsgi mode_menu:

Fungsi mode_menu digunakan untuk menampilkan menu pemilihan mode permainan, termasuk mode Solo, Multiplayer, dan pengaturan tema.

- self.reset_layar() Mereset tampilan layar agar siap menampilkan menu.
- pygame.mixer.music.load(self.MENU_MUSIC)danygame.mixer.music.play(-1) memuat dan memutar musik latar menu secara terus-menerus.
- self.blit_scaled_background() menampilkan background yang disesuaikan dengan ukuran layar.
- image = pygame.image.load("assets/images2/45.png").convert_alpha() Memuat gambar judul/menu utama dan menampilkannya di bagian atas layar.
- draw_image_button("assets/images2/play.png", ...) Menampilkan tombol Play:
    - Jika diklik, memanggil wait_release() untuk menunggu pelepasan klik.
    - Memanggil fade_transition() untuk efek transisi.

    - Memasuki menu pemilihan mode dengan self.mode_menu().
- draw_image_button("assets/images2/exit.png", ...) Menampilkan tombol Exit:
    - Jika diklik, memanggil wait_release() dan menutup aplikasi dengan pygame.quit() dan sys.exit().
- pygame.display.update() memperbarui tampilan layar setiap frame.
- for event in pygame.event.get(): if event.type == pygame.QUIT: running = False menangani event keluar dari aplikasi melalui tombol close window.

```
def mode_menu(self):
    self.reset_layar()
    running = True
    while running:
        self.blit_scaled_background()
        width, height = self.screen.get_size()

        title = self.font_big.render("Select Mode", True, (255, 255, 0))
        self.screen.blit(title, (width // 2 - title.get_width() // 2, 80))

        btn_w, btn_h = self.BUTTON_SIZE
        center_x = width // 2 - btn_w // 2
        solo_y = 220
        multi_y = solo_y + btn_h + 10
        theme_y = multi_y + btn_h + 10

        if self.draw_image_button("assets/images2/solo.png", (center_x, solo_y)):
            self.wait_release()
            self.fade_transition()
            game = SoloFruitNinjaGame(self.themes[self.selected_theme])
            result = game.run()
            if result == "menu":
                self.reset_layar()
                pygame.mixer.music.load(self.MENU_MUSIC)
                pygame.mixer.music.play(-1)
                self.fade_transition()
                return
            elif result == "exit":
                pygame.quit()
                sys.exit()

        if self.draw_image_button("assets/images2/multi.png", (center_x, multi_y)):
            self.wait_release()
            self.fade_transition()
            game = MultiFruitNinjaGame(self.themes[self.selected_theme])
            result = game.run()
            if result == "menu":
                self.reset_layar()
                pygame.mixer.music.load(self.MENU_MUSIC)
                pygame.mixer.music.play(-1)
                self.fade_transition()
                return
            elif result == "exit":
                pygame.quit()
                sys.exit()

        if self.draw_image_button("assets/images2/tema.png", (center_x, theme_y)):
            self.wait_release()
            self.fade_transition()
            self.theme_menu()

        exit_x = 20
        exit_y = 20
        if self.draw_image_button("assets/images2/1.png", (exit_x, exit_y),scale=(70,70)):
            self.wait_release()
            self.fade_transition()
            return

        pygame.display.update()
        for event in pygame.event.get():
            if event.type == pygame.QUIT:
                running = False
```

Fungsi theme_menu:

```python
def theme_menu(self):
    running = True
    while running:
        WIDTH, HEIGHT = self.screen.get_size()
        bg = pygame.image.load(self.themes[self.selected_theme])
        bg = pygame.transform.scale(bg, (WIDTH, HEIGHT))
        self.screen.blit(bg, (0, 0))

        title = self.font_big.render("Pilih tema", True, (255, 255, 0))
        self.screen.blit(title, (WIDTH // 2 - title.get_width() // 2, 60))

        num_themes = len(self.themes)
        img_w, img_h = 260, 160
        padding = 60
        total_w = num_themes * img_w + (num_themes - 1) * padding
        start_x = (WIDTH - total_w) // 2
        y_img = 240
        y_icon = y_img + img_h + 25

        mouse_pos = pygame.mouse.get_pos()
        click = pygame.mouse.get_pressed()[0]
        current_x = start_x

        for name, path in self.themes.items():
            img = pygame.image.load(path)
            img = pygame.transform.scale(img, (img_w, img_h))
            img_rect = pygame.Rect(current_x, y_img, img_w, img_h)
            self.screen.blit(img, img_rect.topleft)

            if name == self.selected_theme:
                pygame.draw.rect(self.screen, (255, 215, 0), img_rect.inflate(10, 10), 4)

            icon_path = self.theme_icons.get(name)
            if icon_path:
                icon = pygame.image.load(icon_path)
                if img_rect.collidepoint(mouse_pos):
                    icon = pygame.transform.scale(icon, (165, 165))
                else:
                    icon = pygame.transform.scale(icon, (140, 140))
                icon_rect = icon.get_rect(center=(current_x + img_w // 2, y_icon + 35))
                self.screen.blit(icon, icon_rect.topleft)

            if click and (img_rect.collidepoint(mouse_pos) or icon_rect.collidepoint(mouse_pos)):
                self.selected_theme = name
                self.bg_main = pygame.image.load(self.themes[self.selected_theme])
                self.fade_transition()
                return

            current_x += img_w + padding

        back_rect = self.back_img.get_rect(center=(WIDTH // 2, 600))
        self.screen.blit(self.back_img, back_rect.topleft)

        if back_rect.collidepoint(mouse_pos) and click:
            self.wait_release()
            self.fade_transition()
            return

        pygame.display.update()

        for event in pygame.event.get():
            if event.type == pygame.QUIT:
                pygame.quit()
                sys.exit()
```

Fungsi theme_menu digunakan untuk menampilkan menu pemilihan tema permainan, memungkinkan pengguna memilih latar belakang dan ikon visual.

- WIDTH, HEIGHT = self.screen.get_size() mendapatkan ukuran layar saat ini untuk menyesuaikan tampilan tema.
- bg = pygame.image.load(self.themes[self.selected_theme]) memuat gambar latar belakang tema yang sedang dipilih.
- bg = pygame.transform.scale(bg, (WIDTH, HEIGHT)) dan self.screen.blit(bg, (0, 0)) menskalakan dan menampilkan latar belakang agar sesuai ukuran layar.
- title = self.font_big.render("Pilih tema", True, (255, 255, 0)) Menampilkan judul menu tema di bagian atas layar.
- Mengatur posisi dan ukuran tema:
    - img_w, img_h = 260, 160 → ukuran gambar tema.
    - padding = 60 → jarak antar tema.
    - start_x dan y_img → posisi awal menampilkan tema di layar.
- Menampilkan setiap tema:
    - Memuat gambar tema dan menampilkannya di posisi yang sesuai.
    - Jika tema sedang dipilih, menambahkan border emas (pygame.draw.rect) untuk menandai tema aktif.
- Menampilkan ikon tema:
    - Memuat ikon untuk setiap tema dari self.theme_icons.
    - Menskalakan ikon lebih besar saat mouse berada di atas tema (165x165) dan lebih kecil saat tidak (140x140).
- Pemilihan tema:
    - Jika klik pada gambar atau ikon, mengubah self.selected_theme.
    - Memuat ulang background (self.bg_main) dan memanggil fade_transition().
    - Mengembalikan kontrol ke menu sebelumnya setelah pemilihan.
- Tombol kembali:
    - Menampilkan tombol kembali (self.back_img) di bawah layar.
    - Jika diklik, menunggu pelepasan klik, memanggil fade_transition(), dan kembali ke menu sebelumnya.
- pygame.display.update() memperbarui tampilan layar setiap frame.
- for event in pygame.event.get(): if event.type == pygame.QUIT: pygame.quit(); sys.exit() menangani event keluar dari aplikasi melalui tombol close window.

3.2 Screenshot Aplikasi

Berikut adalah tangkapan layar dari setiap slide permainan ini:

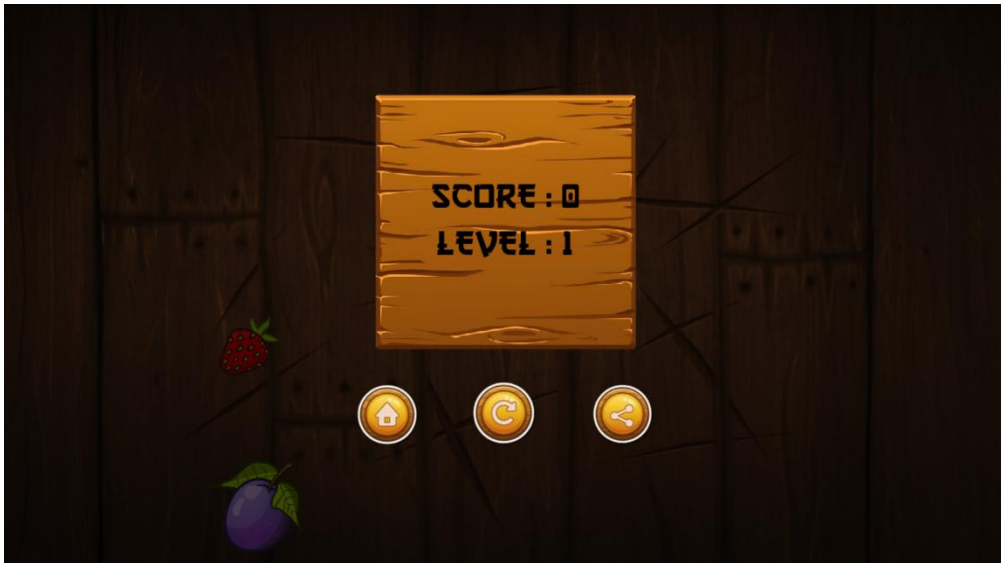
3.2.1 Main Screen

Gambar 3.1Main Screen

3.2.2 Menu Mode

3.2.3 Solo Player

Gambar 3.3 Dalam Game



Gambar 3.4 GameOver

3.2.4 Multi Player

Gambar 3.5 (Dalam Game)

Gambar 3.6 (GameOver)

3.2.5 Tema

Gambar 3.7

BAB 4

LAMPIRAN

4.1 File

Berikut adalah folder lengkap berisi aset dan kode lengkap Fruits Ninja:

https://drive.google.com/drive/folders/1KxLFhe-awbZnZT5Hp8yuCQSDBAHXewn?usp=drive_link

DAFTAR PUSTAKA

Fitri Barokah, Sari, Z., & Chanifudin. (2024). Peluang Dan Tantangan Pendidikan Karakter Di Era Digital. *AL-MUADDIB: Jurnal Kajian Ilmu Kependidikan*, *6*(3), 721–737. https://doi.org/10.46773/muaddib.v6i3.1209

Game, P., Puzzle, E., Computer, B., Hand, V., Untuk, T., & Usia, A. (2025). *PENGEMBANGAN GAME EDUKASI PUZZLE BERBASIS COMPUTER VISION HAND*. *16*(c), 16–22.

Pun, C., & Zhu, H. (2011). *Real-Time Hand Gesture Recognition using Motion Tracking Received : 28-06-2010 Accepted : 31-01-2011*. *4*(2), 277–286.