LAPORAN AKHIR PEMROGAMAN BERORIENTASI OBJEK

Battle Hero: Game Edukasi Sederhana (Paygame) Berbasis OOP sebagai Media Pembelajaran Coding Dasar untuk Siswa Sekolah Dasar

Dosen Pengampu:

M Adamu Islam Mashuri, S.Tr.T., M.Tr.Kom

Disusun Oleh:

Fikro Nabila (24091397110)

PROGRAM STUDI MANAJEMEN INFORMATIKA

FAKULTAS VOKASI

UNIVERSITAS NEGERI SURABAYA

TAHUN 2025

DAFTAR ISI

DAFTAR GAMBAR

BAB I
PENDAHULUAN

1.1 Latar Belakang

Perkembangan teknologi informasi mendorong pentingnya pengenalan kemampuan berpikir logis dan terstruktur sejak usia dini. Namun, pembelajaran coding untuk siswa Sekolah Dasar sering kali masih disampaikan secara abstrak dan kurang kontekstual, sehingga sulit dipahami dan kurang menarik bagi siswa. Salah satu pendekatan yang dinilai efektif untuk meningkatkan minat belajar anak adalah melalui media permainan edukatif. Game edukasi memungkinkan siswa belajar melalui pengalaman langsung, sehingga proses pembelajaran menjadi lebih menyenangkan dan interaktif. Berdasarkan permasalahan tersebut, dikembangkan sebuah game edukasi sederhana berjudul Battle Hero menggunakan pustaka Pygame.

Game ini dirancang untuk mengenalkan dasar-dasar cara berpikir pemrograman melalui interaksi karakter, aturan permainan, serta alur instruksi dan respon sistem yang sederhana. Dalam pengembangannya, game ini menerapkan konsep Object-Oriented Programming (OOP) sebagai pendekatan perancangan sistem, di mana setiap karakter dan komponen permainan dimodelkan sebagai objek dengan atribut dan perilaku tertentu. Batasan materi coding dalam game Battle Hero difokuskan pada pengenalan cara berpikir dasar pemrograman yang disesuaikan dengan tingkat kognitif siswa Sekolah Dasar. Materi yang dikenalkan meliputi pemahaman urutan instruksi, logika sebab–akibat, pengenalan aturan atau kondisi sederhana, konsep objek dalam konteks dunia nyata, serta interaksi input dan output melalui permainan.

Game ini tidak bertujuan mengajarkan sintaks bahasa pemrograman atau konsep teknis seperti penulisan kode, struktur data, maupun Object-Oriented Programming secara formal kepada siswa. Konsep OOP digunakan pada sisi pengembang sebagai pendekatan perancangan sistem untuk menghasilkan struktur program yang terorganisasi dan mudah dikembangkan. Dengan demikian, diharapkan game Battle Hero dapat menjadi media pembelajaran alternatif yang membantu siswa Sekolah Dasar memahami dasar-dasar coding secara sederhana, kontekstual, dan menyenangkan.

1.2 Tujuan Pengembangan

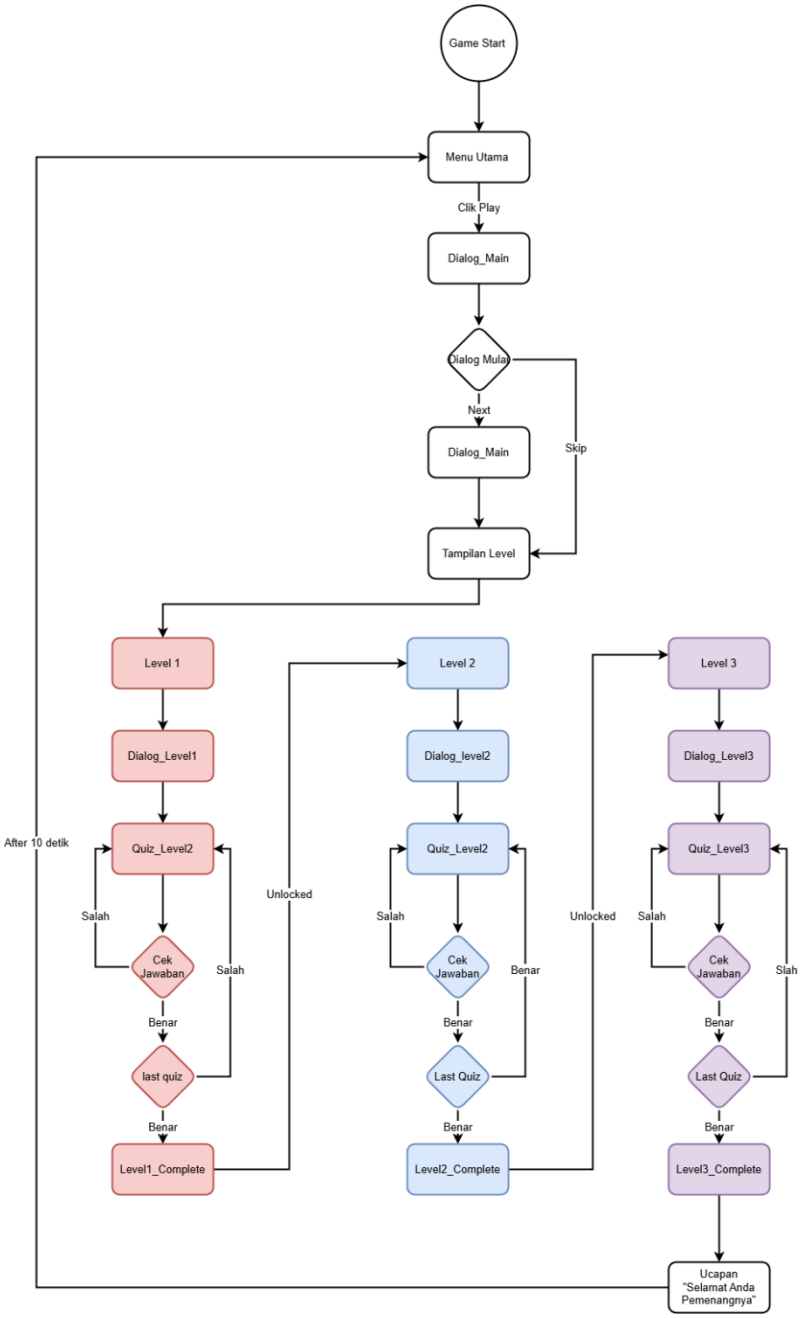
Tujuan umum dari pengembangan game Battle Hero adalah mengembangkan sebuah game edukasi sederhana berbasis Pygame yang dapat digunakan sebagai media pembelajaran alternatif untuk mengenalkan cara berpikir dasar pemrograman kepada siswa Sekolah Dasar secara interaktif dan menyenangkan.

BAB II
PERANCANGAN APLIKASI

2.1 Jenis Proyek

Proyek ini adalah sebuah game edukasi sederhana menggunakan paygame sebagai media pembelajaran coding dasar untuk siswa Sekolah Dasar, dengan mengimplementasikan konsep Object-Oriented Progamming (OOP) yang didukung dengan penggunaan struktur data dan implementasi interaksi untuk pengguna.

2.2 Alur Aplikasi



Gambar 2. 1 Flowchart Alur Permainan

Ketika game dijalankan, PayGame akan melakukan proses inisialisasi awal. Pada tahap ini sistem Pygame diinisialisasi, audio sistem disiapkan untuk memutar background music dan sound effect, serta seluruh aset game seperti gambar latar, karakter, dan tombol dimuat ke dalam memori. Selain itu, data game berupa dialog cerita dan soal kuis juga diproses. Setelah seluruh proses inisialisasi berhasil, game akan diarahkan ke tampilan menu utama.

1. **Menu Utama (Main Menu)**
   Pada tampilan menu utama, pemain akan melihat background utama dengan satu tombol utama yaitu tombol PLAY yang berada di tengah layar. Background music diputar secara looping untuk meningkatkan suasana permainan. Ketika pemain menekan tombol PLAY, maka game akan berpindah ke cerita pembuka.

2. **Cerita Pembuka (Dialog Awal)**
   Pada tahap ini, game menampilkan urutan dialog cerita sebagai pengantar awal permainan. Pemain dapat menekan tombol NEXT untuk melanjutkan ke dialog berikutnya atau SKIP untuk melewati seluruh dialog. Setelah seluruh dialog pembuka selesai ditampilkan, sistem secara otomatis akan mengarahkan pemain ke menu pemilihan level.

3. **Pemilihan Level**
   Pada menu pemilihan level, pemain akan melihat tiga pilihan level, yaitu Level 1, Level 2, dan Level 3. Pada awal permainan, hanya Level 1 yang terbuka, sedangkan Level 2 dan Level 3 masih terkunci. Pemain dapat memilih Level 1 untuk memulai permainan. Level berikutnya akan terbuka secara otomatis setelah pemain menyelesaikan level sebelumnya.

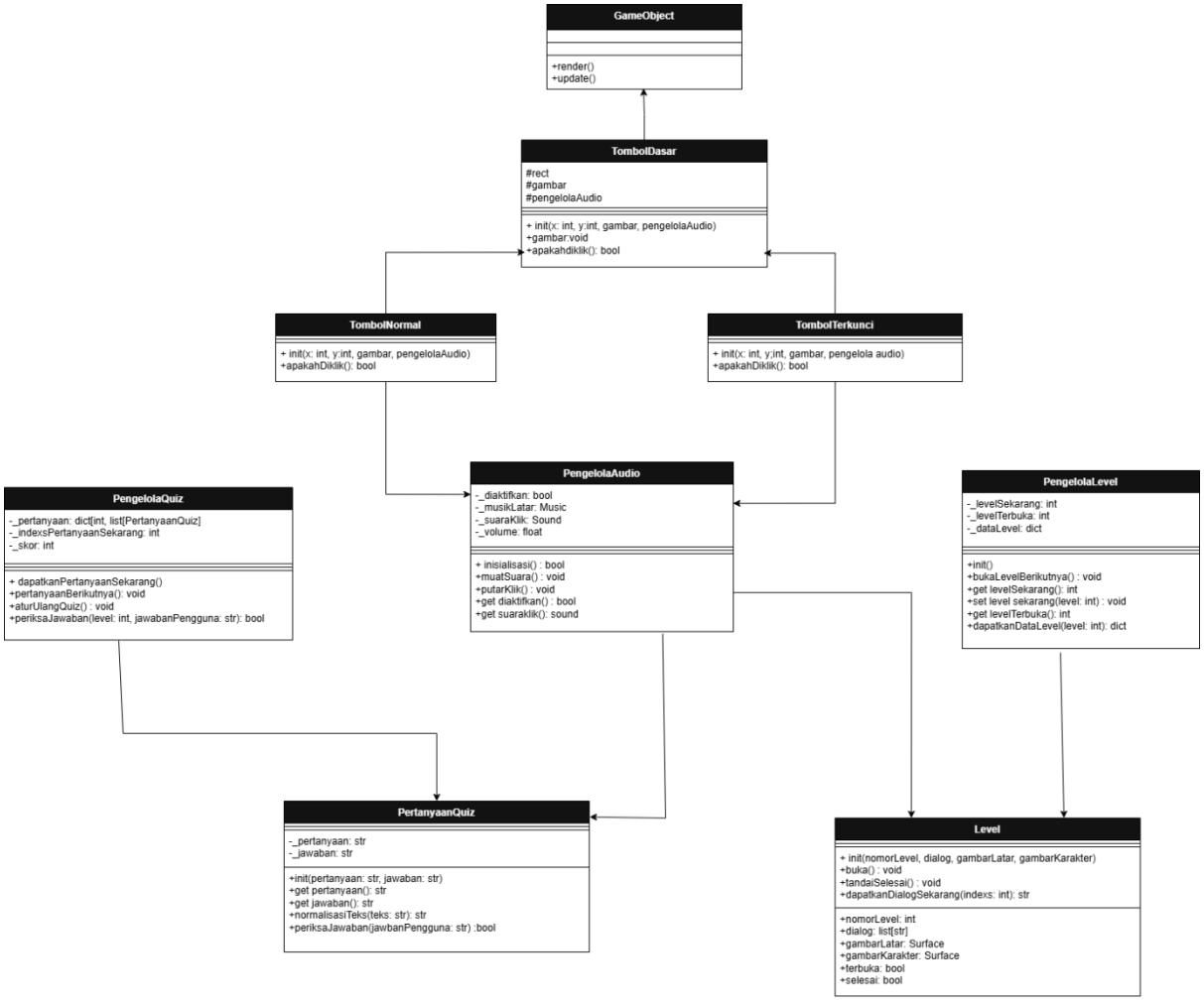
4. **Dialog Level**
   Setiap level diawali dengan dialog perkenalan karakter musuh. Pada Level 1 pemain akan berhadapan dengan karakter Jelly, Level 2 dengan Mummis, dan Level 3 dengan Ablis. Tampilan dialog sama seperti dialog pembuka, lengkap dengan tombol NEXT untuk melanjutkan ke dialog berikutnya atau SKIP untuk melewati seluruh dialog. Setelah dialog level selesai, pemain akan langsung masuk ke tahap kuis.

5. **Quiz & Penyelesaian**
   Tahap kuis merupakan inti dari game. Pemain akan diberikan beberapa soal pemrograman Python sesuai dengan tingkat level. Pemain dapat memasukkan jawaban melalui keyboard atau menekan tombol JAWAB menggunakan mouse. Setiap jawaban yang dimasukkan akan diproses oleh sistem dengan cara menormalkan input dan membandingkannya dengan jawaban yang benar. Sistem akan menampilkan feedback jawaban benar atau salah. Jika jawaban benar, pemain akan melanjutkan ke soal berikutnya, sedangkan jika salah, pemain tetap berada pada soal yang sama.
   Setelah seluruh soal dalam satu level selesai, sistem akan memproses penyelesaian level. Untuk Level 1 dan Level 2, penyelesaian level akan membuka level berikutnya dan mengarahkan pemain kembali ke menu pemilihan level. Sedangkan pada Level 3,

setelah soal terakhir diselesaikan, pemain akan langsung diarahkan ke tampilan kemenangan.

6. **Tampilan Kemenangan**
   Pada tampilan kemenangan, game menampilkan latar belakang berwarna biru gelap, efek kembang api, serta teks ucapan selamat kepada pemain. Tampilan ini tidak memiliki tombol interaktif. Setelah 10 detik, sistem secara otomatis akan mengembalikan pemain ke menu utama, dan seluruh status permainan akan diatur ulang.

2.3 Class Diagram



Gambar 2. 2 Class Diagram

Pada bagian paling atas terdapat kelas abstrak GameObject yang berperan sebagai fondasi utama bagi objek-objek game. Kelas ini mendefinisikan method umum seperti render() dan update() yang wajib dimiliki oleh seluruh objek turunan. Dengan menjadikannya abstrak, sistem memaksa setiap kelas turunan untuk mengimplementasikan perilaku rendering dan

pembaruan logika sesuai kebutuhan masing-masing objek, sehingga tercipta konsistensi antarkomponen visual dalam game.

Kelas abstrak BaseButton merupakan turunan dari GameObject dan merepresentasikan konsep umum sebuah tombol. Kelas ini menerapkan enkapsulasi dengan menyimpan properti penting seperti rect, image, dan audio_manager, serta menyediakan method draw() dan is_clicked() untuk menggambar tombol dan mendeteksi interaksi pengguna. Dari kelas ini, diturunkan dua kelas konkret yaitu NormalButton dan LockedButton. Hubungan ini menunjukkan penerapan inheritance, di mana kedua kelas mewarisi struktur dasar tombol namun memiliki perilaku klik yang berbeda. NormalButton dapat merespons klik pengguna, sedangkan LockedButton membatasi interaksi, sehingga mencerminkan penerapan polymorphism dalam sistem tombol.

Komponen AudioManager dihubungkan melalui relasi komposisi dengan tombol dan komponen lain yang membutuhkan suara. Kelas ini bertanggung jawab mengelola audio game, seperti musik latar, suara klik, status aktif audio, dan volume. Dengan memusatkan pengelolaan suara dalam satu kelas, sistem menjadi lebih terorganisir dan mudah dipelihara, sekaligus menghindari duplikasi logika audio di berbagai bagian game.

Untuk pengelolaan kuis, diagram menampilkan kelas QuizManager yang menggunakan struktur data dictionary untuk menyimpan kumpulan soal berdasarkan level. QuizManager memiliki tanggung jawab mengatur alur kuis, mulai dari mengambil soal aktif, berpindah ke soal berikutnya, mereset kuis, hingga memeriksa jawaban pengguna. Kelas ini memiliki relasi “contains” dengan QuizQuestion, yang berarti satu QuizManager dapat memiliki banyak objek QuizQuestion. Setiap QuizQuestion menyimpan pertanyaan dan jawaban serta menyediakan method check_answer() yang menangani logika validasi jawaban, termasuk normalisasi teks agar input pengguna lebih fleksibel.

Pengelolaan level game ditangani oleh LevelManager, yang bertugas mengatur level aktif, membuka level berikutnya, serta menyimpan data level. LevelManager memiliki relasi “manages” terhadap kelas Level, di mana satu LevelManager dapat mengelola banyak Level. Kelas Level sendiri menyimpan informasi penting seperti nomor level, dialog, background, karakter, serta status unlock dan completed. Method seperti unlock(), complete(), dan get_current_dialog() menunjukkan bahwa setiap level memiliki siklus hidup dan alur cerita yang terkontrol dengan baik.

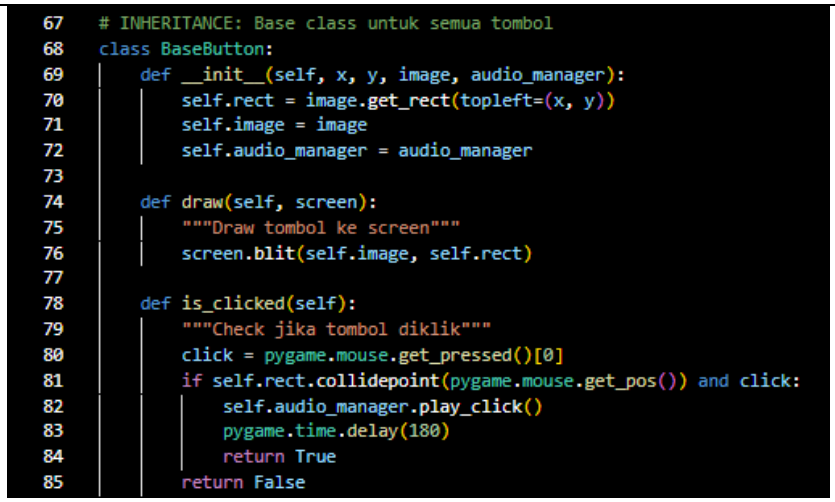
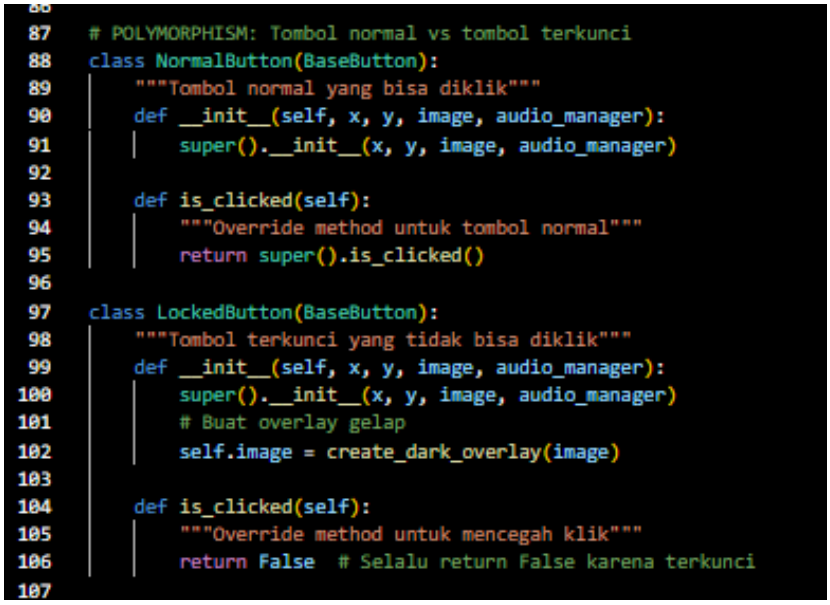
2.4 Desain Sistem

berikut adalah analisis dari sistem

Nomer	Analisis
	import pygame, sys.py > ... 1 import pygame 2 import sys 3 import os 4 import random 5 6

	Pada tahap awal , membuat beberapa modul pendukung, antara lain pygame sebagai pustaka utama untuk pengembangan game, sys untuk pengelolaan eksekusi program, os untuk pengaturan akses file dan asset permainan, serta random untuk mendukung variasi elemen permainan
	 Membuat kelas AudioManager. Kelas ini dirancang dengan menerapkan konsep encapsulation, di mana seluruh data dan status audio disimpan sebagai atribut internal kelas dan hanya dapat diakses atau dimanipulasi melalui method tertentu. Proses dimulai dengan inisialisasi objek, Selanjutnya, sistem menerima input konfigurasi audio melalui inisialisasi mixer Pygame yang dikendalikan oleh

```python
# ENCAPSULATION: Kelas untuk mengelola audio dengan atribut private
class AudioManager:
    def __init__(self):
        self._enabled = False  # Private attribute
        self._background_music = None
        self._click_sound = None
        self._volume = 0.7

    def initialize(self):
        """Initialize audio system"""
        try:
            pygame.mixer.pre_init(frequency=44100, size=-16, channels=2, buffer=512)
            pygame.mixer.init()
            self._enabled = True
            return True
        except pygame.error as e:
            print(f"Audio initialization failed: {e}")
            return False

    def load_sounds(self):
        """Load semua sound"""
        if not self._enabled:
            return

        try:
            # Load background music
            bg_music_path = "background_musik.mp3"
            if os.path.exists(bg_music_path):
                pygame.mixer.music.load(bg_music_path)
                pygame.mixer.music.set_volume(0.5)
                pygame.mixer.music.play(-1)
                print("Background music loaded successfully")

            # Load sound efek klik
            click_sound_path = "click.wav.mp3"
            if os.path.exists(click_sound_path):
                self._click_sound = pygame.mixer.Sound(click_sound_path)
                self._click_sound.set_volume(self._volume)
                print("Click sound loaded successfully")
        except Exception as e:
            print(f"Error loading sounds: {e}")

    def play_click(self):
        """Play click sound"""
        if self._click_sound and self._enabled:
            try:
                self._click_sound.play()
            except Exception as e:
                print(f"Error playing sound: {e}")

    @property
    def enabled(self):
        """Getter untuk enabled status"""
        return self._enabled

    @property
    def click_sound(self):
        """Getter untuk click sound"""
        return self._click_sound
```

	method initialize(). Proses pemuatan aset suara dari file eksternal juga dilakukan melalui method load_sounds(), sehingga pengelolaan audio tetap terpusat dalam satu kelas tanpa bergantung pada komponen lain. Saat pengguna melakukan interaksi dalam permainan, sistem merespon dengan memutar efek suara melalui pemanggilan method play_click() sebagai bentuk output audio.
	  Kode ini mengimplementasikan sistem tombol yang elegan menggunakan konsep Object-Oriented Programming (OOP), khususnya inheritance (pewarisan) dan polymorphism (banyak bentuk). Sistem ini dibangun dengan tiga kelas yang saling terkait: BaseButton sebagai kelas induk, serta NormalButton dan LockedButton sebagai kelas turunan yang menunjukkan polimorfisme. BaseButton berfungsi sebagai blueprint dasar untuk semua jenis tombol dalam game. Konstruktor __init__ menerima empat parameter: x dan y untuk menentukan posisi tombol, image untuk representasi visual, dan audio_manager

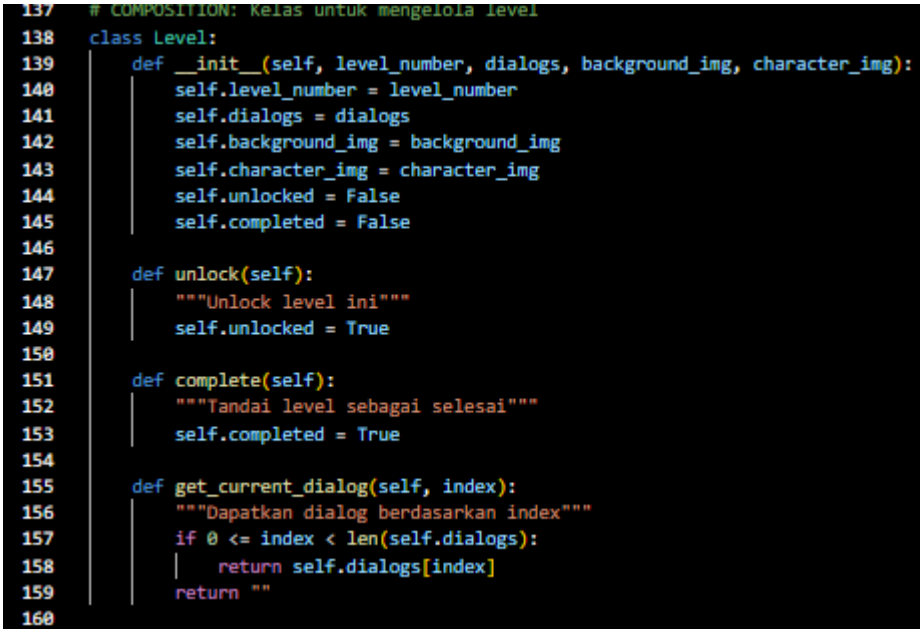
```
67  # INHERITANCE: Base class untuk semua tombol
68  class BaseButton:
69      def __init__(self, x, y, image, audio_manager):
70          self.rect = image.get_rect(topleft=(x, y))
71          self.image = image
72          self.audio_manager = audio_manager
73
74      def draw(self, screen):
75          """Draw tombol ke screen"""
76          screen.blit(self.image, self.rect)
77
78      def is_clicked(self):
79          """Check jika tombol diklik"""
80          click = pygame.mouse.get_pressed()[0]
81          if self.rect.collidepoint(pygame.mouse.get_pos()) and click:
82              self.audio_manager.play_click()
83              pygame.time.delay(180)
84              return True
85          return False
```

```
87  # POLYMORPHISM: Tombol normal vs tombol terkunci
88  class NormalButton(BaseButton):
89      """Tombol normal yang bisa diklik"""
90      def __init__(self, x, y, image, audio_manager):
91          super().__init__(x, y, image, audio_manager)
92
93      def is_clicked(self):
94          """Override method untuk tombol normal"""
95          return super().is_clicked()
96
97  class LockedButton(BaseButton):
98      """Tombol terkunci yang tidak bisa diklik"""
99      def __init__(self, x, y, image, audio_manager):
100         super().__init__(x, y, image, audio_manager)
101         # Buat overlay gelap
102         self.image = create_dark_overlay(image)
103
104     def is_clicked(self):
105         """Override method untuk mencegah klik"""
106         return False  # Selalu return False karena terkunci
107
```

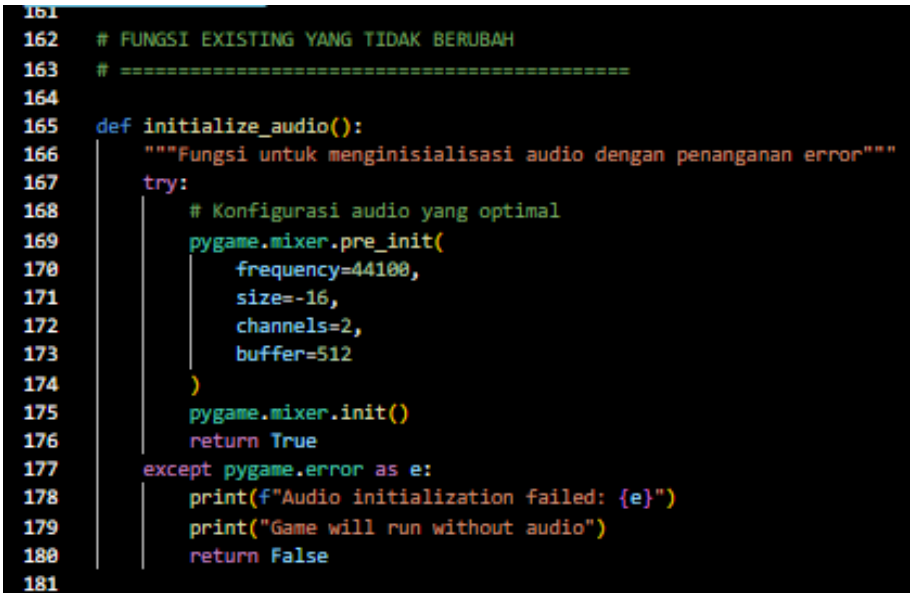
	untuk menangani feedback suara. Dalam konstruktor, properti rect dibuat menggunakan metode get_rect() dari gambar, yang menghasilkan area persegi panjang untuk deteksi tabrakan (collision detection). Metode draw() bertanggung jawab untuk merender tombol ke layar menggunakan fungsi blit() Pygame, sementara metode is_clicked() mengimplementasikan logika deteksi interaksi pengguna dengan memeriksa apakah tombol mouse kiri ditekan (pygame.mouse.get_pressed()[0]) dan apakah posisi mouse berada dalam batas rect tombol (collidepoint()). Ketika kedua kondisi terpenuhi, sistem akan memainkan suara klik melalui audio_manager.play_click(), memberikan jeda 180 milidetik untuk efek responsif, dan mengembalikan True untuk menandakan klik yang berhasil. NormalButton mewarisi seluruh fungsionalitas dari BaseButton tanpa modifikasi signifikan. Konstruktor kelas ini hanya memanggil konstruktor induk menggunakan super().__init__(), yang meneruskan semua parameter ke kelas BaseButton. Demikian pula, metode is_clicked() hanya memanggil versi induknya dengan super().is_clicked(). Meskipun tampak redundan, pendekatan ini memiliki nilai dokumentasi yang jelas—kelas ini secara eksplisit mendeklarasikan dirinya sebagai tombol normal yang berperilaku standar. Dalam praktiknya, kelas ini bisa saja tidak mengoverride metode is_clicked() sama sekali karena perilaku default dari BaseButton sudah sesuai, tetapi override ini memungkinkan fleksibilitas untuk modifikasi di masa depan jika diperlukan penyesuaian khusus untuk tombol normal. LockedButton mendemonstrasikan polimorfisme sejati dengan mengubah perilaku dan tampilan yang diwarisi dari BaseButton. Dalam konstruktor, setelah memanggil konstruktor induk, kelas ini mengganti properti image dengan hasil dari fungsi create_dark_overlay(image), yang menambahkan lapisan gelap transparan pada gambar asli untuk memberikan isyarat visual bahwa tombol tidak aktif. Modifikasi yang lebih signifikan terjadi pada metode is_clicked()—berbeda dengan NormalButton yang mewarisi logika deteksi klik, LockedButton mengoverride metode ini untuk selalu mengembalikan False, tanpa memeriksa posisi mouse atau status klik. Ini menciptakan tombol yang terlihat tetapi tidak dapat berinteraksi, konsep yang sempurna untuk level-level yang belum terbuka dalam game. Ketika is_clicked() dipanggil pada objek NormalButton, sistem akan menjalankan logika deteksi klik dari BaseButton. Sebaliknya, jika objek tersebut adalah LockedButton, sistem akan langsung mengembalikan False tanpa melakukan deteksi apa pun. Pendekatan ini menghilangkan kebutuhan untuk pengecekan kondisi seperti if unlocked_level >= 2 setiap kali tombol diproses, menghasilkan kode yang lebih bersih dan efisien.

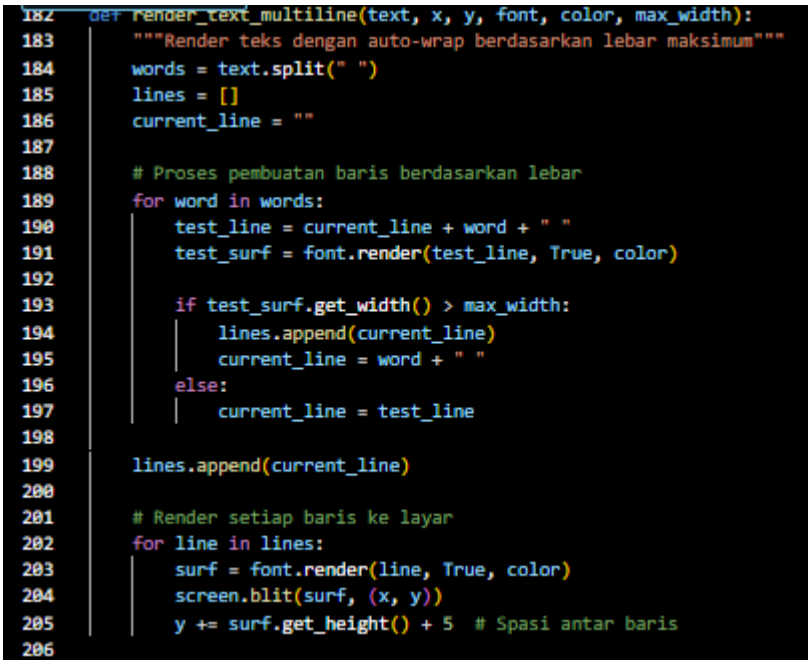
```python
# ENCAPSULATION: Kelas untuk mengelola soal quiz
class QuizQuestion:
    def __init__(self, question, answer):
        self._question = question  # Private attribute
        self._answer = answer      # Private attribute

    @property
    def question(self):
        """Getter untuk soal"""
        return self._question

    @property
    def answer(self):
        """Getter untuk jawaban"""
        return self._answer

    def normalize_text(self, text):
        """Normalisasi teks untuk perbandingan"""
        t = text.replace(" ", "").replace("\n", "").replace("\t", "").lower()
        for c in [';', ':', ',', '"', "'"]:
            t = t.replace(c, "")
        return t

    def check_answer(self, user_answer):
        """Cek jawaban user"""
        user_normalized = self.normalize_text(user_answer)
        correct_normalized = self.normalize_text(self._answer)
        return user_normalized == correct_normalized

```

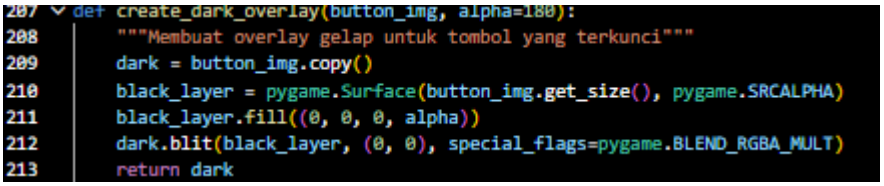
Kelas QuizQuestion disini menerapkan konsep encapsulation (enkapsulasi). Pada konstruktor, terlihat penggunaan underscore prefix (_) pada nama atribut—_question dan _answer. Dalam konvensi Python, ini menandakan bahwa atribut tersebut bersifat "protected" atau semi-private, meskipun Python tidak benar-benar menerapkan private attributes seperti bahasa Java atau C++. Ini adalah sinyal kepada programmer lain bahwa atribut ini seharusnya tidak diakses langsung dari luar kelas, melainkan melalui getter yang disediakan. Enkapsulasi ini melindungi data soal dan jawaban dari modifikasi tidak sah yang bisa merusak integritas sistem quiz. Kelas ini menggunakan property decorator (@property) untuk membuat getter methods yang elegan. Property decorator mengubah metode menjadi atribut yang dapat diakses seperti atribut biasa, tetapi sebenarnya menjalankan kode metode di balik layar. Dengan @property, kita bisa mengakses soal dengan quiz.question (bukan quiz.question()), yang memberikan interface yang lebih bersih dan intuitif. Metode normalize_text() adalah jantung dari sistem penilaian otomatis yang toleran terhadap variasi format. Metode ini menerapkan serangkaian transformasi untuk menstandarkan teks sebelum perbandingan: pertama, menghapus semua spasi, newline, dan tab; kedua, mengubah semua karakter menjadi huruf kecil (case-insensitive); ketiga, menghapus karakter tanda baca umum seperti titik koma, titik dua, koma, dan tanda kutip. Pendekatan ini memungkinkan sistem untuk menganggap "print('hello')" dan "print ('hello')" sebagai jawaban yang sama, yang sangat penting dalam konteks pembelajaran pemrograman di mana format spasi dan kapitalisasi sering kali bervariasi di antara peserta didik. Namun, perlu diperhatikan bahwa algoritma ini mungkin terlalu agresif—dengan menghapus semua spasi, kode seperti "a = b + c" dan "a=b+c" akan dianggap sama, padahal dalam Python, kedua bentuk tersebut memang valid.

	Ini sebenarnya tepat untuk konteks quiz pemula yang berfokus pada logika daripada sintaksis detail. Metode check_answer() memanfaatkan enkapsulasi dengan cara yang cerdas. Meskipun _answer adalah atribut private, metode ini dapat mengaksesnya secara langsung karena berada dalam kelas yang sama
	 Kelas Level disini menerapan konsep composition (komposisi). Konstruktor kelas Level menunjukkan dengan jelas prinsip komposisi: sebuah level terdiri dari beberapa komponen berbeda yang masing-masing memiliki peran spesifik. Parameter level_number menyimpan identifikasi numerik level, dialogs berisi daftar dialog naratif (biasanya dalam bentuk list string), background_img merupakan objek gambar untuk latar belakang visual, dan character_img adalah objek gambar untuk karakter yang muncul di level tersebut. Dengan mengelompokkan semua data terkait level ke dalam satu objek yang kohesif, desain ini menciptakan unit yang mandiri dan dapat dikelola secara independen. Keunggulan utama dari pendekatan komposisi ini adalah fleksibilitas—setiap komponen dapat diubah atau diganti tanpa mempengaruhi komponen lain atau struktur keseluruhan level. Status unlocked dan completed yang diinisialisasi sebagai False merupakan state management yang penting dalam sistem game. Inisialisasi default ini memastikan bahwa level baru selalu dimulai dalam keadaan terkunci dan belum selesai, yang sesuai dengan alur game progresif di mana pemain harus membuka level secara berurutan. Desain ini memisahkan dengan jelas antara data konfigurasi (yang diberikan saat pembuatan objek) dan data state (yang berubah selama gameplay), sebuah prinsip desain yang baik untuk sistem yang dinamis. Kedua metode sederhana ini menyediakan interface yang bersih untuk mengubah state level. Meskipun kelihatannya sepele, desain ini memiliki beberapa keunggulan penting. Pertama, encapsulation—daripada mengizinkan kode luar untuk mengubah

```
137  # COMPOSITION: Kelas untuk mengelola level
138  class Level:
139      def __init__(self, level_number, dialogs, background_img, character_img):
140          self.level_number = level_number
141          self.dialogs = dialogs
142          self.background_img = background_img
143          self.character_img = character_img
144          self.unlocked = False
145          self.completed = False
146
147      def unlock(self):
148          """Unlock level ini"""
149          self.unlocked = True
150
151      def complete(self):
152          """Tandai level sebagai selesai"""
153          self.completed = True
154
155      def get_current_dialog(self, index):
156          """Dapatkan dialog berdasarkan index"""
157          if 0 <= index < len(self.dialogs):
158              return self.dialogs[index]
159          return ""
160
```

	unlocked dan completed secara langsung, perubahan hanya boleh dilakukan melalui metode yang ditentukan. Kedua, consistency—jika di masa depan diperlukan logika tambahan saat membuka atau menyelesaikan level (misalnya, memainkan efek suara, mencatat waktu penyelesaian, atau memberi reward), perubahan hanya perlu dilakukan dalam metode ini tanpa mempengaruhi kode yang memanggilnya. Ketiga, readability—level.unlock() jauh lebih deskriptif daripada level.unlocked = True, terutama untuk developer lain yang membaca kode. Metode get_current_dialog() adalah contoh defensive programming yang baik.
	 ``` 161 162 # FUNGSI EXISTING YANG TIDAK BERUBAH 163 # == 164 165 def initialize_audio(): 166 """Fungsi untuk menginisialisasi audio dengan penanganan error""" 167 try: 168 # Konfigurasi audio yang optimal 169 pygame.mixer.pre_init(170 frequency=44100, 171 size=-16, 172 channels=2, 173 buffer=512 174) 175 pygame.mixer.init() 176 return True 177 except pygame.error as e: 178 print(f"Audio initialization failed: {e}") 179 print("Game will run without audio") 180 return False 181 ``` Fungsi initialize_audio(). Konfigurasi awal (pre_init) menggunakan parameter-parameter yang telah dipilih secara cermat untuk perform audio yang optimal dalam konteks game edukasi. Parameter frequency=44100 menetapkan sample rate 44.1 kHz, yang merupakan standar CD quality dan memberikan keseimbangan yang baik antara kualitas audio dan penggunaan resource. Parameter size=-16 mengatur depth audio menjadi 16-bit dengan signed format (tanda negatif menandakan menggunakan format default sistem), yang memberikan dynamic range yang cukup untuk efek suara game tanpa konsumsi memori berlebihan. Parameter channels=2 mengonfigurasi output stereo, penting untuk efek spasial dan pengalaman audio yang imersif. Terakhir, buffer=512 menetapkan buffer size yang relatif kecil—nilai yang lebih kecil mengurangi latency (penundaan antara aksi dan suara) tetapi membutuhkan proses yang lebih stabil; nilai 512 merupakan kompromi yang baik untuk kebanyakan sistem modern. Penting untuk dicatat bahwa pre_init() harus dipanggil sebelum pygame.init() atau pygame.mixer.init() untuk efektif. Ini adalah detail teknis yang menunjukkan pemahaman mendalam tentang API Pygame. Setelah konfigurasi pre-init,

	pygame.mixer.init() melakukan inisialisasi aktual dari sistem audio berdasarkan parameter yang telah ditetapkan. Struktur try-except dalam fungsi ini menunjukkan penerapan defensive programming yang matang. Blok try berisi operasi yang berisiko gagal—inisialisasi sistem audio, yang bergantung pada hardware dan driver yang mungkin tidak tersedia atau bermasalah di beberapa sistem. Ketika exception pygame.error terjadi (exception spesifik untuk kesalahan Pygame), blok except menangkapnya dengan elegan. Penanganan error dilakukan pada beberapa level: pertama, logging informatif dengan print(f"Audio initialization failed: {e}") yang mencatat error detail untuk debugging developer. Kedua, user-friendly messaging dengan print("Game will run without audio") yang menginformasikan pengguna tentang konsekuensinya tanpa jargon teknis. Ketiga, clean recovery dengan mengembalikan False yang memungkinkan kode pemanggil untuk melanjutkan tanpa audio, daripada menghentikan eksekusi game.
	 Fungsi render_text_multiline(). Fungsi ini menerima enam parameter yang masing-masing memiliki peran spesifik: text berisi string yang akan ditampilkan, x dan y menentukan posisi awal teks di layar, font menyediakan informasi typography, color mengatur tampilan visual, dan max_width menjadi batas tak terlihat yang menjadi garda depan dalam pertempuran antara konten dan ruang. Proses dimulai dengan pembelahan teks menjadi unit-unit terkecil melalui text.split(" "), sebuah pendekatan sederhana namun efektif yang mengasumsikan spasi sebagai pemisah kata. Setelah tokenisasi, algoritma masuk ke fase inti dengan inisialisasi dua struktur data: lines sebagai wadah untuk baris-baris yang telah terbentuk, dan current_line

	sebagai kanvas sementara untuk baris yang sedang ditulis. Di sinilah kejeniusan algoritma mulai terlihat melalui loop yang memproses setiap kata. Untuk setiap kata yang ditambahkan ke current_line, algoritma tidak langsung berasumsi bahwa kombinasi tersebut akan muat, melainkan membuat surface sementara melalui font.render(test_line, True, color) dan secara teliti mengukur lebarnya dengan test_surf.get_width(). Pendekatan empiris ini—mengukur hasil aktual rendering daripada mengandalkan estimasi—menunjukkan pemahaman yang dalam tentang sifat font rendering yang tidak selalu linear. Ketika pengukuran menunjukkan bahwa lebar telah melebihi max_width, algoritma dengan bijak memutuskan untuk menyimpan current_line ke dalam lines dan memulai baris baru dengan kata yang memicu pelanggaran batas tersebut. Proses ini berlanjut sampai semua kata terproses, dengan perhatian khusus pada kata-kata terakhir yang ditambahkan ke dalam lines setelah loop selesai, sebuah sentuhan akhir yang mencegah kehilangan konten. Setelah fase pembentukan baris selesai, fungsi beralih ke tahap rendering yang sesungguhnya. Setiap baris dalam lines di-render secara individual dan diposisikan pada koordinat y yang terus meningkat. Di sini muncul elemen estetika yang halus namun penting: y += surf.get_height() + 5.
	 Fungsi create_dark_overlay(). Proses dimulai dengan dark = button_img.copy(), membuat salinan dari gambar asli, fungsi ini memastikan bahwa gambar original tetap utuh. Parameter alpha=180 dengan nilai default 180 menunjukkan pemilihan yang matang: alpha 255 akan membuat tombol hampir hitam total, sementara alpha yang terlalu rendah akan membuat efek terlalu halus. Nilai 180 adalah sweet spot—cukup gelap untuk terlihat terkunci, cukup transparan untuk memperlihatkan bentuk asli tombol. Bagian paling menarik dari fungsi ini terletak pada pembuatan black_layer. Dengan pygame.Surface(button_img.get_size(), pygame.SRCALPHA), fungsi menciptakan sebuah surface baru dengan ukuran yang sama persis dengan tombol, tetapi dengan channel alpha yang diaktifkan (SRCALPHA). Kemudian black_layer.fill((0, 0, 0, alpha)) mengisi layer ini dengan warna hitam murni (0,0,0) dengan transparansi sesuai parameter alpha. Operasi blending yang sebenarnya terjadi pada baris dark.blit(black_layer, (0, 0), special_flags=pygame.BLEND_RGBA_MULT). Flagpygame.BLEND_RGBA_MULT adalah efek visual ini. Dengan tumpukan berwarna hitam, operasi perkalian warna (multiply blend) dilakukan. Dalam

```
207 def create_dark_overlay(button_img, alpha=180):
208     """Membuat overlay gelap untuk tombol yang terkunci"""
209     dark = button_img.copy()
210     black_layer = pygame.Surface(button_img.get_size(), pygame.SRCALPHA)
211     black_layer.fill((0, 0, 0, alpha))
212     dark.blit(black_layer, (0, 0), special_flags=pygame.BLEND_RGBA_MULT)
213     return dark
```

	grafik komputer, blend mode multiply bekerja dengan mengalikan nilai channel warna dari layer bawah dengan layer atas. Karena layer atas berwarna hitam (0,0,0), hasil perkaliannya akan selalu mendekati nol, sehingga menggelapkan gambar. Namun, karena layer atas memiliki alpha 180 (bukan 255), efeknya tidak sepenuhnya hitam—beberapa cahaya asli tetap terlihat. Hasilnya adalah tombol yang tampak "redup" atau "dalam bayangan".
	 Fungsi image_btn(). Fungsi menerima tiga parameter dasar: image (gambar tombol), x dan y (koordinat posisi). Langkah pertama, rect = image.get_rect(topleft=(x, y)), menciptakan rectangle Pygame dari gambar, yang kemudian digunakan untuk dua tujuan: rendering visual dan deteksi tabrakan. Segera setelah rectangle dibuat, fungsi langsung merender gambar ke layar dengan screen.blit(image, rect). Bagian inti dari fungsi ini adalah logika deteksi klik yang terkonsentrasi dalam satu blok kondisional: if rect.collidepoint(pygame.mouse.get_pos()) and click:. Kondisi ini memeriksa dua hal sekaligus: pertama, apakah posisi mouse berada dalam batas rectangle tombol (menggunakan collidepoint()), dan kedua, apakah tombol mouse kiri sedang ditekan (pygame.mouse.get_pressed()[0]). Pendekatan ini efisien dan langsung—semua logika terkandung dalam satu baris. pygame.mouse.get_pressed() akan mengembalikan state saat ini, bukan event, Ketika klik terdeteksi, fungsi memberikan feedback multi-saluran. Pertama, ia memeriksa apakah sound_click tersedia (bukan None). Di sini muncul global kedua: sound_click diasumsikan sebagai variabel global. Jika suara tersedia, fungsi mencoba memutarnya dengan blok try-except yang menangani kemungkinan error. Setelah feedback audio (atau upaya untuk memberikan feedback audio), fungsi memberikan feedback melalui pygame.time.delay(180). Delay 180 milidetik ini adalah sentuhan untuk memberikan sensasi "tombol ditekan", cukup singkat untuk tidak mengganggu alur gameplay. Akhirnya, fungsi mengembalikan True untuk menandakan klik yang berhasil.

```
214
215 def image_btn(image, x, y):
216     """Membuat tombol dari gambar dengan feedback klik"""
217     rect = image.get_rect(topleft=(x, y))
218     screen.blit(image, rect)
219     click = pygame.mouse.get_pressed()[0]
220
221     # Cek jika tombol diklik
222     if rect.collidepoint(pygame.mouse.get_pos()) and click:
223         # Mainkan sound efek jika tersedia
224         if sound_click is not None:
225             try:
226                 sound_click.play()
227             except Exception as e:
228                 print(f"Error playing sound: {e}")
229         pygame.time.delay(180)  # Delay kecil untuk feedback
230         return True
231     return False
232
```

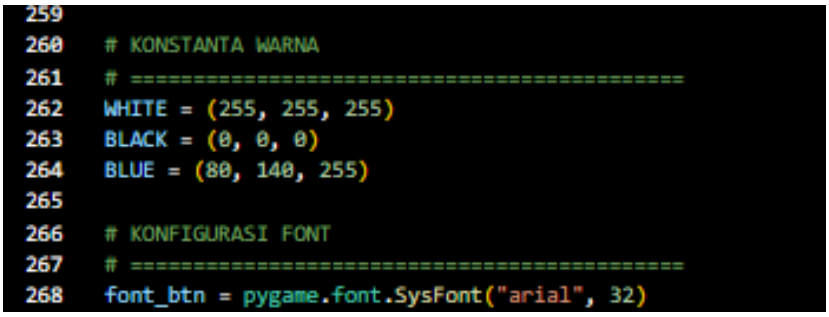
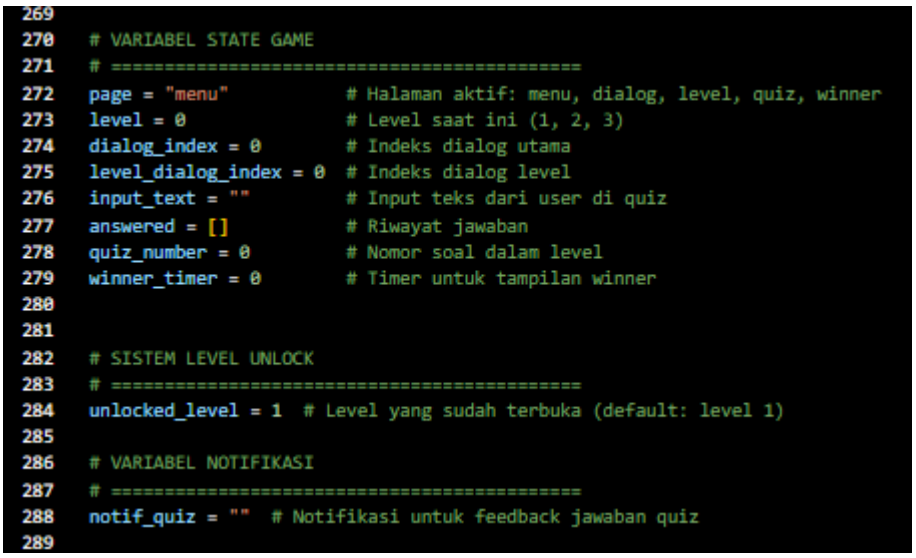
```
# INISIALISASI
# ========================================

# Inisialisasi Pygame
pygame.init()

# Buat instance AudioManager (OOP)
audio_manager = AudioManager()
audio_enabled = audio_manager.initialize()

# Load sounds menggunakan AudioManager
if audio_enabled:
    audio_manager.load_sounds()
    sound_click = audio_manager.click_sound
else:
    print("Audio disabled. Game will run without sound.")
    sound_click = None
```

Dimulai dengan pygame.init(), AudioManager() diinstansiasi sebagai objek OOP yang lengkap—sebuah entitas yang meng-encapsulate state dan behavior terkait audio. Ini adalah sistem baru, dirancang dengan prinsip-prinsip OOP modern: encapsulation (dengan atribut _enabled, _volume yang protected), abstraction (metode seperti initialize(), load_sounds()), dan potential untuk polymorphism dan inheritance di masa depan.

Ketika audio_manager.initialize() dipanggil, kita menyaksikan konsep OOP dalam aksi. Menggunakan pygame.mixer.pre_init() dengan parameter-parameter yang dioptimalkan, menunjukkan bahwa desainer sistem telah mempertimbangkan aspek teknis yang mendalam tentang bagaimana audio harus diatur untuk pengalaman game yang optimal.

```


# KONFIGURASI WINDOW
# ========================================
WIDTH, HEIGHT = 800, 500
screen = pygame.display.set_mode((WIDTH, HEIGHT))
pygame.display.set_caption("PayGame Edukasi")

```

kode ini untuk mengonfigurasi jendela (window) utama aplikasi game berbasis Pygame. Pada bagian awal, komentar # KONFIGURASI WINDOW digunakan sebagai penanda bahwa kode di bawahnya berkaitan dengan pengaturan tampilan, sehingga meningkatkan keterbacaan dan struktur program. Variabel WIDTH dan HEIGHT didefinisikan dengan nilai 800 dan 500, yang merepresentasikan lebar dan tinggi jendela game dalam satuan piksel. Pendefinisian ukuran ini bersifat eksplisit agar mudah diubah dan digunakan kembali pada bagian lain kode yang membutuhkan informasi dimensi layar.

	Selanjutnya, fungsi pygame.display.set_mode((WIDTH, HEIGHT)) digunakan untuk membuat surface utama bernama screen, yang menjadi media utama untuk menampilkan seluruh elemen visual seperti background, karakter, tombol, dan teks. Surface ini sangat penting karena hampir semua proses rendering di Pygame dilakukan melalui objek screen. Terakhir, pygame.display.set_caption("PayGame Edukasi") digunakan untuk memberi judul pada jendela aplikasi, sehingga pengguna dapat mengenali nama game saat dijalankan.
	 Kode ini bagian dari awal tampilan game kode mendefinisikan konstanta warna menggunakan format RGB, yaitu WHITE untuk warna putih, BLACK untuk warna hitam, dan BLUE untuk warna biru dengan intensitas sedang. Pendefinisian warna sebagai konstanta bertujuan untuk meningkatkan keterbacaan kode, menjaga konsistensi penggunaan warna di seluruh program, serta memudahkan pengelolaan dan perubahan warna jika diperlukan. Pada bagian selanjutnya, kode mengatur konfigurasi font dengan membuat objek font menggunakan pygame.font.SysFont dengan jenis huruf Arial dan ukuran 32 piksel. Font ini umumnya digunakan untuk menampilkan teks pada elemen antarmuka seperti tombol atau menu.
	

```
259
260   # KONSTANTA WARNA
261   # ========================================
262   WHITE = (255, 255, 255)
263   BLACK = (0, 0, 0)
264   BLUE = (80, 140, 255)
265
266   # KONFIGURASI FONT
267   # ========================================
268   font_btn = pygame.font.SysFont("arial", 32)
```

```
269
270   # VARIABEL STATE GAME
271   # ========================================
272   page = "menu"           # Halaman aktif: menu, dialog, level, quiz, winner
273   level = 0               # Level saat ini (1, 2, 3)
274   dialog_index = 0        # Indeks dialog utama
275   level_dialog_index = 0  # Indeks dialog level
276   input_text = ""         # Input teks dari user di quiz
277   answered = []           # Riwayat jawaban
278   quiz_number = 0         # Nomor soal dalam level
279   winner_timer = 0        # Timer untuk tampilan winner
280
281
282   # SISTEM LEVEL UNLOCK
283   # ========================================
284   unlocked_level = 1  # Level yang sudah terbuka (default: level 1)
285
286   # VARIABEL NOTIFIKASI
287   # ========================================
288   notif_quiz = ""  # Notifikasi untuk feedback jawaban quiz
289
```

	Kode ini bagian dari pengaturan awal dan pengendali alur permainan dalam sebuah game berbasis Pygame. Proses kode dimulai dengan pembuatan objek font melalui font_btn, yang akan digunakan untuk merender seluruh teks pada game seperti tombol, dialog, soal kuis, dan notifikasi, sehingga tampilan teks menjadi konsisten. Selanjutnya, game menggunakan sekumpulan variabel state untuk menentukan kondisi permainan yang sedang berjalan. Variabel page berfungsi sebagai penentu halaman aktif sehingga program dapat memutuskan proses apa yang harus dijalankan, apakah menampilkan menu utama, dialog cerita, level permainan, kuis, atau layar kemenangan. Variabel level menyimpan informasi level yang sedang dimainkan, sementara dialog_index dan level_dialog_index mengatur urutan dialog agar ditampilkan secara bertahap sesuai alur cerita. Ketika pemain memasuki sesi kuis, proses input dan evaluasi jawaban dikendalikan oleh input_text, answered, dan quiz_number, di mana jawaban yang diketik pemain disimpan, dicatat, lalu diproses berdasarkan nomor soal yang sedang aktif. Jika pemain berhasil menyelesaikan tantangan dalam satu level, winner_timer akan mulai menghitung durasi tampilan layar kemenangan sebelum game berpindah ke proses selanjutnya. Progres permainan dikontrol melalui variabel unlocked_level, yang memastikan hanya level tertentu saja yang dapat diakses sesuai dengan pencapaian pemain. Terakhir, variabel notif_quiz digunakan sebagai media umpan balik untuk menampilkan pesan benar atau salah terhadap jawaban kuis, sehingga interaksi pemain dengan sistem menjadi lebih informatif.

```
# DATA DIALOG CERITA
# ========================================
# Dialog utama (pembuka game)
dialogs = [
    "Selamat Datang dalam Arena Perang",
    "Banyak Musuh yang Sedang Menunggu Anda",
    "Selesaikan Misi dan Kalahkan Musuh",
    "Selamat Berperang!"
]

# Dialog per level
dialog_level1 = [
    "Selamat datang di Level 1!",
    "Perkenalkan Saya Jelly",
    "Musuh Anda di Level ini",
    "Mari Kalahkan Saya!!",
    "Semoga Kemenangan Berpihak Pada Anda"
]

dialog_level2 = [
    "Selamat datang di Level 2!",
    "Perkenalkan Saya Mummis",
    "Musuh Anda di Level ini",
    "Mari Kalahkan Saya!!",
    "Semoga Kemenangan Berpihak Pada Anda"
]

dialog_level3 = [
    "Selamat datang di Level 3!",
    "Perkenalkan Saya Ablis",
    "Musuh Anda di Level ini",
    "Mari Kalahkan Saya!!",
    "Semoga Kemenangan Berpihak Pada Anda"
]

```

Kode ini menyimpan data dialog cerita yang akan ditampilkan kepada pemain sepanjang permainan. Secara proses, saat game dimulai, list dialogs digunakan sebagai dialog pembuka untuk memperkenalkan latar cerita dan tujuan permainan sebelum pemain masuk ke tahap level. Dialog ini ditampilkan secara berurutan menggunakan indeks dialog yang dikontrol oleh variabel state, sehingga teks muncul satu per satu sesuai alur cerita.

Selanjutnya, setiap level memiliki dialog khusus yang disimpan dalam list terpisah, yaitu dialog_level1, dialog_level2, dan dialog_level3. Data ini digunakan ketika pemain memasuki level tertentu, di mana sistem akan membaca dialog sesuai level yang aktif dan menampilkannya secara bertahap sebelum pertempuran dimulai. Pendekatan ini memungkinkan alur cerita yang terstruktur dan mudah dikembangkan, karena penambahan level baru cukup dengan menambahkan list dialog baru tanpa mengubah logika utama program.

```python
# DATA QUIZ DENGAN OOP (MODIFIKASI KECIL)
# ----------------------------------------
# QuizQuestion untuk setiap soal (OOP)
quiz_questions = {
    1: [
        QuizQuestion(
            "Budi ingin menyapa semua temannya lewat komputer. "
            "Tulislah program yang menampilkan tulisan: Halo, Budi!",
            'print("Halo, Budi!")'
        ),
        QuizQuestion(
            "Di sekolah, bel pagi berbunyi 3 kali untuk memanggil para siswa. "
            "Buat program yang menampilkan tulisan 'Selamat pagi semuanya!' sebanyak 3 kali.",
            'for i in range(3): print("Selamat pagi semuanya!")'
        ),
        QuizQuestion(
            "Doni punya 3 permen, lalu ibunya memberi 5 permen lagi."
            "Buat program untuk menghitung total permen Doni.",
            'hasil = 3 + 5\nprint(hasil)'
        ),
        QuizQuestion(
            "Di taman sekolah ada ubin berbentuk persegi dengan panjang sisi 4 cm."
            "Buat program untuk menghitung luas ubin persegi tersebut.",
            'sisi = 4 luas = sisi * sisi print(luas)'
        ),
        QuizQuestion(
            "Robot mini mempunyai tenaga sebesar angka 4, dan ketika diaktifkan, tenaganya menjadi dua kali lipat."
            "Buat program untuk menampilkan kekuatan robot setelah digandakan.",
            'angka = 4 print(angka * 2)'
        )
    ],
    2: [
        QuizQuestion(
            "Diberikan list angka: [2, 4, 6, 8]."
            "Buat program untuk menjumlahkan semua angkanya.",
            'angka = [2,4,6,8] total = sum(angka) print(total)'
        ),
        QuizQuestion(
            "Jika jumlah = 4, tampilkan kata 'Belajar' sebanyak 4 kali.",
            'jumlah = 4 for i in range(jumlah): print("Belajar")'
        ),
        QuizQuestion(
            "Ubah variabel warna 'merah' menjadi 'biru' lalu tampilkan",
            'warna = "merah" warna = "biru" print(warna)'
        ),
        QuizQuestion(
            "Hitung berapa huruf 'a' dalam kata 'cerita'",
            'print("cerita".count("a"))'
        )
    ],
    3: [
        QuizQuestion(
            "Budi ingin memperkenalkan dirinya kepada teman-teman di kelas."
            "Buatlah program Python sederhana yang menampilkan perkenalan Budi di layar."
            "'Perkenalkan aku Budi,umur aku 5 thn'",
            'nama = "Budi" umur = 5 print(f"Perkenalkan aku {nama},umur aku {umur} thn")'
        )
    ]
}
```

Kode ini ebagai penyimpan dan pengelola data kuis berbasis Object-Oriented Programming (OOP) dalam permainan. Secara proses, data soal kuis disusun dalam sebuah dictionary bernama quiz_questions, di mana setiap key (1, 2, dan 3) merepresentasikan level permainan. Setiap level berisi list objek QuizQuestion, yang masing-masing menyimpan dua komponen utama, yaitu teks soal dan contoh jawaban berupa kode Python. Ketika pemain memasuki mode kuis pada suatu level, sistem akan membaca level yang sedang aktif, lalu mengambil daftar soal yang sesuai dari dictionary ini.

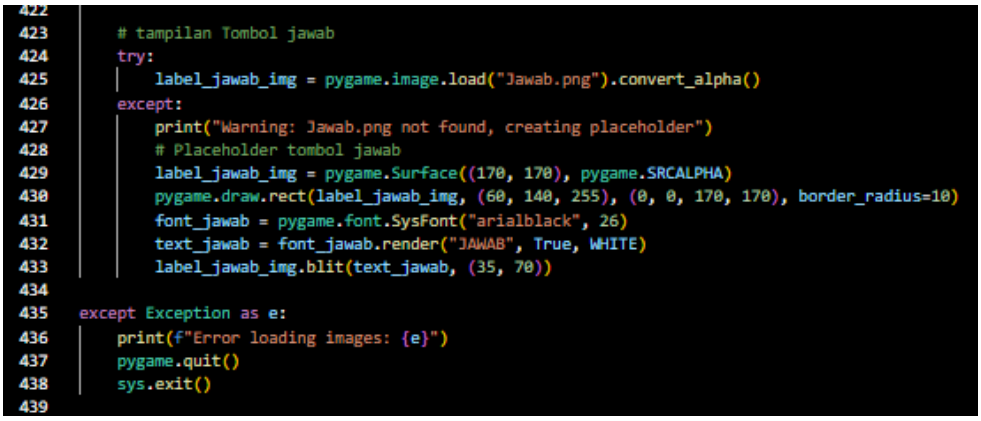
Dalam alur permainan, variabel seperti quiz_number digunakan untuk menentukan soal ke berapa yang sedang ditampilkan. Soal diambil satu per satu dari list QuizQuestion, kemudian teks pertanyaan ditampilkan ke layar, sementara pemain diminta menuliskan jawaban melalui input. Jawaban yang dimasukkan pemain akan dibandingkan atau divalidasi berdasarkan logika yang telah ditentukan (misalnya mencocokkan output atau struktur kode). Jika ingin menambah soal atau level baru, pengembang cukup menambahkan objek QuizQuestion ke dalam dictionary tanpa mengubah alur logika utama game.

```
# Buat dictionary quiz lama untuk kompatibilitas
quiz = {}
for level_num, questions in quiz_questions.items():
    quiz[level_num] = []
    for q in questions:
        quiz[level_num].append((q.question, q.answer))

```

Kode ini mengonversi struktur data kuis berbasis OOP menjadi format lama agar tetap kompatibel dengan bagian kode lain yang masih menggunakan struktur non-OOP. Secara proses, program membuat dictionary baru bernama quiz yang akan menampung data kuis dalam bentuk sederhana. Loop pertama membaca setiap pasangan level dan daftar soal dari quiz_questions, di mana level_num merepresentasikan level permainan dan questions berisi objek-objek QuizQuestion. Untuk setiap level, program menyiapkan list kosong sebagai penampung soal. Selanjutnya, loop kedua menelusuri setiap objek soal q, lalu mengambil atribut question dan answer dari objek tersebut dan menyimpannya sebagai tuple (pertanyaan, jawaban) ke dalam list level yang sesuai.

```

# ASSET GAMBAR
# ========================================
try:
    # Background utama
    bg_menu = pygame.transform.scale(pygame.image.load("baground utama.png"), (WIDTH, HEIGHT))
    bg_dialog = pygame.transform.scale(pygame.image.load("bagorund dialog.png"), (WIDTH, HEIGHT))
    bg_level = pygame.transform.scale(pygame.image.load("Baground level.png"), (WIDTH, HEIGHT))

    # Background dialog per level
    bg_dialoglevel1 = pygame.transform.scale(pygame.image.load("bg_level_1.jpg"), (WIDTH, HEIGHT))
    bg_dialoglevel2 = pygame.transform.scale(pygame.image.load("bg_level_2.jpg"), (WIDTH, HEIGHT))
    bg_dialoglevel3 = pygame.transform.scale(pygame.image.load("bg_level_3.jpg"), (WIDTH, HEIGHT))

    # Karakter utama dan musuh setiap level
    char_img = pygame.transform.scale(pygame.image.load("karakter utama.png"), (430, 550))
    char_imglevel1 = pygame.transform.scale(pygame.image.load("karakter level 1.png"), (430, 550))
    char_imglevel2 = pygame.transform.scale(pygame.image.load("karakter level 2.png"), (430, 550))
    char_imglevel3 = pygame.transform.scale(pygame.image.load("karakter level 3.png"), (430, 550))

    # Tombol-tombol navigasi
    btn_play = pygame.transform.scale(pygame.image.load("PLAY.png"), (230, 240))
    btn_next = pygame.transform.scale(pygame.image.load("NEXT.png"), (100, 120))
    btn_skip = pygame.transform.scale(pygame.image.load("SKIP.png"), (100, 120))
    btn_back = pygame.transform.scale(pygame.image.load("back.png"), (140, 150))

    # Tombol setiap level
    btn_level1 = pygame.transform.scale(pygame.image.load("level 1.png"), (230, 240))
    btn_level2 = pygame.transform.scale(pygame.image.load("level 2.png"), (230, 240))
    btn_level3 = pygame.transform.scale(pygame.image.load("level 3.png"), (230, 240))

```

Kode ini sebagai proses pemuatan dan penyiapan aset gambar yang digunakan dalam game berbasis Pygame. Secara proses, kode dijalankan di dalam blok try untuk memastikan bahwa seluruh file gambar dapat dimuat dengan aman; jika terjadi kesalahan seperti file tidak ditemukan atau nama file tidak sesuai, program dapat ditangani pada bagian except (yang biasanya berisi pesan error atau penghentian program). Pada tahap awal, sistem memuat berbagai gambar latar belakang (background) untuk menu utama, dialog, dan halaman level, lalu

	menyesuaikan ukurannya dengan resolusi layar menggunakan pygame.transform.scale agar tampil penuh dan proporsional. Selanjutnya, kode memuat gambar karakter utama serta karakter musuh untuk setiap level, yang diskalakan ke ukuran tertentu agar konsisten dengan desain tampilan permainan. Setelah itu, berbagai gambar tombol navigasi seperti tombol play, next, skip, dan back dimuat untuk mendukung interaksi pemain dalam berpindah halaman. Terakhir, tombol khusus untuk setiap level juga disiapkan sehingga pemain dapat memilih level yang tersedia.
	 Kode ini menyiapkan tampilan tombol “Jawab” pada game dengan mekanisme penanganan kesalahan (error handling) yang aman. Secara proses, program terlebih dahulu mencoba memuat file gambar Jawab.png menggunakan pygame.image.load() dan convert_alpha() agar gambar mendukung transparansi dan tampil lebih optimal. Jika file gambar tidak ditemukan atau gagal dimuat, blok except pertama akan dijalankan dan menampilkan peringatan di konsol, lalu sistem membuat tombol cadangan (placeholder) secara manual menggunakan pygame.Surface dengan latar transparan. Pada proses pembuatan placeholder, program menggambar persegi panjang berwarna biru dengan sudut membulat sebagai bentuk tombol, kemudian membuat objek font dan merender teks “JAWAB” berwarna putih, yang selanjutnya ditempelkan (blit) ke tengah permukaan tombol tersebut. Dengan cara ini, fungsi tombol tetap tersedia meskipun aset gambar asli tidak ada. Blok except terakhir digunakan sebagai penanganan kesalahan tingkat lanjut, di mana jika terjadi error lain yang lebih serius saat pemuatan aset, program akan menampilkan pesan error, menghentikan Pygame, dan menutup aplikasi secara aman.

```
423     # tampilan Tombol jawab
424     try:
425         label_jawab_img = pygame.image.load("Jawab.png").convert_alpha()
426     except:
427         print("Warning: Jawab.png not found, creating placeholder")
428         # Placeholder tombol jawab
429         label_jawab_img = pygame.Surface((170, 170), pygame.SRCALPHA)
430         pygame.draw.rect(label_jawab_img, (60, 140, 255), (0, 0, 170, 170), border_radius=10)
431         font_jawab = pygame.font.SysFont("arialblack", 26)
432         text_jawab = font_jawab.render("JAWAB", True, WHITE)
433         label_jawab_img.blit(text_jawab, (35, 70))
434
435 except Exception as e:
436     print(f"Error loading images: {e}")
437     pygame.quit()
438     sys.exit()
439
```

```
440
441  # BUAT LEVEL DENGAN OOP
442  # ========================================
443
444  # Buat instance Level untuk setiap level
445  level1 = Level(1, dialog_level1, bg_dialoglevel1, char_imglevel1)
446  level2 = Level(2, dialog_level2, bg_dialoglevel2, char_imglevel2)
447  level3 = Level(3, dialog_level3, bg_dialoglevel3, char_imglevel3)
448
449  # Unlock level 1 secara default
450  level1.unlock()
451
452  # Dictionary untuk akses level
453  levels = {1: level1, 2: level2, 3: level3}
454
```

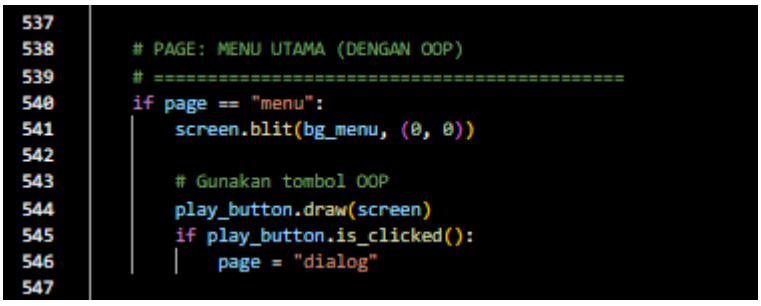
Kode ini membangun sistem level permainan menggunakan pendekatan Object-Oriented Programming (OOP). Secara proses, setiap level dibuat sebagai sebuah objek dari kelas Level dengan parameter berupa nomor level, data dialog cerita, background khusus level, dan gambar karakter musuh yang akan ditampilkan. Dengan cara ini, seluruh informasi yang berkaitan dengan satu level dikemas dalam satu objek, sehingga pengelolaan level menjadi lebih rapi dan terstruktur.
Setelah objek level dibuat, level pertama langsung dibuka dengan memanggil metode unlock(), yang menandakan bahwa pemain dapat mengakses level tersebut sejak awal permainan. Level lain tetap dalam kondisi terkunci sampai sistem membuka kuncinya berdasarkan progres pemain. Selanjutnya, seluruh objek level disimpan dalam sebuah dictionary bernama levels, dengan nomor level sebagai key. Struktur ini memudahkan game untuk mengakses data level secara dinamis, misalnya saat pemain memilih level tertentu, sistem cukup memanggil levels[level] tanpa perlu kondisi yang rumit.

```
455
456  # BUAT TOMBOL DENGAN OOP
457  # ========================================
458
459  # Buat tombol dengan OOP
460  play_button = NormalButton(280, 180, btn_play, audio_manager)
461  next_button = NormalButton(540, 400, btn_next, audio_manager)
462  skip_button = NormalButton(650, 400, btn_skip, audio_manager)
463  back_button = NormalButton(20, 20, btn_back, audio_manager)
464
465  # Tombol level dengan polymorphism
466  level1_button = NormalButton(290, 100, btn_level1, audio_manager)
467  level2_button = NormalButton(290, 200, btn_level2, audio_manager) if unlocked_level >= 2 else LockedButton(290, 200, btn_level2, audio_manager)
468  level3_button = NormalButton(290, 300, btn_level3, audio_manager) if unlocked_level >= 3 else LockedButton(290, 300, btn_level3, audio_manager)
469
```

Kode ini membuat dan mengelola tombol antarmuka permainan menggunakan pendekatan Object-Oriented Programming (OOP). Secara proses, setiap tombol dibuat sebagai objek dari kelas NormalButton, dengan parameter posisi tombol pada layar (koordinat x dan y), gambar tombol yang akan ditampilkan, serta audio_manager yang berfungsi memutar efek suara saat tombol ditekan. Tombol-tombol umum seperti play, next, skip, dan back selalu dibuat sebagai NormalButton karena dapat langsung digunakan oleh pemain.
Pada bagian tombol level, kode ini menerapkan konsep polymorphism, di mana satu jenis tombol level dapat memiliki perilaku yang berbeda tergantung kondisi

	progres permainan. Jika level sudah terbuka sesuai nilai unlocked_level, maka tombol dibuat sebagai NormalButton dan dapat ditekan. Sebaliknya, jika level belum terbuka, tombol dibuat sebagai LockedButton yang secara tampilan maupun fungsi akan menunjukkan bahwa level tersebut masih terkunci dan tidak dapat diakses.
	 Kode ini nti dari proses jalannya game (game loop) yang terus berjalan selama variabel running bernilai True. Secara proses, game loop dijalankan dengan

```
# GAME LOOP
# ==========================================
clock = pygame.time.Clock()
mouse_clicked = False  # Flag untuk mencegah klik berulang
running = True  # Flag utama game loop
while running:
    clock.tick(60)  # 60 FPS

    # EVENT HANDLING
    # ==========================================
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            running = False

        # Handle input keyboard di halaman quiz
        if page == "quiz" and event.type == pygame.KEYDOWN:
            # Hapus karakter dengan backspace
            if event.key == pygame.K_BACKSPACE:
                input_text = input_text[:-1]

            # Submit jawaban dengan Enter
            elif event.key == pygame.K_RETURN:
                # Gunakan QuizQuestion untuk cek jawaban
                if quiz_number < len(quiz_questions[level]):
                    current_question = quiz_questions[level][quiz_number]

                    if current_question.check_answer(input_text):
                        answered.append(f"Soal {quiz_number + 1}: ✓")
                        quiz_number += 1

                        # Jika semua soal selesai
                        if quiz_number >= len(quiz_questions[level]):
                            # Jika level 3 selesai, tampilkan winner screen
                            if level == 3:
                                page = "winner"
                                winner_timer = pygame.time.get_ticks()
                            else:
                                # Untuk level 1 dan 2, unlock level berikutnya
                                if level < 3:
                                    unlocked_level = max(unlocked_level, level + 1)
                                    # Update tombol dengan polymorphism
                                    if level == 1:
                                        level2_button = NormalButton(290, 200, btn_level2, audio_manager)
                                    elif level == 2:
                                        level3_button = NormalButton(290, 300, btn_level3, audio_manager)

                                    # Unlock level di OOP
                                    levels[level + 1].unlock()

                                quiz_number = 0
                                page = "level"
                            notif_quiz = ""
                        else:
                            notif_quiz = "Jawaban Anda BENAR!"
                    else:
                        answered.append(f"Soal {quiz_number + 1}: ✗ ({input_text})")
                        notif_quiz = "Jawaban Anda SALAH!"

                    input_text = ""

            # Tambah karakter biasa
            else:
                input_text += event.unicode
```

	kecepatan 60 frame per detik menggunakan clock.tick(60) agar permainan berjalan stabil dan tidak terlalu cepat. Di dalam loop ini, sistem menangani seluruh event yang terjadi, seperti menutup jendela game, input keyboard, dan interaksi pemain. Jika pemain menutup jendela, event pygame.QUIT akan mengubah nilai running menjadi False sehingga game berhenti dengan aman. Fokus utama kode ini adalah penanganan input keyboard saat game berada pada halaman quiz. Ketika pemain menekan tombol backspace, karakter terakhir pada input_text akan dihapus, sedangkan saat pemain menekan tombol Enter, sistem akan memproses jawaban kuis. Program mengambil soal aktif berdasarkan level dan quiz_number, lalu memeriksa jawaban pemain menggunakan metode check_answer() dari objek QuizQuestion. Jika jawaban benar, sistem mencatat hasil, menaikkan nomor soal, dan mengecek apakah semua soal dalam level telah selesai. Jika seluruh soal pada level terakhir (level 3) telah dijawab, game akan berpindah ke halaman kemenangan (winner) dan mencatat waktu tampilannya. Jika masih ada level berikutnya, sistem akan membuka level baru, memperbarui tombol level menggunakan konsep polymorphism, serta memanggil metode unlock() pada objek level terkait. Sebaliknya, jika jawaban salah, sistem tetap mencatat hasil dan menampilkan notifikasi kesalahan. Setiap kali proses pengecekan selesai, input pemain akan dikosongkan untuk soal berikutnya. Jika pemain menekan tombol keyboard selain Enter dan Backspace, karakter tersebut akan ditambahkan ke input_text.
	 Kode ini menampilkan dan mengatur halaman menu utama dalam game menggunakan pendekatan Object-Oriented Programming (OOP). Secara proses, ketika variabel page bernilai "menu", sistem akan menampilkan background menu utama dengan menggambar (blit) gambar bg_menu ke layar. Setelah itu, objek play_button dipanggil melalui metode draw() untuk menampilkan tombol Play di layar. Tombol ini merupakan objek OOP yang sudah memiliki perilaku sendiri, seperti mendeteksi klik dan memutar efek suara jika ditekan. Dalam alur eksekusi, setiap frame game akan mengecek apakah tombol Play ditekan melalui metode is_clicked(). Jika metode tersebut mengembalikan nilai benar, berarti pemain menekan tombol Play, sehingga game akan mengubah nilai page menjadi "dialog". Perubahan nilai page ini menyebabkan game berpindah dari menu utama ke halaman dialog pembuka pada frame berikutnya.

```
548
549     # PAGE: DIALOG UTAMA (DENGAN OOP)
550     # ========================================
551     elif page == "dialog":
552         screen.blit(bg_dialog, (0, 0))
553         screen.blit(char_img, (10, 0))
554
555         # Buat kotak dialog
556         dialog_box = pygame.Rect(50, 360, 700, 120)
557         pygame.draw.rect(screen, WHITE, dialog_box, border_radius=18)
558         pygame.draw.rect(screen, BLACK, dialog_box, 5, border_radius=18)
559
560         # Render teks dialog
561         text = dialogs[dialog_index]
562         surf = pygame.font.SysFont("arialblack", 27).render(text, True, BLACK)
563         screen.blit(surf, (70, 395))
564
565         # Gunakan tombol OOP
566         next_button.draw(screen)
567         skip_button.draw(screen)
568
569         if next_button.is_clicked():
570             dialog_index += 1
571
572         if skip_button.is_clicked():
573             dialog_index = len(dialogs)
574
575         # Jika dialog selesai, lanjut ke level selection
576         if dialog_index >= len(dialogs):
577             page = "level"
578
```

Kode ini mengatur halaman dialog utama (pembuka cerita) dalam game dengan pendekatan OOP. Secara proses, ketika nilai page adalah "dialog", sistem akan menampilkan background dialog dan karakter utama di layar sebagai pengantar cerita. Setelah itu, program membuat sebuah kotak dialog menggunakan objek Rect yang digambar sebagai persegi panjang berwarna putih dengan garis tepi hitam, sehingga teks dialog terlihat jelas dan terpisah dari background.

Dialog ditampilkan secara bertahap dengan mengambil teks dari list dialogs berdasarkan nilai dialog_index. Setiap frame, teks dialog tersebut dirender menjadi surface lalu ditampilkan ke layar. Interaksi pemain diatur melalui dua tombol OOP, yaitu tombol next dan skip. Ketika tombol next ditekan, indeks dialog bertambah satu sehingga dialog berikutnya akan ditampilkan. Jika tombol skip ditekan, indeks dialog langsung diatur ke panjang list dialog, yang menandakan bahwa seluruh dialog telah dilewati.

Pada tahap akhir proses, sistem mengecek apakah seluruh dialog telah ditampilkan. Jika dialog_index sudah mencapai atau melebihi jumlah dialog, maka game otomatis berpindah ke halaman pemilihan level (page = "level").

```
579
580     # PAGE: LEVEL SELECTION (DENGAN OOP)
581     # ========================================
582     elif page == "level":
583         screen.blit(bg_level, (0, 0))
584
585         # Gunakan tombol OOP dengan polymorphism
586         back_button.draw(screen)
587         level1_button.draw(screen)
588         level2_button.draw(screen)
589         level3_button.draw(screen)
590
591         # LEVEL 1 (selalu terbuka)
592         if level1_button.is_clicked():
593             level = 1
594             level_dialog_index = 0
595             page = "dialog_level"
596
597         # LEVEL 2 (tergantung unlock status) - polymorphism
598         if unlocked_level >= 2 and level2_button.is_clicked():
599             level = 2
600             level_dialog_index = 0
601             page = "dialog_level"
602
603         # LEVEL 3 (tergantung unlock status) - polymorphism
604         if unlocked_level >= 3 and level3_button.is_clicked():
605             level = 3
606             level_dialog_index = 0
607             page = "dialog_level"
608
609         # Tombol kembali ke menu
610         if back_button.is_clicked():
611             page = "menu"
612             dialog_index = 0
613
```

Kode ini mengatur halaman pemilihan level (level selection) dalam game dengan pendekatan Object-Oriented Programming (OOP). Secara proses, ketika nilai page adalah "level", sistem akan menampilkan background halaman level dan menggambar seluruh tombol yang dibutuhkan, termasuk tombol kembali dan tombol level. Tombol-tombol level dibuat menggunakan konsep polymorphism, di mana tombol dapat berperilaku berbeda tergantung status level yang telah terbuka atau masih terkunci.

Dalam alur interaksi, tombol level 1 selalu aktif sehingga pemain dapat langsung memulai permainan dari level pertama. Tombol level 2 dan level 3 hanya dapat digunakan jika nilai unlocked_level telah memenuhi syarat, yang menandakan bahwa pemain telah menyelesaikan level sebelumnya. Ketika salah satu tombol level ditekan, sistem akan menyimpan level yang dipilih, mengatur ulang indeks dialog level, dan mengubah halaman ke "dialog_level" agar dialog khusus level ditampilkan terlebih dahulu. Selain itu, tombol kembali memungkinkan pemain kembali ke menu utama dengan mengatur ulang indeks dialog utama.

```
614
615     # PAGE: DIALOG LEVEL (DENGAN OOP)
616     # ========================================
617     elif page == "dialog_level":
618         # Gunakan Level class untuk mendapatkan data
619         current_level = levels[level]
620
621         # Render background dan karakter dari Level object
622         screen.blit(current_level.background_img, (0, 0))
623         screen.blit(current_level.character_img, (10, 0))
624
625         # Kotak dialog
626         dialog_box = pygame.Rect(50, 360, 700, 120)
627         pygame.draw.rect(screen, WHITE, dialog_box, border_radius=18)
628         pygame.draw.rect(screen, BLACK, dialog_box, 5, border_radius=18)
629
630         # Render teks dialog dari Level object
631         text = current_level.get_current_dialog(level_dialog_index)
632         surf = pygame.font.SysFont("arialblack", 27).render(text, True, BLACK)
633         screen.blit(surf, (70, 395))
634
635         # Gunakan tombol OOP
636         next_button.draw(screen)
637         skip_button.draw(screen)
638
639         if next_button.is_clicked():
640             level_dialog_index += 1
641
642         if skip_button.is_clicked():
643             level_dialog_index = len(current_level.dialogs)
644
645         # Jika dialog selesai, lanjut ke quiz
646         if level_dialog_index >= len(current_level.dialogs):
647             page = "quiz"
648
```

Kode ini mengatur halaman dialog khusus setiap level dengan pendekatan Object-Oriented Programming (OOP). Secara proses, ketika nilai page adalah "dialog_level", sistem terlebih dahulu mengambil objek level aktif dari dictionary levels berdasarkan level yang dipilih pemain. Objek current_level ini menyimpan seluruh data penting level, seperti background, karakter musuh, dan daftar dialog. Dengan memanfaatkan objek level, kode menjadi lebih rapi karena data tidak lagi diambil secara terpisah.

Selanjutnya, program merender background dan karakter yang sesuai langsung dari properti objek Level. Setelah itu, sebuah kotak dialog digambar untuk menampilkan teks dialog. Teks dialog diambil melalui metode get_current_dialog() milik objek Level, yang menerima indeks dialog saat ini, sehingga dialog dapat ditampilkan secara berurutan. Interaksi pemain kembali diatur menggunakan tombol next dan skip. Tombol next menampilkan dialog berikutnya, sedangkan tombol skip langsung melewati seluruh dialog level.

Ketika seluruh dialog level telah selesai ditampilkan, sistem akan mengubah nilai page menjadi "quiz"

```
650     # PAGE: QUIZ (DENGAN OOP)
651     # ========================================
652     elif page == "quiz":
653         screen.fill(WHITE)
654
655         # Ambil soal dari QuizQuestion object
656         current_question = quiz_questions[level][quiz_number]
657         soal = current_question.question
658         jawab = current_question.answer
659
660         # Judul level
661         surf = pygame.font.SysFont("arialblack", 50).render(f"LEVEL {level}", True, BLUE)
662         screen.blit(surf, (WIDTH / 2 - surf.get_width() / 2, 40))
663
664         # Render soal (multiline)
665         font_question = pygame.font.SysFont("arialblack", 24)
666         render_text_multiline(soal, 50, 150, font_question, BLACK, max_width=670)
```

Kode ini mengelola halaman kuis, yang menjadi inti interaksi pemain dalam menjawab soal pada setiap level menggunakan pendekatan Object-Oriented Programming (OOP). Secara proses, ketika page bernilai "quiz", sistem mengambil objek soal aktif dari quiz_questions berdasarkan level dan nomor soal (quiz_number). Teks soal kemudian direnderer ke layar secara multiline agar tetap rapi meskipun panjang, sementara judul level ditampilkan sebagai penanda progres pemain.

Input jawaban pemain ditangani melalui input box dengan lebar yang menyesuaikan panjang teks yang diketik, sehingga tampilan tetap responsif. Pemain dapat mengirim jawaban melalui tombol “JAWAB” yang ditampilkan sebagai aset gambar. Ketika tombol tersebut diklik, sistem memanggil method check_answer() dari objek QuizQuestion untuk memvalidasi jawaban pemain. Jika jawaban benar, notifikasi akan ditampilkan, nomor soal bertambah, dan sistem mengecek apakah seluruh soal dalam level telah selesai. Jika level terakhir selesai, game berpindah ke halaman kemenangan dan level ditandai selesai melalui method OOP. Jika belum, level berikutnya akan dibuka, tombol level diperbarui menggunakan polymorphism, dan game kembali ke halaman pemilihan level. Sebaliknya, jika jawaban salah, sistem hanya menampilkan notifikasi kesalahan tanpa menaikkan progres soal.

```
668         # ================ INPUT BOX ================
669         font = pygame.font.SysFont("arialblack", 12)
670         text_surf = font.render(input_text, True, BLACK)
671
672         # Lebar input box dinamis
673         min_width = 200
674         max_width = 500
675         padding = 20
676
677         box_width = max(min_width, text_surf.get_width() + padding)
678         box_width = min(box_width, max_width)
679
680         # Buat dan render input box
681         input_box_rect = pygame.Rect(50, 300, box_width, 50)
682         pygame.draw.rect(screen, WHITE, input_box_rect, border_radius=10)
683         pygame.draw.rect(screen, BLACK, input_box_rect, 3, border_radius=10)
684
685         # Render teks input
686         screen.blit(text_surf, (input_box_rect.x + 10, input_box_rect.y + 10))
687
```

Kode ini tampilan utama halaman kuis, tempat pemain membaca soal dan bersiap memasukkan jawaban. Seluruh logika di dalamnya hanya dijalankan ketika variabel page bernilai "quiz", yang menandakan bahwa permainan telah memasuki tahap evaluasi kemampuan pemain.

Langkah pertama yang dilakukan sistem adalah membersihkan layar menggunakan warna putih. Hal ini bertujuan agar tampilan kuis tidak bercampur dengan elemen visual dari halaman sebelumnya seperti dialog atau pemilihan level, sehingga fokus pemain tertuju pada soal.

Selanjutnya, sistem mengambil soal yang sedang aktif melalui struktur data quiz_questions. Pengambilan dilakukan berdasarkan level yang sedang dimainkan dan nomor soal saat ini. Karena setiap soal direpresentasikan sebagai objek QuizQuestion, pendekatan ini menunjukkan penerapan Object-Oriented Programming, di mana soal dan jawabannya dibungkus dalam satu entitas yang terstruktur.

Setelah objek soal diperoleh, teks soal dan jawaban diekstrak dari atribut objek tersebut. Dengan cara ini, sistem tidak perlu memproses data mentah, melainkan cukup mengakses properti yang telah disediakan oleh class, sehingga kode menjadi lebih rapi, mudah dipahami, dan mudah dikembangkan.

Bagian berikutnya menampilkan judul level di layar. Judul ini dirender secara dinamis sesuai level yang sedang dimainkan, sehingga pemain selalu mengetahui konteks permainan yang sedang dijalani. Penempatan teks di tengah layar memberikan kesan visual yang jelas dan profesional. Terakhir, soal ditampilkan menggunakan fungsi render_text_multiline.

```
688         # ================ TOMBOL JAWAB ================
689         btn_jawab = pygame.Rect(540, 230, 180, 180)
690         label_jawab = pygame.transform.scale(label_jawab_img, (btn_jawab.width, btn_jawab.height))
691         screen.blit(label_jawab, (btn_jawab.x, btn_jawab.y))
692
```

kode ini membuat dan menampilkan tombol “JAWAB” pada halaman kuis, yang digunakan pemain untuk mengirimkan jawaban mereka. Proses dimulai dengan pembuatan objek Rect bernama btn_jawab, yang merepresentasikan area tombol secara posisi dan ukuran di layar. Objek ini sangat penting karena selain menentukan letak tombol, Rect juga digunakan untuk mendeteksi apakah tombol tersebut diklik oleh mouse.

Selanjutnya, gambar tombol label_jawab_img disesuaikan ukurannya menggunakan pygame.transform.scale agar pas dengan ukuran Rect tombol. Penyesuaian ini memastikan tampilan tombol tetap proporsional dan konsisten dengan area kliknya, sehingga tidak terjadi perbedaan antara tampilan visual dan area interaksi.

Terakhir, gambar tombol yang telah diskalakan ditampilkan ke layar menggunakan screen.blit pada posisi yang sama dengan Rect tombol. Dengan demikian, pemain dapat melihat tombol “JAWAB” secara jelas di layar, dan sistem siap mendeteksi interaksi klik pada area tersebut.

```
693         # ================ NOTIFIKASI HASIL ================
694         if notif_quiz != "":
695             color = (0, 180, 0) if "BENAR" in notif_quiz else (200, 0, 0)
696             notif_surf = pygame.font.SysFont("arialblack", 28).render(notif_quiz, True, color)
697             screen.blit(notif_surf, (50, 370))
698
```

Kode ini menampilkan notifikasi hasil jawaban kuis kepada pemain sebagai bentuk umpan balik langsung. Prosesnya dimulai dengan pengecekan kondisi notif_quiz, yang menyimpan pesan hasil evaluasi jawaban. Jika variabel ini tidak kosong, berarti sistem memiliki pesan yang perlu ditampilkan di layar.

Selanjutnya, warna teks notifikasi ditentukan secara otomatis berdasarkan isi pesan. Jika di dalam notif_quiz terdapat kata “BENAR”, maka warna teks diatur menjadi hijau untuk menandakan jawaban yang benar. Sebaliknya, jika tidak mengandung kata tersebut, warna teks diubah menjadi merah sebagai indikator jawaban yang salah. Mekanisme ini membantu pemain memahami hasil jawaban secara visual dengan cepat.

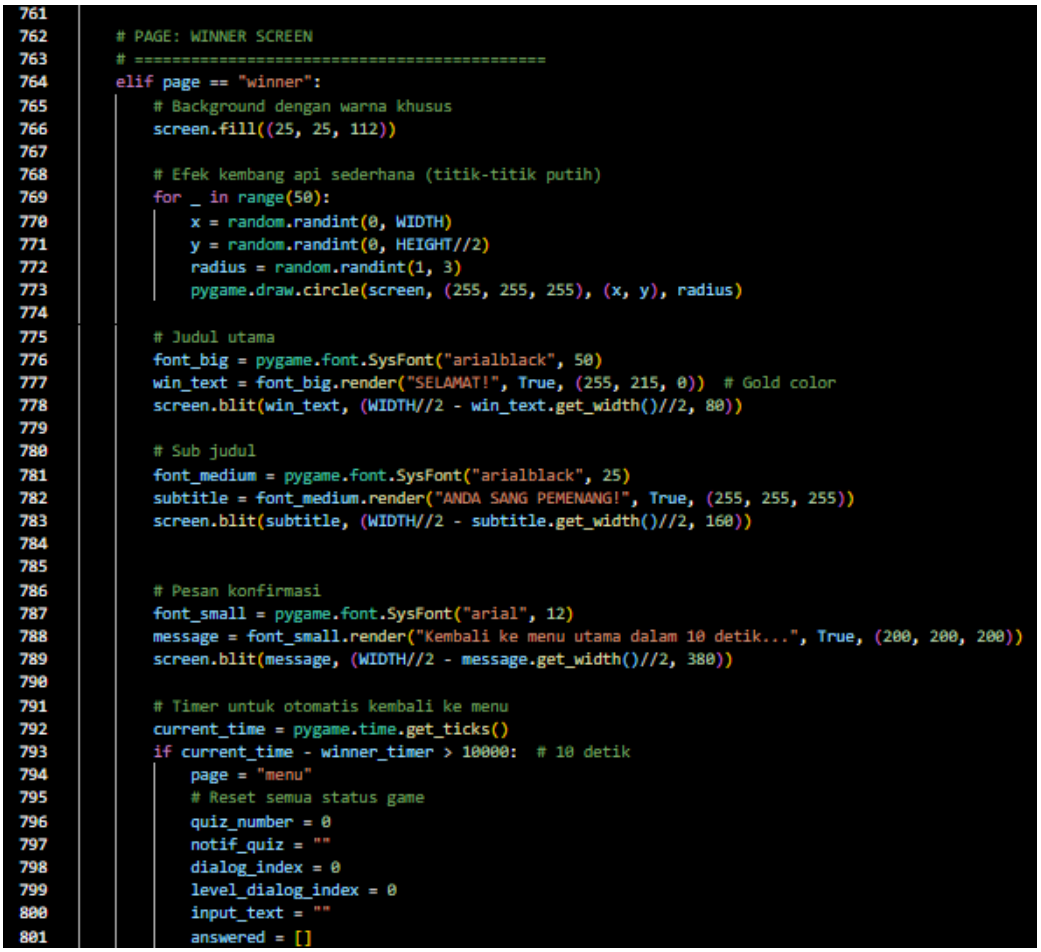
Pesan notifikasi kemudian dirender menjadi sebuah surface menggunakan font tebal agar mudah dibaca, lalu ditampilkan pada posisi tertentu di layar menggunakan screen.blit.

```
699         # ================ HANDLE MOUSE CLICK ================
700         if pygame.mouse.get_pressed()[0] and not mouse_clicked:
701             mouse_clicked = True
702
703             # Gunakan AudioManager untuk play sound
704             audio_manager.play_click()
705
706             mx, my = pygame.mouse.get_pos()
707
708             # Tombol kembali ke level selection
709             if image_btn(btn_back, 20, 20):
710                 quiz_number = 0
711                 notif_quiz = ""
712                 page = "level"
713
```

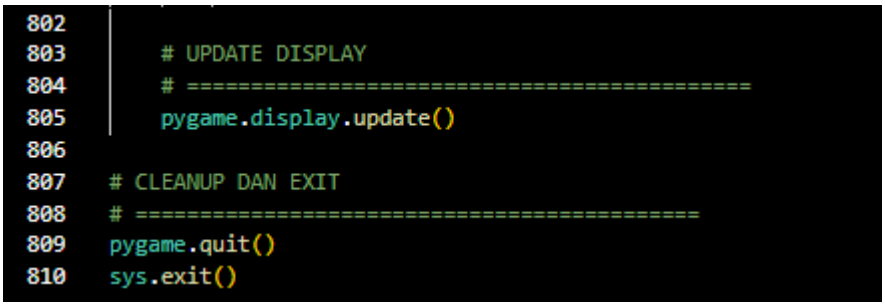
```
            # Tombol "JAWAB" ditekan
            if btn_jawab.collidepoint(mx, my):
                # Gunakan method dari QuizQuestion untuk cek jawaban
                if current_question.check_answer(input_text):
                    notif_quiz = "Jawaban Anda BENAR!"
                    quiz_number += 1

                    # Jika semua soal selesai
                    if quiz_number >= len(quiz_questions[level]):
                        # Jika level 3 selesai, tampilkan winner screen
                        if level == 3:
                            page = "winner"
                            winner_timer = pygame.time.get_ticks()
                            # Tandai level sebagai selesai di OOP
                            levels[level].complete()
                        else:
                            # Untuk level 1 dan 2, unlock level berikutnya
                            if level < 3:
                                unlocked_level = max(unlocked_level, level + 1)
                                # Update tombol dengan polymorphism
                                if level == 1:
                                    level2_button = NormalButton(290, 200, btn_level2, audio_manager)
                                elif level == 2:
                                    level3_button = NormalButton(290, 300, btn_level3, audio_manager)

                                # Unlock level di OOP
                                levels[level + 1].unlock()

                            quiz_number = 0
                            notif_quiz = ""
                            page = "level"
                else:
                    notif_quiz = "Jawaban Anda SALAH!"

                # Reset input
                input_text = ""
```

```
        # Reset flag klik mouse
        if not pygame.mouse.get_pressed()[0]:
            mouse_clicked = False

        # Tombol kembali (alternative handling dengan fungsi lama)
        if image_btn(btn_back, 20, 20):
            quiz_number = 0
            notif_quiz = ""
            page = "level"
```

Kode ini menangani interaksi klik mouse pemain pada halaman kuis, khususnya saat menekan tombol JAWAB dan tombol kembali. Proses dimulai dengan pengecekan apakah tombol mouse kiri ditekan dan flag mouse_clicked masih bernilai False, yang bertujuan untuk mencegah klik berulang dalam satu kali tekan. Ketika klik terdeteksi, sistem langsung memutar efek suara melalui AudioManager sebagai umpan balik audio, lalu mengambil posisi kursor mouse. Jika pemain menekan tombol kembali, sistem akan mereset nomor soal dan notifikasi, kemudian mengarahkan kembali ke halaman pemilihan level. Sebaliknya, jika area tombol JAWAB terdeteksi melalui collidepoint, sistem akan mengecek jawaban pemain menggunakan method check_answer() dari objek QuizQuestion. Jika jawaban benar, variabel notif_quiz diisi dengan pesan “Jawaban Anda BENAR!” dan nomor soal ditambah. Jika semua soal pada level telah selesai, game akan menentukan apakah pemain masuk ke halaman winner

	(jika level terakhir) atau membuka level berikutnya menggunakan mekanisme unlock berbasis OOP dan polymorphism pada tombol level. Jika jawaban salah, notif_quiz diisi dengan pesan “Jawaban Anda SALAH!”. Nilai notif_quiz inilah yang kemudian digunakan oleh sistem notifikasi untuk menampilkan hasil jawaban di layar. Warna teks ditentukan secara otomatis berdasarkan isi pesan, yaitu hijau untuk jawaban benar dan merah untuk jawaban salah, sehingga pemain dapat memahami hasil jawabannya secara visual dengan cepat.
	 Kode ini menampilkan halaman kemenangan (Winner Screen) setelah pemain berhasil menyelesaikan seluruh level permainan. Ketika variabel page bernilai "winner", sistem akan mengganti tampilan layar dengan latar belakang berwarna biru gelap untuk menciptakan suasana spesial dan berbeda dari halaman lainnya. Sebagai efek visual tambahan, kode membuat efek kembang api sederhana berupa titik-titik putih yang digambar secara acak di bagian atas layar. Efek ini dihasilkan menggunakan perulangan dan fungsi random, sehingga setiap frame menampilkan pola titik yang bervariasi dan memberi kesan perayaan kemenangan.

```
761
762     # PAGE: WINNER SCREEN
763     # ========================================
764     elif page == "winner":
765         # Background dengan warna khusus
766         screen.fill((25, 25, 112))
767
768         # Efek kembang api sederhana (titik-titik putih)
769         for _ in range(50):
770             x = random.randint(0, WIDTH)
771             y = random.randint(0, HEIGHT//2)
772             radius = random.randint(1, 3)
773             pygame.draw.circle(screen, (255, 255, 255), (x, y), radius)
774
775         # Judul utama
776         font_big = pygame.font.SysFont("arialblack", 50)
777         win_text = font_big.render("SELAMAT!", True, (255, 215, 0))  # Gold color
778         screen.blit(win_text, (WIDTH//2 - win_text.get_width()//2, 80))
779
780         # Sub judul
781         font_medium = pygame.font.SysFont("arialblack", 25)
782         subtitle = font_medium.render("ANDA SANG PEMENANG!", True, (255, 255, 255))
783         screen.blit(subtitle, (WIDTH//2 - subtitle.get_width()//2, 160))
784
785
786         # Pesan konfirmasi
787         font_small = pygame.font.SysFont("arial", 12)
788         message = font_small.render("Kembali ke menu utama dalam 10 detik...", True, (200, 200, 200))
789         screen.blit(message, (WIDTH//2 - message.get_width()//2, 380))
790
791         # Timer untuk otomatis kembali ke menu
792         current_time = pygame.time.get_ticks()
793         if current_time - winner_timer > 10000:  # 10 detik
794             page = "menu"
795             # Reset semua status game
796             quiz_number = 0
797             notif_quiz = ""
798             dialog_index = 0
799             level_dialog_index = 0
800             input_text = ""
801             answered = []
```

	Selanjutnya, sistem menampilkan judul utama “SELAMAT!” dengan ukuran font besar dan warna emas sebagai simbol keberhasilan, diikuti dengan subjudul “ANDA SANG PEMENANG!” yang mempertegas status kemenangan pemain. Seluruh teks diposisikan di tengah layar agar terlihat fokus dan mudah dibaca. Di bagian bawah, ditampilkan pesan informasi bahwa pemain akan otomatis kembali ke menu utama dalam waktu tertentu. Untuk mendukung hal ini, sistem menggunakan pygame.time.get_ticks() sebagai timer. Jika selisih waktu saat ini dengan waktu kemenangan (winner_timer) telah melebihi 10 detik, maka game secara otomatis berpindah kembali ke halaman menu utama.
	 Kode ini merupakan tahap akhir dalam siklus eksekusi game yang berfungsi untuk memperbarui tampilan layar serta menutup aplikasi secara aman. Perintah pygame.display.update() digunakan untuk menampilkan seluruh perubahan visual yang telah digambar pada frame tersebut ke layar. Tanpa pemanggilan fungsi ini, semua proses penggambaran seperti background, tombol, teks, dan animasi tidak akan terlihat oleh pemain, karena masih berada di buffer internal. Setelah game loop selesai atau kondisi running bernilai False, program akan keluar dari loop utama dan masuk ke tahap pembersihan. Fungsi pygame.quit() dipanggil untuk menghentikan seluruh modul Pygame yang sedang berjalan, termasuk sistem grafis, audio, dan input. Langkah ini penting untuk mencegah terjadinya error atau resource yang masih terbuka setelah aplikasi ditutup. Terakhir, sys.exit() digunakan untuk menghentikan eksekusi program secara keseluruhan.

Tabel 2. 1 Tabel Analisis Kode

2.5 Fitur Aplikasi

Dalam pembuatan game ini mengimplementasikan konsep Object-Oriented Progamming (OOP) yang meliputi

2.5.1 Object-Oriented Progamming (OOP)

A. Encapsulation

Encapsulation adalah konsep OOP yang membungkus data (atribut) dan fungsi (method) dalam satu kelas, serta membatasi akses langsung ke data menggunakan

atribut private dan getter.Penerapan konsep encapsulation pada kode ini bertujuan untuk melindungi data penting agar tidak dapat diakses atau dimodifikasi secara langsung dari luar kelas. Penerapan enkapsulasi pada game ini terlihat pada kelas AudioManager

```
7   # ENCAPSULATION: Kelas untuk mengelola audio dengan atribut private
8   class AudioManager:
9       def __init__(self):
10          self._enabled = False  # Private attribute
11          self._background_music = None
12          self._click_sound = None
13          self._volume = 0.7
14
```

Gambar 2. 3 Kode Encapsulation

Akses terhadap atribut privat tersebut dibatasi melalui method tertentu dan property getter, sehingga bagian lain dari program hanya dapat membaca data tanpa mengubahnya secara langsung

```
57      @property
58      def enabled(self):
59          """Getter untuk enabled status"""
60          return self._enabled
61
62      @property
63      def click_sound(self):
64          """Getter untuk click sound"""
65          return self._click_sound
```

Gambar 2. 4 Kode Encapsulation

enkapsulasi juga diterapkan pada kelas QuizQuestion, di mana soal dan jawaban disimpan sebagai atribut privat.

```
108 # ENCAPSULATION: Kelas untuk mengelola soal quiz
109 class QuizQuestion:
110     def __init__(self, question, answer):
111         self._question = question  # Private attribute
112         self._answer = answer      # Private attribute
113
```

Gambar 2. 5 Kode Encapsulation

Proses validasi jawaban tidak dilakukan di luar kelas, melainkan melalui method check_answer()

```
131     def check_answer(self, user_answer):
132         """Cek jawaban user"""
133         user_normalized = self.normalize_text(user_answer)
134         correct_normalized = self.normalize_text(self._answer)
135         return user_normalized == correct_normalized
136
```

Gambar 2. 6 Kode Encapsulation

B. Inheritance

Inheritance konsep OOP yang memungkinkan kelas anak mewarisi atribut dan method dari kelas induk. Penerapan konsep Inheritance (pewarisan) pada kode ini bertujuan untuk mengurangi duplikasi kode, meningkatkan keteraturan struktur program, serta mempermudah pengembangan dan pemeliharaan sistem. Penerapan inheritance pada game ini terlihat pada kelas BaseButton berperan sebagai kelas induk yang menyimpan atribut dan method dasar yang dimiliki oleh seluruh tombol dalam game, seperti posisi, gambar, serta kemampuan menggambar tombol ke layar.

```
67  # INHERITANCE: Base class untuk semua tombol
68  class BaseButton:
69      def __init__(self, x, y, image, audio_manager):
70          self.rect = image.get_rect(topleft=(x, y))
71          self.image = image
72          self.audio_manager = audio_manager
73
74      def draw(self, screen):
75          """Draw tombol ke screen"""
76          screen.blit(self.image, self.rect)
```

Gambar 2. 7 Kode Inheritance

Kelas NormalButton dan LockedButton kemudian mewarisi seluruh atribut dan method dari BaseButton. Melalui pewarisan ini, kedua kelas turunan tersebut secara otomatis memiliki kemampuan dasar sebagai tombol tanpa harus menuliskan ulang kode yang sama.

```
87  # POLYMORPHISM: Tombol normal vs tombol terkunci
88  class NormalButton(BaseButton):
89      """Tombol normal yang bisa diklik"""
90      def __init__(self, x, y, image, audio_manager):
91          super().__init__(x, y, image, audio_manager)
92
93      def is_clicked(self):
94          """Override method untuk tombol normal"""
95          return super().is_clicked()
96
```

Gambar 2. 8 Kode Inheritance

```
97  class LockedButton(BaseButton):
98      """Tombol terkunci yang tidak bisa diklik"""
99      def __init__(self, x, y, image, audio_manager):
100         super().__init__(x, y, image, audio_manager)
101         # Buat overlay gelap
102         self.image = create_dark_overlay(image)
103
104     def is_clicked(self):
105         """Override method untuk mencegah klik"""
106         return False  # Selalu return False karena terkunci
107
```

Gambar 2. 9 Kode Inheritance

C. Polymorphism

Polymorphism (Polimorfisme) konsep OOP yang memungkinkan method yang sama digunakan oleh objek yang berbeda dengan perilaku yang berbeda pula. Penerapan konsep Polymorphism (Polimorfisme) pada kode ini bertujuan untuk memberikan perilaku yang berbeda pada objek yang berasal dari kelas induk yang sama, tanpa mengubah cara pemanggilan objek tersebut. Penerapan polymorphism pada kode ini terlihat dari penggunaan Method is_clicked() dengan nama yang sama namun memiliki perilaku berbeda pada kelas turunan.

```
67  # INHERITANCE: Base class untuk semua tombol
68  class BaseButton:
69      def __init__(self, x, y, image, audio_manager):
70          self.rect = image.get_rect(topleft=(x, y))
71          self.image = image
72          self.audio_manager = audio_manager
73
74      def draw(self, screen):
75          """Draw tombol ke screen"""
76          screen.blit(self.image, self.rect)
77
78      def is_clicked(self):
79          """Check jika tombol diklik"""
80          click = pygame.mouse.get_pressed()[0]
81          if self.rect.collidepoint(pygame.mouse.get_pos()) and click:
82              self.audio_manager.play_click()
83              pygame.time.delay(180)
84              return True
85          return False
```

Gambar 2. 10 Kode Polymorphism

Kelas NormalButton dan LockedButton sama-sama mewarisi kelas induk BaseButton, namun masing-masing memiliki implementasi method is_clicked() yang berbeda. Walaupun method yang dipanggil sama, yaitu is_clicked(), perilaku yang dihasilkan bergantung pada jenis objek tombol yang digunakan. Pada NormalButton, method is_clicked() berfungsi untuk mendeteksi apakah tombol benar-benar diklik oleh pemain dan kemudian menjalankan aksi yang sesuai, termasuk memutar efek suara klik. Sebaliknya, pada LockedButton, method is_clicked() dioverride untuk selalu mengembalikan nilai False, sehingga tombol tersebut terlihat di layar tetapi tidak dapat diinteraksi

```
87  # POLYMORPHISM: Tombol normal vs tombol terkunci
88  class NormalButton(BaseButton):
89      """Tombol normal yang bisa diklik"""
90      def __init__(self, x, y, image, audio_manager):
91          super().__init__(x, y, image, audio_manager)
92
93      def is_clicked(self):
94          """Override method untuk tombol normal"""
95          return super().is_clicked()
96
```

Gambar 2. 11 Kode Polymorphism

```python
class LockedButton(BaseButton):
    """Tombol terkunci yang tidak bisa diklik"""
    def __init__(self, x, y, image, audio_manager):
        super().__init__(x, y, image, audio_manager)
        # Buat overlay gelap
        self.image = create_dark_overlay(image)

    def is_clicked(self):
        """Override method untuk mencegah klik"""
        return False  # Selalu return False karena terkunci

```

Gambar 2. 12 Kode Polymorphism

2.5.2 Penggunaan Struktur Data

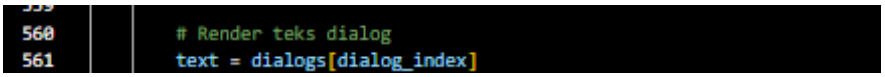
A. Penggunaan List

Penggunaan Struktur data list untuk menyimpan data dialog cerita awal dan dialog setiap level yang bersifat berurutan dan perlu diakses menggunakan indeks.

```python
# DATA DIALOG CERITA
# ========================================
# Dialog utama (pembuka game)
dialogs = [
    "Selamat Datang dalam Arena Perang",
    "Banyak Musuh yang Sedang Menunggu Anda",
    "Selesaikan Misi dan Kalahkan Musuh",
    "Selamat Berperang!"
]

# Dialog per level
dialog_level1 = [
    "Selamat datang di Level 1!",
    "Perkenalkan Saya Jelly",
    "Musuh Anda di Level ini",
    "Mari Kalahkan Saya!!",
    "Semoga Kemenangan Berpihak Pada Anda"
]

dialog_level2 = [
    "Selamat datang di Level 2!",
    "Perkenalkan Saya Mummis",
    "Musuh Anda di Level ini",
    "Mari Kalahkan Saya!!",
    "Semoga Kemenangan Berpihak Pada Anda"
]

dialog_level3 = [
    "Selamat datang di Level 3!",
    "Perkenalkan Saya Ablis",
    "Musuh Anda di Level ini",
    "Mari Kalahkan Saya!!",
    "Semoga Kemenangan Berpihak Pada Anda"
]

```

Gambar 2. 13 Kode Struktur Data (List)

Didukung dengan Algoritma yang digunakan adalah akses indeks linear, di mana sistem menampilkan dialog berdasarkan urutan yang telah ditentukan.



```
560         # Render teks dialog
561         text = dialogs[dialog_index]
```

Gambar 2. 14 Kode Algoritma Pendukung

Penggunaan struktur list of objects digunakan untuk menyimpan kumpulan soal kuis dalam setiap level. Setiap soal direpresentasikan sebagai objek QuizQuestion

```
# DATA QUIZ DENGAN OOP (MODIFIKASI KECIL)
# ========================================
# QuizQuestion untuk setiap soal (OOP)
quiz_questions = {
    1: [
        QuizQuestion(
            "Budi ingin menyapa semua temannya lewat komputer. "
            "Tulislah program yang menampilkan tulisan: Halo, Budi!",
            'print("Halo, Budi!")'
        ),
        QuizQuestion(
            "Di sekolah, bel pagi berbunyi 3 kali untuk memanggil para siswa. "
            "Buat program yang menampilkan tulisan 'Selamat pagi semuanya!' sebanyak 3 kali.",
            'for i in range(3): print("Selamat pagi semuanya!")'
        ),
        QuizQuestion(
            "Doni punya 3 permen, lalu ibunya memberi 5 permen lagi."
            "Buat program untuk menghitung total permen Doni.",
            'hasil = 3 + 5\nprint(hasil)'
        ),
        QuizQuestion(
            "Di taman sekolah ada ubin berbentuk persegi dengan panjang sisi 4 cm."
            "Buat program untuk menghitung luas ubin persegi tersebut.",
            'sisi = 4 luas = sisi * sisi print(luas)'
        ),
        QuizQuestion(
            "Robot mini mempunyai tenaga sebesar angka 4, dan ketika diaktifkan, tenaganya menjadi dua kali lipat."
            "Buat program untuk menampilkan kekuatan robot setelah digandakan.",
            'angka = 4 print(angka * 2)'
        )
    ],
    2: [
        QuizQuestion(
            "Diberikan list angka: [2, 4, 6, 8]."
            "Buat program untuk menjumlahkan semua angkanya.",
            'angka = [2,4,6,8] total = sum(angka) print(total)'
        ),
        QuizQuestion(
            "Jika jumlah = 4, tampilkan kata 'Belajar' sebanyak 4 kali.",
            'jumlah = 4 for i in range(jumlah): print("Belajar")'
        ),
        QuizQuestion(
            "Ubah variabel warna 'merah' menjadi 'biru' lalu tampilkan",
            'warna = "merah" warna = "biru" print(warna)'
        ),
        QuizQuestion(
            "Hitung berapa huruf 'a' dalam kata 'cerita'",
            'print("cerita".count("a"))'
        )
    ],
    3: [
        QuizQuestion(
            "Budi ingin memperkenalkan dirinya kepada teman-teman di kelas."
            "Buatlah program Python sederhana yang menampilkan perkenalan Budi di layar."
            "'Perkenalkan aku Budi,umur aku 5 thn'",
            'nama = "Budi" umur = 5 print(f"Perkenalkan aku {nama},umur aku {umur} thn")'
        )
    ]
}
```

Gambar 2. 15 Kode Struktur Data(List)

didukung dengan algoritma iterasi berbasis indeks yang memastikan pemain menyelesaikan soal satu per satu.

```
655         # Ambil soal dari QuizQuestion object
656         current_question = quiz_questions[level][quiz_number]
```

Gambar 2. 16 Kode Algoritma Pendukung

2.5.3 Implementasi Interaksi

A. Interaksi Mouse

Game ini menggunakan tombol berbasis gambar yang dapat diklik dengan mouse untuk berpindah halaman dan melakukan aksi tertentu. Tombol-tombol tersebut dibungkus dalam class OOP BaseButton dan turunannya.

```
67  # INHERITANCE: Base class untuk semua tombol
68  class BaseButton:
69      def __init__(self, x, y, image, audio_manager):
70          self.rect = image.get_rect(topleft=(x, y))
71          self.image = image
72          self.audio_manager = audio_manager
73
74      def draw(self, screen):
75          """Draw tombol ke screen"""
76          screen.blit(self.image, self.rect)
77
78      def is_clicked(self):
79          """Check jika tombol diklik"""
80          click = pygame.mouse.get_pressed()[0]
81          if self.rect.collidepoint(pygame.mouse.get_pos()) and click:
82              self.audio_manager.play_click()
83              pygame.time.delay(180)
84              return True
85          return False
```

Gambar 2. 17 Kode Implementasi Interaksi

B. Interaksi Keyboard

Game ini menggunakan interaksi keyboard pada halaman kuis, pemain dapat mengetik jawaban langsung menggunakan keyboard. Sistem menangani input karakter, penghapusan, dan pengiriman jawaban dengan tombol Enter.

```
488             # Hapus karakter dengan backspace
489             if event.key == pygame.K_BACKSPACE:
490                 input_text = input_text[:-1]
491
492             # Submit jawaban dengan Enter
493             elif event.key == pygame.K_RETURN:
494                 # Gunakan QuizQuestion untuk cek jawaban
495                 if quiz_number < len(quiz_questions[level]):
496                     current_question = quiz_questions[level][quiz_number]
497
```

Gambar 2. 18 Kode Implementasi Interaksi

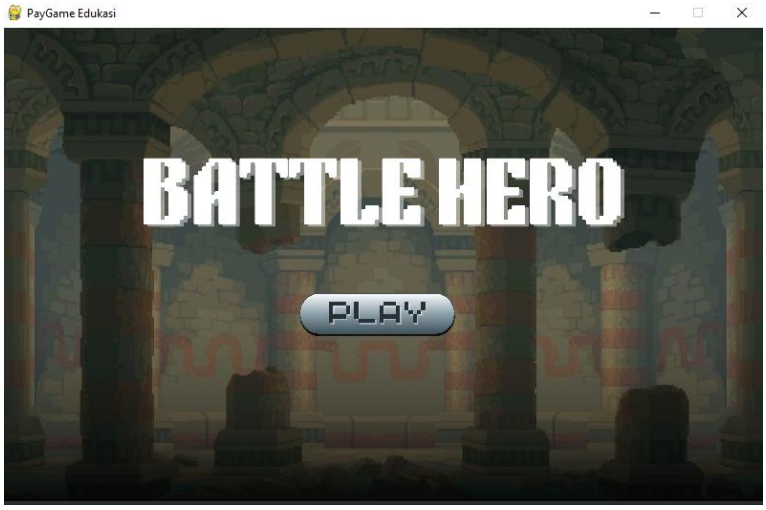
BAB III
DESKRIPSI APLIKASI

3.1 Tampilan Aplikasi

Berikut adalah tampilan dari aplikasi game sederhana menggunakan PayGame

a. Tampilan Menu Utama

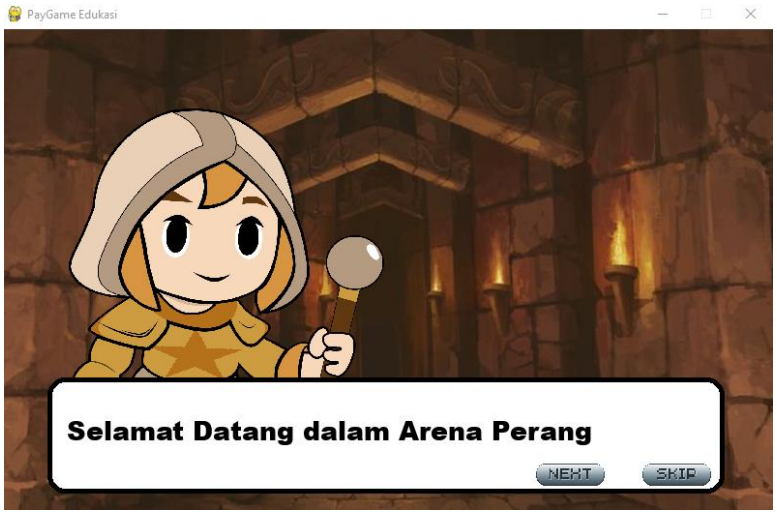
Tampilan menu utama game, terdapat judul game yaitu “Battle Hero” dan button tombol play untuk menjalankan permainan.



Gambar 3. 1 Tampilan Game (Menu Utama)

b. Tampilan Dialog Karakter Utama/Pembuka

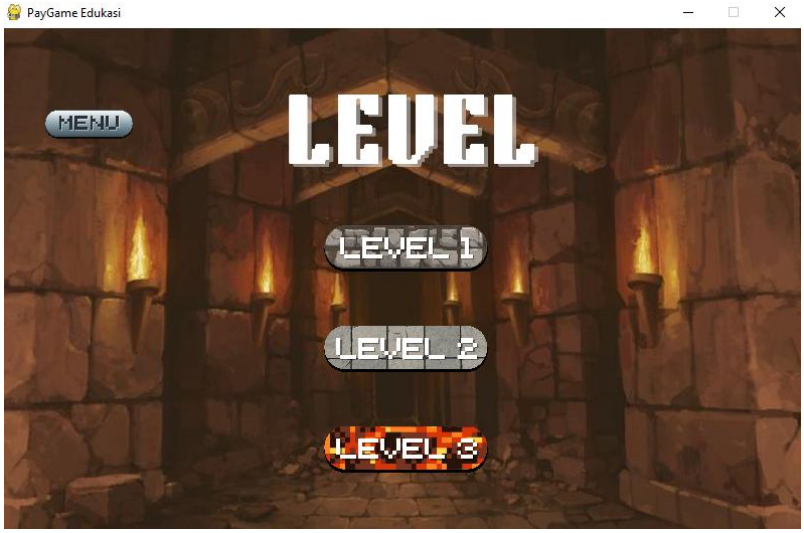
Tampilan Dialog Karakter Utama sekaligus menjadi pembuka game, terdapat gambar karakter utama.



Gambar 3. 2 Tampilan Game (Dialog Karakter Utama)

c. Tampilan Pilihan Level

Tampilan pilihan level game yang terdiri dari level 1,2, dan 3



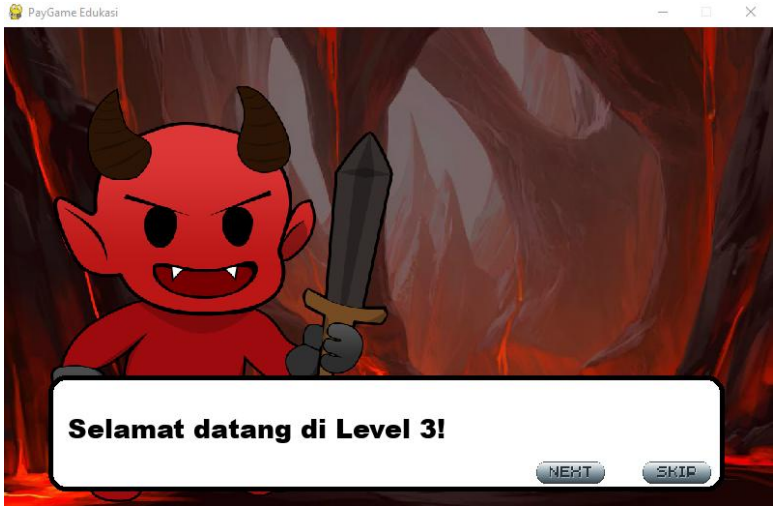
Gambar 3. 3 Tampilan Game (Level Selection)

d. Tampilan Dialog Karakter Setiap Level

Tampilan dialog karakter setiap level. Sebelum memulai permainan pada setiap level akan muncul dialog karakter sesuai dengan level yang ditentukan.

Gambar 3. 4 Tampilan Game (Dialog Level1)

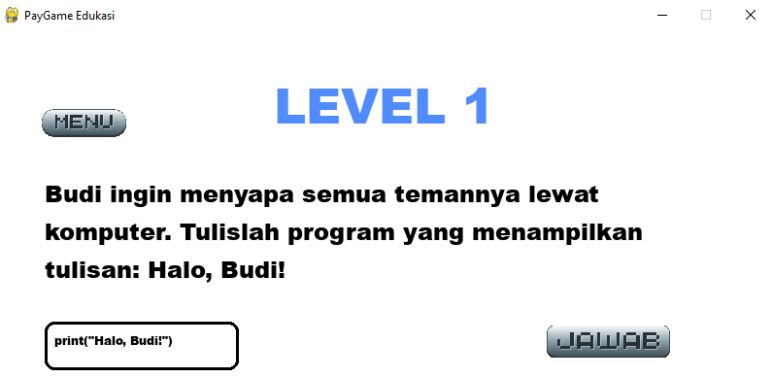
Gambar 3. 5 Tampilan Game (Dialog Level2)



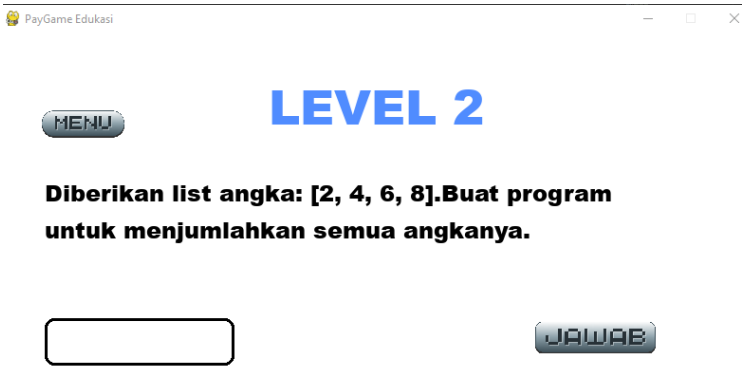
Gambar 3. 6 Tampilan Game (Dialog Level3)

e. Tampilan Quiz Setiap Level

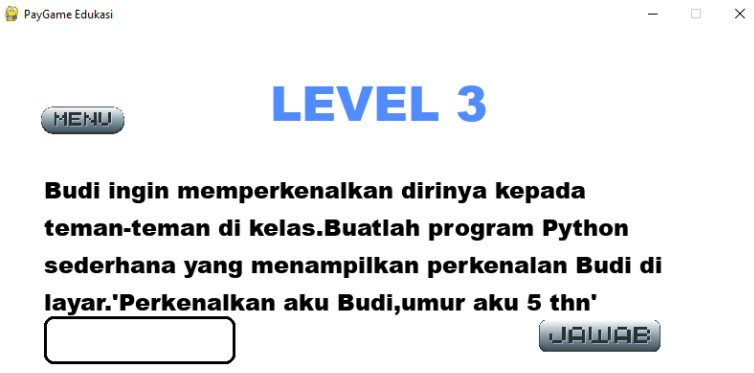
Tampilan quiz atau misi yang harus dijawab di setiap levelnya



Gambar 3. 7 Tampilan Game (Quiz Level1)



Gambar 3. 8 Tampilan Game (Quiz Level2)



Gambar 3. 9 Tampilan Game (Quiz Level3)

f. Tampilan Winner

Tampilan pemenang, apabila pengguna telah menyelesaikan semua level maka akan muncul ucapan selamat.

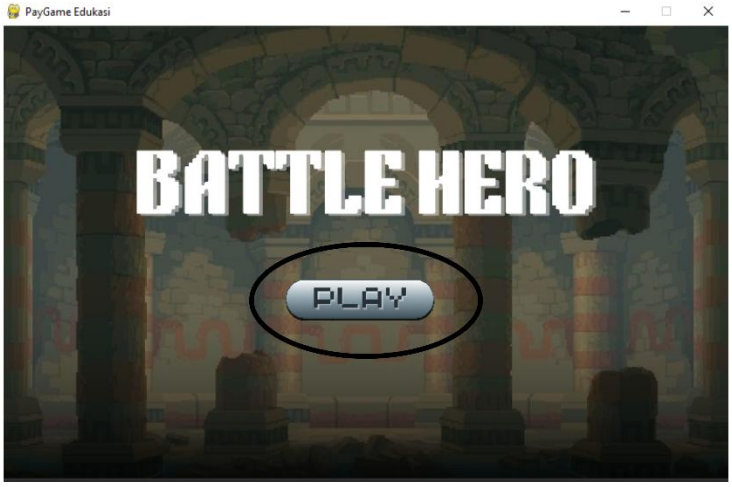
Gambar 3. 10 Tampilan Game (Winner)

3.2 Fitur Aplikasi

Berikut adalah fitur – fitur yang ada di dalam game

a. Menu Utama

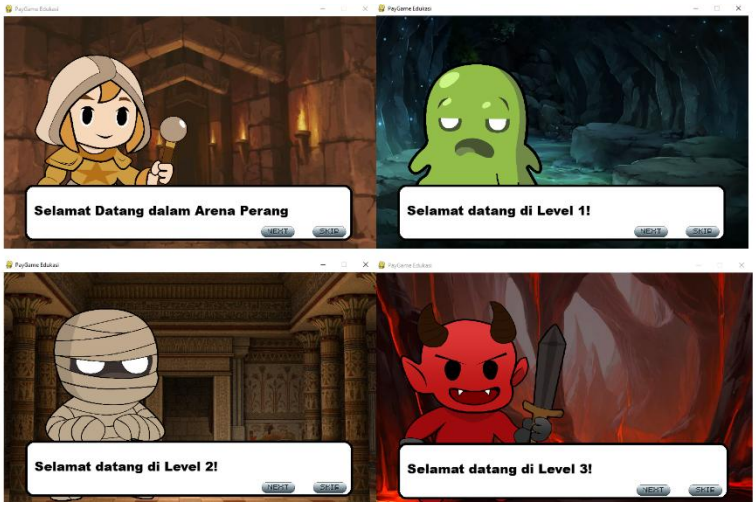
Didalam menu utama terdapat button play untuk memulai permainan. Apabila menekan tombol play audio atau music dalam game otomatis akan menyala



Gambar 3. 11 Fitur Menu Utama

b. Dialog Karakter

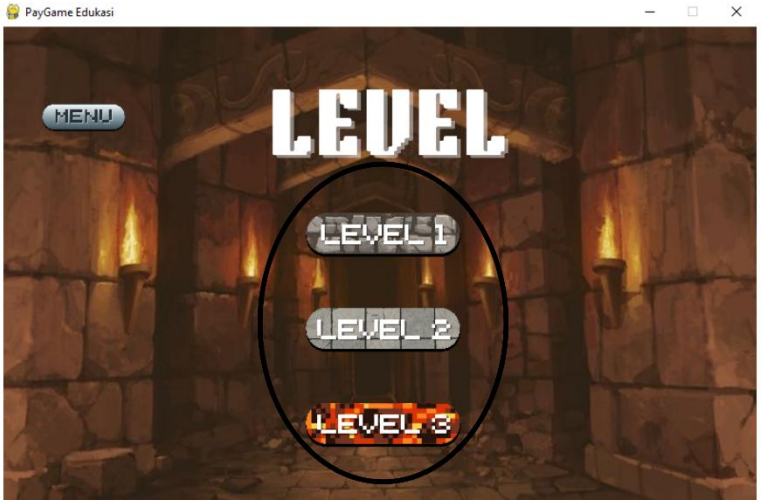
Didalam game terdapat dialog karakter untuk interaksi kepada pengguna game. dialog karakter selalu berada di awal, sebelum memilih level dan berada disetiap level.



Gambar 3. 12 Fitur Dialog

c. Pemilihan Level

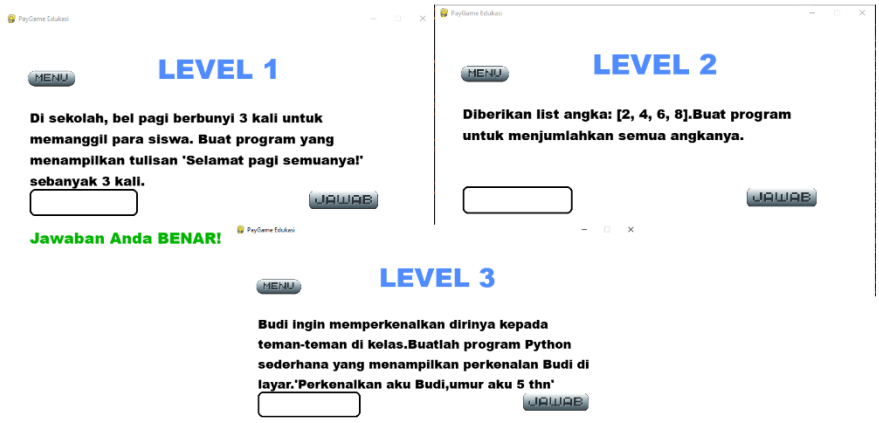
Terdapat pemilihan level dimana pengguna harus menyelesaikan level pertama terlebih dahulu agar bisa lanjut dan menyelesaikan level berikutnya



Gambar 3. 13 Fitur Level Selection

d. Quiz

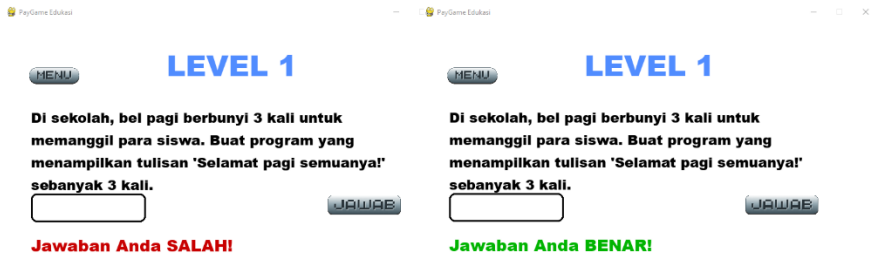
Terdapat quiz disetiap level, pengguna harus menyelesaikan quiz disetiap level agar menjadi pemenang. Dan inti dari game ini adalah menyelesaikan quiz pada setiap level



Gambar 3. 14 Fitur Quiz

e. Notifikasi Benar & Salah pada Quiz

Disetiap quiz terdapat notifikasi benar dan salah, apabila jawaban salah maka akan muncul notifikasi "jawaban anda salah" dan soal akan tetap sama sampai jawaban benar. Apabila jawaban benar maka akan muncul notifikasi "jawaban anda benar" dan soal akan otomatis berubah ke soal berikutnya.



Gambar 3. 15 Fitur Notifikasi

f. Notifikasi Pemenang

Apabila pengguna sudah menyelesaikan semua level maka akan otomatis muncul ucapan selamat selama 10 detik, setelah itu akan otomatis kembali ke menu utama

Gambar 3. 16 Fitur Ucapan

3.3 Desain Asset dan Audio

Berikut adalah asset yang digunakan dalam game

A. Desain Karakter

Gambar 3. 17 Desain Karakter

B. Desain Latar Tempat (Baground)

Gambar 3. 18 Desain Latar Tempat

C. Desain Button

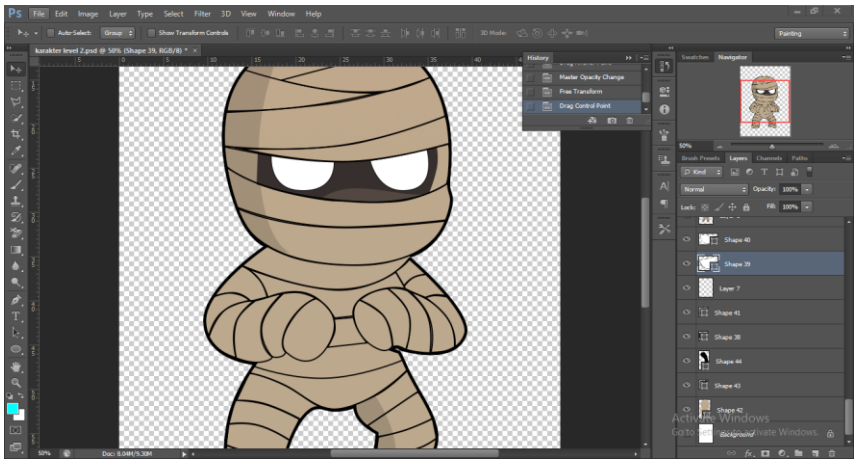
Gambar 3. 19 Desain Button

3.4 Tools dan Teknologi

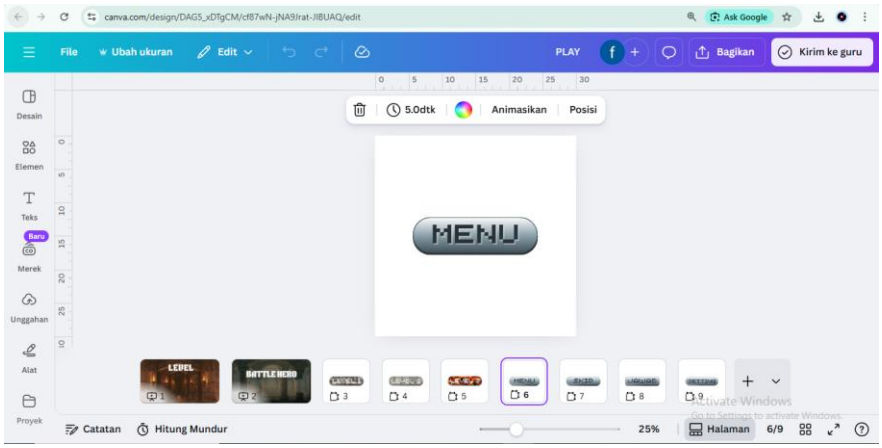
Berikut adalah tools yang digunakan untuk membuat dan merancang game

a. PhotoShop & Canva

Membuat desain asset karakter, desain button, latar tempat atau baground menggunakan tools PhotoShop dan Canva



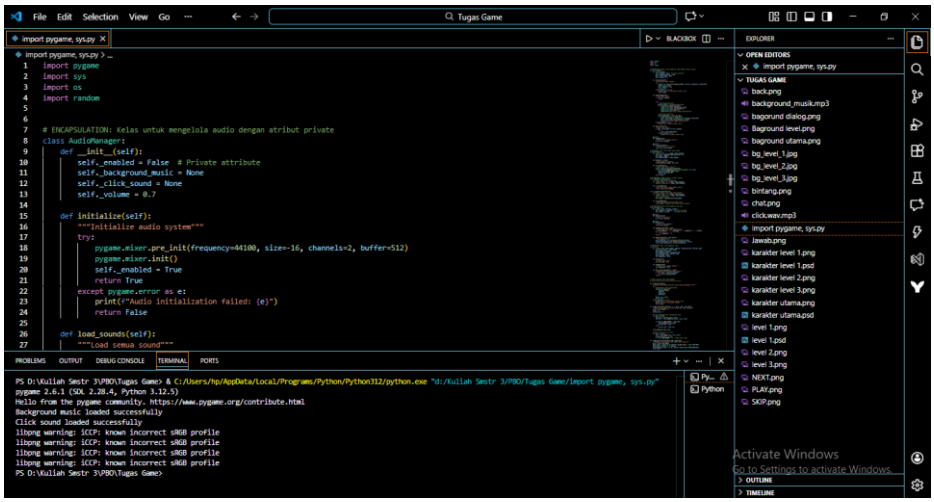
Gambar 3. 20 Tools Photoshop



Gambar 3. 21 Tools Canva

b. VsCode & Phython

Menggunakan VsCode sebagai tempat untuk implementasi kode paygame. Dan menggunakan bahasa pemrograman phyton untuk pendukung jalannya sistem pada game.



Gambar 3. 22 Tools VSCode

BAB IV
PENUTUP

4.1 Kesimpulan

Pengembangan game Battle Hero merupakan upaya untuk menghadirkan media pembelajaran alternatif yang relevan dengan kebutuhan siswa Sekolah Dasar dalam mengenal dasar-dasar cara berpikir pemrograman. Permasalahan pembelajaran coding yang cenderung abstrak dan kurang menarik diatasi melalui pendekatan game edukasi yang bersifat interaktif, kontekstual, dan menyenangkan. Game Battle Hero dirancang menggunakan pustaka Pygame dengan alur permainan yang sederhana namun terstruktur, mulai dari inisialisasi sistem, menu utama, cerita pembuka, pemilihan level, dialog level, tahap kuis, hingga tampilan kemenangan. Alur ini membantu siswa memahami konsep urutan instruksi, logika sebab–akibat, aturan atau kondisi sederhana, serta interaksi input dan output melalui pengalaman bermain secara langsung.

Dalam sisi pengembangan, penerapan konsep Object-Oriented Programming (OOP) digunakan sebagai pendekatan perancangan sistem untuk membangun struktur program yang terorganisasi, mudah dipahami, dan mudah dikembangkan. Namun demikian, OOP tidak diajarkan secara formal kepada siswa, melainkan dimanfaatkan oleh pengembang untuk memastikan kualitas dan keberlanjutan aplikasi. Materi yang disajikan dalam game juga dibatasi agar sesuai dengan tingkat kognitif siswa Sekolah Dasar, tanpa menekankan pada sintaks atau aspek teknis pemrograman yang kompleks.

Dengan demikian, game Battle Hero diharapkan dapat menjadi media pembelajaran pendukung yang efektif dalam mengenalkan dasar-dasar berpikir komputasional kepada siswa Sekolah Dasar. Melalui pendekatan permainan, proses belajar menjadi lebih menarik, bermakna, dan mampu meningkatkan minat serta pemahaman siswa terhadap konsep dasar coding secara sederhana dan menyenangkan.

LinkGithub:
https://github.com/nabila060126/Battle-Hero-Game-Edukasi-Sederhana-Paygame-Berbasis-OOP-