

LAPORAN PEMROGRAMAN BERORIENTASI OBJEK

Game 2D “ Rainbow Riddle”



Mata Kuliah:

Pemrograman Berorientasi Objek

Dosen Pengampu:

I Gde Agung Sri Sidhimantra, S.Kom., M.Kom.

Binti Kholifah, S.Kom., M.Tr.Kom.

Moch Deny Pratama, S.Tr.Kom.,M.Kom

Disusun Oleh | Kelompok 11 2023E

1. Regha Rahmadian Bintang (23091397156)
2. Refany Dwi Lestari (23091397170)
3. Vergi Mutia Rahmasyani (23091397171)

Program Studi D4 Manajemen Informatika

Fakultas Vokasi

Universitas Negeri Surabaya

2024

DAFTAR ISI

| | |
|--|----|
| DAFTAR ISI..... | 2 |
| BAB I..... | 4 |
| Pendahuluan..... | 4 |
| 1.1 Latar Belakang..... | 4 |
| 1.2 Tujuan..... | 4 |
| BAB II..... | 5 |
| Analisa dan Perancangan Sistem | 5 |
| 2.1 Analisis Kebutuhan..... | 5 |
| 1) Kebutuhan Fungsional | 5 |
| 2) Kebutuhan Non-Fungsional..... | 6 |
| 3) Kebutuhan Perangkat Keras | 6 |
| 4) Kebutuhan Perangkat Lunak..... | 7 |
| 2.2 Konsep Game..... | 7 |
| 2.2.1 Nama Game | 7 |
| 2.2.2 Genre Game | 7 |
| 2.2.3 Deskripsi Umum | 7 |
| 2.2.4 Konsep Inti..... | 8 |
| 2.2.5 Fitur Utama | 8 |
| 2.2.5 Visual dan Estetika..... | 8 |
| 2.3 Perancangan Objek dan Class Diagram..... | 9 |
| 2.3.1 Deskripsi Kelas dan Hubungan Antar Kelas..... | 9 |
| BAB III | 11 |
| Implementasi dan Penjelasan Kode | 11 |
| 3.1 Penggunaan Pygame dan Alasan Pemilihan | 11 |
| try dan except pygame.error as e: | 13 |
| • pygame.display.flip() Fungsi: Mengupdate layar (display) dengan semua perubahan yang telah dibuat pada frame saat ini..... | 29 |
| • pygame.quit() Fungsi: Menutup aplikasi Pygame dengan benar. | 29 |
| 3.3 Implementasi OOP Pada Game..... | 30 |
| 3.3.1 Encapsulation..... | 30 |
| 3.3.2 Abstraction..... | 35 |
| 3.3.3 Polymorphism | 36 |
| BAB IV | 38 |

| | |
|---|----|
| Hasil dan Pembahasan | 38 |
| 4.1 Hasil Pengembangan Game | 38 |
| 4.1.1 Screenshot dan Demonstrasi Fitur | 38 |
| 4.2 Cara Bermain | 40 |
| 4.2.1 Aturan Permainan: | 40 |
| 4.2.2 Strategi Bermain: | 40 |
| 4.2.3 Kondisi: | 40 |
| 4.2 Evaluasi Gameplay | 40 |
| 4.2.1 Analisis Kelebihan dan Kekurangan | 40 |
| 4.2.2 Evaluasi Keberlanjutan dan Pengembangan | 41 |
| BAB V | 43 |
| Kesimpulan dan Saran | 43 |
| 5.1 Kesimpulan | 43 |
| 5.2 Saran | 44 |
| DAFTAR PUSTAKA | 45 |

BAB I

Pendahuluan

1.1 Latar Belakang

Seiring dengan berkembangnya teknologi, permainan berbasis digital semakin diminati oleh berbagai kalangan, baik untuk hiburan maupun sebagai alat untuk melatih kemampuan berpikir. Salah satu genre yang cukup populer adalah game puzzle, yang menawarkan tantangan logika dan pemecahan masalah. Game jenis ini mampu mengasah keterampilan kognitif pemain, seperti perencanaan, analisis, dan pengambilan keputusan.

Puzzle Game 2D "Rainbow Riddle" hadir sebagai salah satu solusi hiburan yang juga memiliki nilai edukatif. Dengan mekanika permainan yang sederhana namun menantang, game ini memberikan pengalaman bermain yang seru sekaligus melatih logika dan strategi pemain. Pemain ditantang untuk menyortir cairan berwarna dalam botol dengan aturan tertentu, sehingga setiap botol hanya berisi satu warna. Tingkat kesulitan yang meningkat di setiap level memberikan tantangan berkelanjutan yang menarik.

Pengembangan game ini dilakukan menggunakan Python dengan pendekatan Pemrograman Berorientasi Objek (PBO). Pendekatan ini memungkinkan pengelolaan elemen-elemen permainan, seperti botol, cairan, dan logika permainan, secara modular dan terstruktur. Dengan demikian, pengembangan dan pemeliharaan game menjadi lebih efisien. Melalui Puzzle Game 2D "Rainbow Riddle", diharapkan pemain tidak hanya mendapatkan hiburan, tetapi juga mampu meningkatkan kemampuan berpikir logis dan strategis. Game ini juga menjadi salah satu implementasi nyata penerapan konsep PBO dalam dunia pengembangan perangkat lunak.

1.2 Tujuan

Permainan ini dirancang untuk melatih kemampuan pemain dalam berpikir logis dan strategis.

1. Game ini tidak hanya berfungsi sebagai hiburan, tetapi juga sebagai alat edukasi yang dapat meningkatkan keterampilan pemecahan masalah pemain melalui pengalaman bermain yang menyenangkan dan interaktif.
2. Dengan menggunakan pendekatan PBO dalam pengembangan, game ini bertujuan menjadi contoh implementasi nyata yang membantu pengembang memahami konsep modularitas dan pengelolaan komponen secara terstruktur dalam pemrograman.
3. Game ini bertujuan untuk memaksimalkan potensi Python dalam pengembangan game 2D sederhana dengan grafik menarik dan gameplay yang intuitif.

BAB II

Analisa dan Perancangan Sistem

2.1 Analisis Kebutuhan

Analisis kebutuhan untuk pembuatan game Riddle Rainbow. Kebutuhan ini dibagi menjadi beberapa aspek utama: kebutuhan fungsional, kebutuhan non-fungsional, kebutuhan perangkat keras, kebutuhan perangkat lunak, dan kebutuhan sumber daya manusia.

1) Kebutuhan Fungsional

Kebutuhan ini mencakup fungsi utama yang harus ada di dalam game:

| Fitur | Penjelasan |
|---------------------|---|
| Gameplay | <ol style="list-style-type: none">1. Pemain dapat memilih dan memindahkan cairan warna dari satu tabung ke tabung lainnya.2. Game memeriksa kondisi kemenangan saat semua tabung diatur dengan warna yang seragam.3. Game mendeteksi kondisi kekalahan jika pemain kehabisan langkah yang memungkinkan.4. Skor akan bertambah setiap kali pemain melakukan langkah yang benar. |
| Suara dan Efek | <ol style="list-style-type: none">1. Suara saat cairan dipindahkan antar tabung.2. Suara kemenangan saat pemain menang.3. Suara kekalahan saat pemain kalah |
| Level dan Kesulitan | <ol style="list-style-type: none">1. Sistem level generator untuk menghasilkan kombinasi tabung dan warna secara acak pada setiap permainan baru.2. Peningkatan kesulitan seiring berjalannya waktu (misalnya, jumlah tabung atau variasi warna bertambah). |
| Kontrol | <ol style="list-style-type: none">1. Pemain menggunakan mouse untuk memilih dan memindahkan cairan warna.2. Keyboard digunakan untuk fitur:3. Space untuk mengulang permainan.4. Enter untuk memulai permainan baru. |

2) Kebutuhan Non-Fungsional

Kebutuhan yang berkaitan dengan kualitas game:

| Aspek | Penjelasan |
|--------------|---|
| Performa | Game harus berjalan dengan FPS stabil (minimal 30 FPS) untuk memberikan pengalaman bermain yang lancar. |
| Usability | Game harus memiliki antarmuka yang intuitif dan mudah dipahami oleh pengguna. Tombol dan elemen interaktif harus jelas dan mudah diakses. |
| Portabilitas | Game harus dapat berjalan di berbagai sistem operasi desktop, seperti: <ol style="list-style-type: none">1. Windows2. macOS3. Linux (jika memungkinkan) |
| Skalabilitas | Mudah menambahkan fitur baru seperti mode permainan tambahan seperti mode timer atau tantangan harian |

3) Kebutuhan Perangkat Keras

Komponen perangkat keras minimum dan rekomendasi untuk menjalankan game ini:

Minimum

- Prosesor: Dual-core 1.8 GHz atau setara.
- RAM: 2 GB.
- Penyimpanan: 100 MB ruang kosong.
- GPU: Intel HD Graphics atau setara.
- Resolusi Layar: 1280x720.

Rekomendasi

- Prosesor: Quad-core 2.5 GHz atau lebih tinggi.
- RAM: 4 GB.
- Penyimpanan: 200 MB ruang kosong.
- GPU: Dedicated GPU seperti NVIDIA GeForce GTX 750 atau lebih tinggi.
- Resolusi Layar: 1920x1080.

4) Kebutuhan Perangkat Lunak

Komponen perangkat lunak yang diperlukan untuk pengembangan dan eksekusi:

a. Untuk Pengembang

1. Python versi 3.8 atau lebih baru.
2. Library:
 - *pygame* untuk rendering grafis, suara, dan kontrol.
 - *copy* untuk manipulasi struktur data.
 - *random* untuk generasi elemen permainan secara acak.
3. Text Editor/IDE:
 - Visual Studio Code, PyCharm, atau editor teks lainnya.
 -

b. Untuk Pengguna

Sistem operasi yang mendukung:

- Windows 7/8/10/11.
- macOS Mojave atau lebih baru.
- Distribusi Linux (Ubuntu 20.04 atau setara).
- File runtime Python (jika game tidak packaged).

2.2 Konsep Game

2.2.1 Nama Game

Rainbow Riddle : Game teka-teki warna yang menggabungkan logika, strategi, dan keseruan memindahkan cairan

2.2.2 Genre Game

Puzzle dan Strategi : Pemain ditantang untuk menyelesaikan teka-teki dengan aturan tertentu, mengasah kemampuan berpikir logis dan pengambilan keputusan.

2.2.3 Deskripsi Umum

Puzzle Game 2D Rainbow Riddle adalah permainan yang menggabungkan elemen logika dan pemecahan masalah sederhana. Dalam game ini, pemain dihadapkan dengan beberapa botol atau tabung transparan yang berisi cairan berwarna-warni. Tujuan permainan adalah menyortir cairan tersebut sehingga setiap botol hanya berisi satu warna.

Permainan memiliki aturan sederhana: air hanya dapat dituangkan ke botol kosong atau ke botol dengan warna yang sama di bagian atasnya. Tantangan dalam permainan ini terletak pada perencanaan langkah yang tepat agar semua warna dapat tersusun dengan benar tanpa menemui jalan buntu. Setiap level memiliki tingkat kesulitan yang meningkat, sehingga memberikan pengalaman bermain yang semakin menarik.

Game ini dikembangkan menggunakan bahasa Python dengan pendekatan Pemrograman Berorientasi Objek (PBO). Konsep ini memungkinkan setiap elemen permainan, seperti botol, air, dan logika permainan, dikelola secara modular sehingga memudahkan pengembangan dan pemeliharaan. Puzzle Game 2D Rainbow Riddle tidak hanya menawarkan hiburan, tetapi juga melatih logika dan kemampuan pemain dalam menyusun strategi.

2.2.4 Konsep Inti

- **Gameplay:**
Pemain diberikan sejumlah tabung berisi cairan warna yang acak. Tugas mereka adalah memindahkan cairan antar tabung sehingga setiap tabung hanya berisi satu jenis warna.
- **Aturan Permainan:**
 1. Pemain hanya bisa menuangkan cairan ke tabung yang:
 - Memiliki ruang kosong.
 - Memiliki cairan di atasnya dengan warna yang sama.
 2. Jika tidak ada langkah yang memungkinkan, permainan berakhir dengan kekalahan.
- **Kemenangan:**
Pemain menang jika semua tabung diisi dengan satu warna yang seragam atau kosong.

2.2.5 Fitur Utama

- 1) **Level Generator:**
 - Level dibuat secara acak setiap permainan, memberikan pengalaman bermain yang selalu baru.
 - Tingkat kesulitan meningkat seiring dengan bertambahnya jumlah tabung dan variasi warna.
- 2) **Sistem Skor:**
 - Pemain mendapatkan skor berdasarkan jumlah langkah yang efisien.
 - Skor lebih tinggi jika cairan dipindahkan ke tabung yang tepat dengan lebih sedikit langkah.
- 3) **Visual Interaktif:**
 - Warna cairan yang cerah dan menarik.
 - Highlight (warna hijau) pada tabung yang dipilih.
- 4) **Suara dan Efek:**
 - Efek suara cairan saat menuangkan.
 - Efek suara kemenangan dan kekalahan untuk meningkatkan pengalaman bermain.
- 5) **Mode Pemula dan Ahli:**
 - Pemula: Jumlah tabung lebih sedikit (4-5 tabung dengan 3 warna).
 - Ahli: Jumlah tabung lebih banyak (8-10 tabung dengan 5 warna).

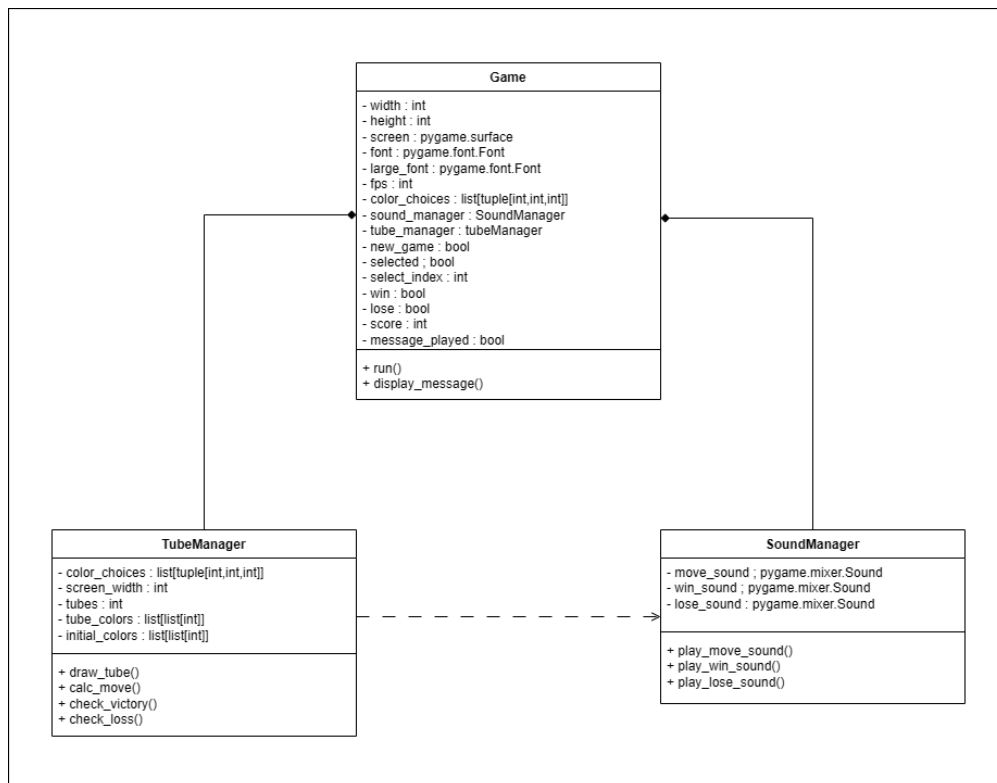
2.2.5 Visual dan Estetika

- 1) **Desain Sederhana dan Modern:**
 - Warna-warna cerah untuk cairan.

- Tabung dibuat dengan bentuk kotak sederhana dengan efek *glossy*.
 - Latar belakang gelap agar warna cairan lebih menonjol.
- 2) **Animasi Cairan:**
- Animasi saat cairan dituang dari satu tabung ke tabung lainnya.
- 3) **Antarmuka Bersih:**
- Elemen antarmuka (UI) minimalis dengan informasi penting saja, seperti skor, tombol restart, dan status permainan.

2.3 Perancangan Objek dan Class Diagram

Pada tahap perancangan perangkat lunak, diagram kelas digunakan untuk merepresentasikan hubungan antara kelas-kelas yang dirancang. Diagram ini memberikan gambaran struktur sistem yang mencakup atribut, metode, dan hubungan antar kelas. Gambar.1 berikut ini menunjukkan class diagram untuk perangkat lunak yang telah dirancang:



Gambar.1 Class Diagram

2.3.1 Deskripsi Kelas dan Hubungan Antar Kelas

1. Game

Kelas Game adalah kelas utama yang bertanggung jawab untuk menjalankan logika permainan. Kelas ini memiliki atribut dan metode untuk menangani

elemen permainan, seperti pengaturan layar, pengaturan font, dan loop utama permainan. Selain itu, kelas ini juga memiliki komposisi dengan kelas SoundManager dan TubeManager, yang masing-masing bertanggung jawab untuk pengelolaan suara dan tabung warna.

2. TubeManager

Kelas TubeManager bertanggung jawab untuk mengelola logika permainan terkait tabung warna. Atributnya meliputi daftar warna dan tata letak tabung, sementara metodenya mencakup pengaturan ulang tata letak tabung, perhitungan gerakan, dan pengecekan kondisi kemenangan atau kekalahan. Hubungan antara Game dan TubeManager juga merupakan composition, karena Game memiliki kendali penuh atas TubeManager.

3. SoundManager

Kelas TubeManager memiliki hubungan dependency dengan kelas SoundManager, karena TubeManager memanfaatkan objek SoundManager untuk memutar suara ketika terjadi gerakan. Hubungan ini bersifat sementara dan tidak permanen, karena TubeManager tidak menyimpan referensi ke SoundManager dalam atributnya.

BAB III

Implementasi dan Penjelasan Kode

3.1 Penggunaan Pygame dan Alasan Pemilihan

Penggunaan Pygame pada pembuatan game *Rainbow Riddle* memiliki beberapa alasan kuat yang mendukung efektivitas dan efisiensi dalam pengembangan game tersebut:

1. Kemudahan Penggunaan : Pygame adalah pustaka Python yang dirancang untuk membuat game dengan cara yang mudah dan cepat. Pygame memiliki API yang sederhana, sehingga cocok bagi pengembang pemula yang ingin membuat game.
2. Grafik 2D yang Efisien : Game *Riddle Rainbow* berfokus pada grafik 2D, yang merupakan kelebihan dari Pygame. Pygame memungkinkan pembuatan dan manipulasi objek grafis dengan mudah, seperti menggambar tabung dan cairan dengan warna berbeda.
3. Pengelolaan Input Pengguna : Input pengguna adalah bagian penting dari game ini, karena pemain berinteraksi dengan permainan menggunakan mouse dan keyboard. Pygame menyediakan berbagai cara untuk menangani input, termasuk deteksi klik mouse, pergerakan kursor, dan input tombol pada keyboard (misalnya, tombol *space* untuk memulai ulang permainan dan *enter* untuk memulai permainan baru).
4. Kemampuan untuk Mengelola Suara : Game ini memanfaatkan efek suara untuk meningkatkan pengalaman bermain (misalnya, suara saat cairan dituangkan atau suara kemenangan/kekalahan). Pygame memiliki modul *pygame.mixer* yang memungkinkan pengelolaan suara, termasuk memainkan file audio dalam format yang berbeda. Ini sangat membantu dalam memberikan umpan balik audio yang meningkatkan immersi pemain.
5. Kompatibilitas dan Portabilitas : Pygame mendukung berbagai platform, termasuk Windows, macOS, dan Linux. Game yang dikembangkan dengan Pygame dapat dijalankan di berbagai sistem operasi tanpa perlu modifikasi besar. Ini penting jika game ingin dijalankan di banyak perangkat dan sistem.

Pygame memungkinkan pengembang untuk fokus pada desain dan mekanika permainan tanpa terjebak dalam detail teknis yang rumit. Dengan kemudahan penggunaan, pengelolaan grafis dan suara yang efisien, serta dukungan untuk input pengguna dan waktu, Pygame memberikan fondasi yang solid untuk membuat game puzzle yang menyenangkan dan interaktif.

3.2 Struktur Code

Berikut merupakan penjelasan dari struktur code yang digunakan untuk membuat Game 2D “Rainbow Riddle” :

1) Persiapan Library



Penjelasan code:

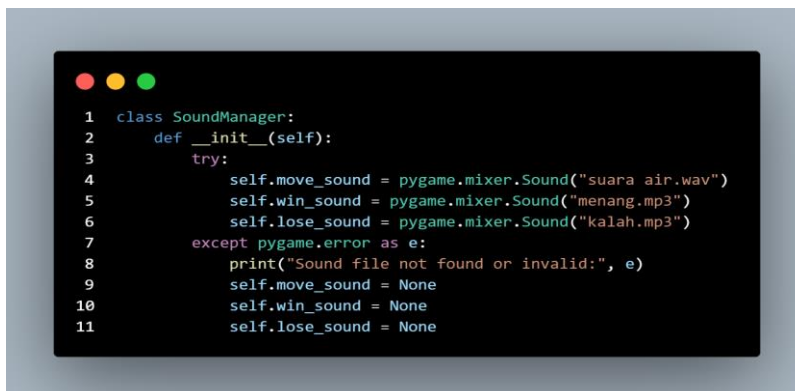
Kode diatas adalah bagian persiapan awal untuk membuat game Rainbow Riddle.

Dengan mengimpor pustaka `pygame`, `random`, dan `copy`, Anda memiliki alat untuk:

- Mengatur elemen game secara dinamis (dengan `random` dan `copy`).
- Menampilkan grafik, suara, dan interaksi (dengan `pygame`).

Setelah `pygame.init()` dipanggil, kita bisa mulai membuat jendela game, menggambar objek, dan menangani input dari pemain.

2) Pembuatan Class SoundManager



Penjelasan code:

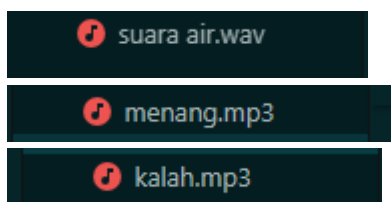
Class `SoundManager`. Kelas ini dirancang untuk mengelola efek suara dalam game, seperti suara saat pemain melakukan gerakan, menang, atau kalah.

- `def __init__(self)` Metode dari konstruktor (initializer) ini dipanggil saat objek dari kelas `SoundManager` dibuat.

`try` dan `except pygame.error as e`:

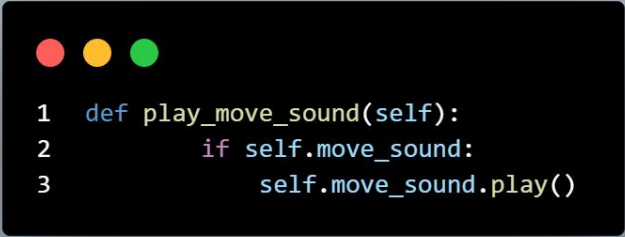
- Blok `try` digunakan untuk mencoba memuat file suara dengan menggunakan `pygame.mixer.Sound()`.
 - `self.move_sound`: Suara yang diputar saat pemain melakukan gerakan.
 - `self.win_sound`: Suara yang diputar saat pemain menang.
 - `self.lose_sound`: Suara yang diputar saat pemain kalah.
 - Jika terjadi kesalahan (misalnya, file tidak ditemukan atau format file tidak didukung), blok `except` akan menangkap kesalahan dan menampilkan pesan error di terminal.
- Menggunakan pustaka `pygame.mixer` untuk memuat dan memutar file. `pygame.mixer.Sound("nama_file")` digunakan untuk memuat file suara ke dalam program. File ini harus berupa file suara yang didukung oleh `pygame`, seperti `.wav` atau `.mp3`.

3) Penambahan Objek Suara



- `suara air.wav` : objek suara yang diputar saat pemain melakukan gerakan pemindahan warna ke tabung yang berisikan warna selaras.
- `menang.mp3` : objek suara yang diputar saat pemain menang atau menyelesaikan permainan dengan sempurna.
- `kalah.mp3` : objek suara yang diputar saat pemain kalah.

4) Pembuatan `def play_move_sound(self)`



```

1  def play_move_sound(self):
2      if self.move_sound:
3          self.move_sound.play()

```

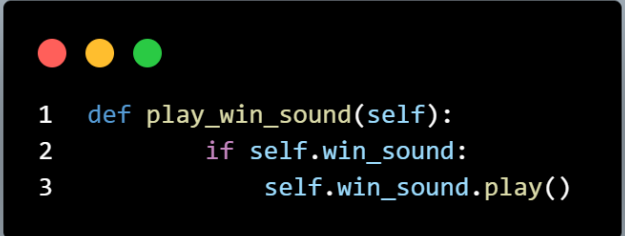
Penjelasan code:

- **def play_move_sound(self):** Deklarasi metode. Metode ini adalah bagian dari kelas **SoundManager** dan memerlukan **self** sebagai parameter untuk mengakses atribut atau metode lain dalam kelas.

if self.move_sound:

- Mengecek apakah atribut **self.move_sound** tidak bernilai **None**.
- Jika **self.move_sound** memiliki nilai (artinya file suara untuk gerakan telah berhasil dimuat), maka program akan melanjutkan ke baris berikutnya.
 - **self.move_sound.play()** Memutar efek suara yang tersimpan di **self.move_sound** menggunakan metode **play()** dari objek **pygame.mixer.Sound**.

5) Pembuatan def play_win_sound(self)



```

1  def play_win_sound(self):
2      if self.win_sound:
3          self.win_sound.play()

```

Penjelasan code:

- **def play_win_sound(self):** Metode ini adalah bagian dari kelas **SoundManager** dan memerlukan parameter **self** untuk mengakses atribut atau metode lain dalam kelas.

if self.win_sound:

- Mengecek apakah atribut `self.win_sound` tidak bernilai `None`.
- Jika `self.win_sound` memiliki nilai (artinya file suara kemenangan telah berhasil dimuat), maka program akan melanjutkan ke baris berikutnya.
 - `self.win_sound.play()`, Memutar efek suara yang tersimpan dalam `self.win_sound` menggunakan metode `play()` dari objek `pygame.mixer.Sound`. File suara yang dimuat (contohnya, "menang.mp3") akan diputar.

6) Pembuatan def play_lose_sound(self)

```

1  def play_lose_sound(self):
2      if self.lose_sound:
3          self.lose_sound.play()

```

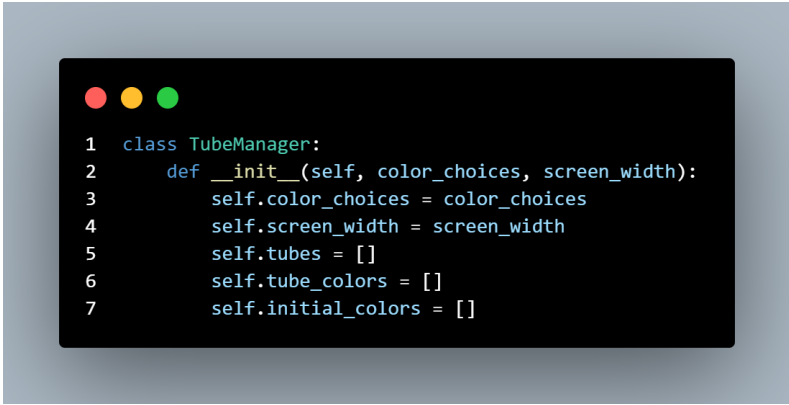
Penjelasan code:

- `def play_lose_sound(self)`: Metode ini adalah bagian dari kelas `SoundManager` dan memerlukan parameter `self` untuk mengakses atribut atau metode lain dalam kelas.

`if self.lose_sound:`

- Mengecek apakah atribut `self.lose_sound` tidak bernilai `None`.
- Jika `self.lose_sound` memiliki nilai (artinya file suara kekalahan telah berhasil dimuat), maka program akan melanjutkan ke baris berikutnya.
 - `self.lose_sound.play()` : Memutar efek suara yang tersimpan dalam `self.lose_sound` menggunakan metode `play()` dari objek `pygame.mixer.Sound`. File suara yang dimuat (contohnya, "kalah.mp3") akan diputar.

7) Pembuatan class TubeManager



```

1 class TubeManager:
2     def __init__(self, color_choices, screen_width):
3         self.color_choices = color_choices
4         self.screen_width = screen_width
5         self.tubes = []
6         self.tube_colors = []
7         self.initial_colors = []

```

Penjelasan code:

Class `TubeManager`, Kelas ini bertanggung jawab untuk mengelola logika utama permainan yang berkaitan dengan tabung dan warna.

- **`def __init__(self, color_choices, screen_width):`** Metode ini adalah konstruktor, yaitu metode khusus yang dipanggil saat sebuah instance (objek) dari kelas `TubeManager` dibuat. Dengan parameter:
 - **`color_choices`**: Daftar warna yang tersedia untuk diisi ke dalam tabung. Format biasanya berupa nilai RGB, misalnya `[(255, 0, 0), (0, 255, 0), (0, 0, 255)]`.
 - **`screen_width`**: Lebar layar permainan, digunakan untuk menentukan tata letak atau posisi tabung di layar.
 - **`self.color_choices = color_choices`**, Menyimpan pilihan warna (dalam format RGB) ke atribut `self.color_choices`, sehingga dapat digunakan di seluruh metode dalam kelas ini.
 - **`self.screen_width = screen_width`**, Menyimpan lebar layar ke atribut `self.screen_width`, yang digunakan untuk mengatur posisi tabung di layar permainan.
 - **`self.tubes = []`**, Inisialisasi daftar kosong untuk atribut `self.tubes`. Fungsinya adalah untuk menyimpan informasi tentang tabung-tabung yang ada dalam permainan. Nanti, tabung-tabung ini akan berisi cairan dengan warna tertentu.
 - **`self.tube_colors = []`**, Inisialisasi daftar kosong untuk atribut `self.tube_colors`, Daftar ini digunakan untuk menyimpan warna-warna , cairan yang ada di dalam masing-masing tabung. Setiap elemen dari daftar ini akan merepresentasikan warna dalam sebuah tabung (dalam bentuk daftar warna bertumpuk).
 - **`self.initial_colors = []`** Inisialisasi daftar kosong untuk atribut `self.initial_colors`.
Daftar ini menyimpan konfigurasi warna awal dari tabung-tabung. Fungsinya adalah untuk mereset permainan ke kondisi awal jika diperlukan.

8) Pembuatan def generate_start

```
1 def generate_start(self):
2     tubes_number = random.randint(10, 14)
3     tubes_colors = [[] for _ in range(tubes_number)]
4     available_colors = [i for i in range(tubes_number - 2) for _ in range(4)]
5
6     for i in range(tubes_number - 2):
7         for _ in range(4):
8             color = random.choice(available_colors)
9             tubes_colors[i].append(color)
10            available_colors.remove(color)
11
12     self.tubes = tubes_number
13     self.tube_colors = tubes_colors
```

Penjelasan Code:

- **def generate_start**, yang bertugas untuk membuat kondisi awal permainan dengan menentukan jumlah tabung, distribusi warna di setiap tabung, serta menyimpan susunan awal tabung sebagai referensi.
- **tubes_number** adalah jumlah tabung yang akan dibuat di permainan.
- **tubes_colors**: List berisi sublist kosong untuk setiap tabung. Setiap sublist akan diisi dengan warna (direpresentasikan sebagai angka) di tahap berikutnya.
- **available_colors**: List yang berisi angka yang merepresentasikan warna. Setiap warna muncul sebanyak 4 kali karena setiap warna harus memenuhi aturan permainan (4 cairan dengan warna yang sama).
- **for i in range(tubes_number - 2)**: Iterasi dilakukan untuk setiap tabung yang akan diisi cairan.. Hanya tabung dari indeks 0 hingga **tubes_number - 3** yang diisi cairan. Dua tabung terakhir dibiarkan kosong.
 - **for _ in range(4)**: Setiap tabung akan diisi dengan 4 warna cairan.
 - **random.choice(available_colors)**: Memilih warna secara acak dari **available_colors**.
 - **tubes_colors[i].append(color)**: Menambahkan warna yang dipilih ke tabung saat ini.
 - **available_colors.remove(color)**: Menghapus warna yang dipilih dari daftar warna yang tersedia untuk menghindari duplikasi warna lebih dari 4 kali.
- **self.tubes**: Menyimpan jumlah total tabung yang dibuat.

- **self.tube_colors**: Menyimpan daftar tabung beserta warna cairannya yang telah diacak.
- **self.initial_colors**: Membuat salinan mendalam (**deepcopy**) dari **tubes_colors** untuk menyimpan konfigurasi awal tabung.

9) Pembuatan def draw_tubes

```

1  def draw_tubes(self, screen, selected_index):
2      tube_boxes = []
3      tubes_per_row = self.tubes // 2 if self.tubes % 2 == 0 else self.tubes // 2 + 1
4      spacing = self.screen_width / tubes_per_row
5
6      def draw_single_row(offset, start, end, y_start):
7          for i in range(start, end):
8              for j in range(len(self.tube_colors[i])):
9                  pygame.draw.rect(screen, self.color_choices[self.tube_colors[i][j]],
10                                 [offset + 5 + spacing * (i - start), y_start - (50 * j), 65, 50], 0, 3)
11                 box = pygame.draw.rect(screen, 'blue',
12                                       [offset + 5 + spacing * (i - start), y_start - 150, 65, 200], 5, 5)
13                 if selected_index == i:
14                     pygame.draw.rect(screen, 'green',
15                                       [offset + 5 + spacing * (i - start), y_start - 150, 65, 200], 3, 5)
16                 tube_boxes.append(box)
17
18         draw_single_row(0, 0, tubes_per_row, 300)
19         if self.tubes % 2 == 0:
20             draw_single_row(0, tubes_per_row, self.tubes, 650)
21         else:
22             draw_single_row(spacing * 0.5, tubes_per_row, self.tubes, 650)
23
24         return tube_boxes

```

Penjelasan code:

- def **draw_tubes** yang bertugas menggambar tabung (tubes) beserta cairan (liquid) di dalamnya pada layar game menggunakan Pygame. Metode ini menggambar tabung-tabung permainan di layar beserta cairan di dalamnya dan mengembalikan posisi serta ukuran tabung dalam bentuk kotak (*rect*).
- **tube_boxes**: List untuk menyimpan kotak (bounding boxes) dari setiap tabung, yang akan digunakan untuk mendeteksi interaksi pemain (misalnya, klik pada tabung).
- **tubes_per_row**: Menghitung jumlah tabung yang akan diletakkan pada satu baris. Jika jumlah tabung (**self.tubes**) adalah bilangan ganjil, baris pertama memiliki satu tabung lebih banyak.
- **spacing**: Menghitung jarak horizontal antar tabung berdasarkan lebar layar (**self.screen_width**) dan jumlah tabung per baris.
- Fungsi def **draw_single_row** Menggambar semua tabung dalam satu baris, beserta cairan di dalamnya, dan menentukan kotak pembatas (bounding box) untuk masing-masing tabung. dengan parameter:

- **offset**: Offset horizontal tambahan untuk menyesuaikan posisi baris (digunakan untuk baris dengan jumlah tabung lebih sedikit).
- **start** dan **end**: Indeks awal dan akhir dari tabung yang akan digambar.
- **y_start**: Posisi vertikal awal dari tabung dalam baris tersebut.

■ Cara Kerja

1) Menggambar Cairan dalam Tabung:

- **for j in range(len(self.tube_colors[i]))**: Iterasi untuk menggambar cairan di dalam tabung berdasarkan warna dalam daftar **self.tube_colors[i]**.
- **pygame.draw.rect**: Gambar persegi panjang mewakili cairan di dalam tabung, dengan warna yang diambil dari **self.color_choices**.

2) Posisi dan ukuran:

- Posisi horizontal: Dihitung berdasarkan indeks tabung dalam baris (**i - start**) dan jarak antar tabung (**spacing**).
- Posisi vertikal: Dihitung berdasarkan urutan cairan dalam tabung (**j**).
- Ukuran persegi panjang: Lebar 65 piksel, tinggi 50 piksel.

3) Menggambar Tabung:

- Tabung digambar menggunakan persegi panjang dengan garis luar biru (*stroke*) setebal 5 piksel.
- Kotak pembatas tabung disimpan dalam **tube_boxes**.

4) Tabung yang Dipilih:

- Jika **selected_index** sama dengan indeks tabung saat ini (**i**), tabung akan diberi bingkai hijau (*stroke*) untuk menunjukkan bahwa tabung tersebut sedang dipilih.

5) Bounding Box:

- Setiap tabung yang digambar memiliki bounding box (kotak pembatas) yang ditambahkan ke dalam **tube_boxes**, yang akan digunakan untuk mendeteksi klik pada tabung.
- Memanggil **draw_single_row** tabung dari indeks 0 hingga indeks **tubes_per_row** digambar pada baris pertama dengan posisi vertikal awal 300.
 - Jika jumlah tabung genap, sisa tabung digambar dengan offset 0 pada baris kedua di posisi vertikal awal 650.
 - Jika jumlah tabung ganjil, baris kedua digeser sedikit ke kanan (**offset sebesar spacing * 0.5**) untuk membuat tata letak lebih simetris.
- **return tube_boxes** Mengembalikan daftar bounding box dari semua tabung yang telah digambar, sehingga tabung-tabung tersebut dapat diakses untuk mendeteksi interaksi seperti klik dengan mouse.

10) Pembuatan def reset(self)

```
1 def reset(self):
2     self.tube_colors = copy.deepcopy(self.initial_colors)
3
```

Penjelasan code

- **def reset** digunakan untuk mengembalikan susunan tabung beserta warnanya ke konfigurasi awal saat permainan dimulai.
- **self.initial_colors**: Menyimpan konfigurasi awal tabung yang telah dihasilkan oleh fungsi **generate_start**.
 - **self.tube_colors**: Menyimpan susunan tabung saat ini. Dalam permainan, warna cairan di tabung ini akan berubah akibat interaksi pemain. Pada saat fungsi **reset** dipanggil, nilai **self.tube_colors** diatur ulang menjadi salinan dari **self.initial_colors** sehingga susunan tabung kembali ke kondisi awal.

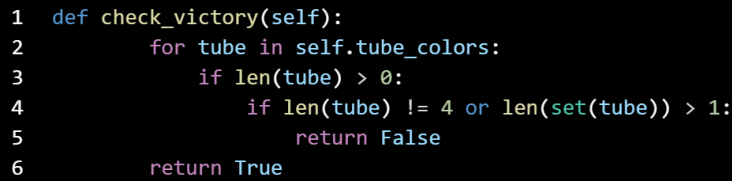
11) Pembuatan def calc_move

```
1 def calc_move(self, selected_index, destination_index, sound_manager):
2     if len(self.tube_colors[selected_index]) > 0:
3         color_to_move = self.tube_colors[selected_index][-1]
4         if len(self.tube_colors[destination_index]) < 4 and \
5             (len(self.tube_colors[destination_index]) == 0 or
6              self.tube_colors[destination_index][-1] == color_to_move):
7             while len(self.tube_colors[destination_index]) < 4 and \
8                 len(self.tube_colors[selected_index]) > 0 and \
9                 self.tube_colors[selected_index][-1] == color_to_move:
10                self.tube_colors[destination_index].append(self.tube_colors[selected_index].pop())
11            sound_manager.play_move_sound()
12            return True
13        return False
```

Penjelasan code:

Fungsi **calc_move** adalah inti dari logika permainan yang mengatur perpindahan cairan berdasarkan aturan. Fungsi ini memastikan permainan berjalan sesuai mekanisme dan memberikan umpan balik kepada pemain melalui efek suara.

12) Pembuatan def check_victory



```

1 def check_victory(self):
2     for tube in self.tube_colors:
3         if len(tube) > 0:
4             if len(tube) != 4 or len(set(tube)) > 1:
5                 return False
6     return True

```

Penjelasan code:

- **def check_victory:** Mengevaluasi semua tabung dalam permainan untuk memastikan bahwa setiap tabung yang tidak kosong memiliki cairan homogen (warna sama) dan terisi penuh (4 elemen).
- **for tube in self.tube_colors:** Fungsi memeriksa setiap tabung dalam list **self.tube_colors**. **self.tube_colors** adalah list yang berisi list warna cairan di setiap tabung.
- **if len(tube) > 0:** Tabung kosong tidak diperiksa lebih lanjut karena tidak memengaruhi kondisi kemenangan.
- **if len(tube) != 4 or len(set(tube)) > 1: return False**

Kondisi kemenangan untuk setiap tabung yang berisi cairan:

1. **Penuh:** Panjang tabung harus 4 (**len(tube) == 4**).
2. **Homogen:** Semua elemen di dalam tabung harus memiliki warna yang sama. Ini diperiksa menggunakan:
 - **set(tube):** Mengubah list menjadi set. Jika semua elemen sama, panjang set akan menjadi 1.
 - Jika panjang set lebih dari 1 (**len(set(tube)) > 1**), berarti ada lebih dari satu warna dalam tabung.
 - Jika salah satu kondisi tidak terpenuhi, fungsi langsung mengembalikan **False**, artinya kemenangan belum tercapai.
- **return True** Jika semua tabung dalam **self.tube_colors** memenuhi syarat kemenangan, fungsi mengembalikan **True**. Artinya, pemain telah memenangkan permainan.

13) Pembuatan def check_loss

```

1 def check_loss(self):
2     for i in range(len(self.tube_colors)):
3         for j in range(len(self.tube_colors)):
4             if i != j and len(self.tube_colors[i]) > 0 and len(self.tube_colors[j]) < 4:
5                 if len(self.tube_colors[j]) == 0 or self.tube_colors[i][-1] == self.tube_colors[j][-1]:
6                     return False
7     return True

```

Penjelasan code:

➤ `def check_loss` bertugas untuk memeriksa apakah kondisi kekalahan telah tercapai dalam permainan. Kondisi kekalahan terjadi ketika tidak ada lagi gerakan valid yang dapat dilakukan

- `for i in range(len(self.tube_colors)):`
- `for j in range(len(self.tube_colors)):`

Fungsi menggunakan **dua loop bertingkat** untuk membandingkan semua kombinasi tabung dalam permainan. **i** adalah indeks tabung sumber **j** adalah indeks tabung tujuan.

- `if i != j`: Memastikan tabung yang dibandingkan bukan tabung yang sama (**tabung sumber dan tabung tujuan berbeda**).
- `len(self.tube_colors[i]) > 0` : Tabung sumber (`self.tube_colors[i]`) harus memiliki cairan (tidak kosong) agar bisa memindahkan cairan ke tabung lain.
- `len(self.tube_colors[j]) < 4` : Tabung tujuan (`self.tube_colors[j]`) harus memiliki ruang kosong (kurang dari 4 cairan).
- `if len(self.tube_colors[j]) == 0 or self.tube_colors[i][-1] == self.tube_colors[j][-1]`: **Tabung tujuan kosong**: Tabung kosong (`len(self.tube_colors[j]) == 0`) selalu memungkinkan untuk menerima cairan dari tabung sumber.
- `return True` Jika setelah memeriksa semua kombinasi tabung tidak ditemukan gerakan valid, fungsi mengembalikan **True**, artinya pemain kalah.

14) Pembuatan class Game

```

1 class Game:

```

Penjelasan code:

`class Game` adalah kerangka utama untuk permainan. Fungsi utamanya adalah untuk menyatukan dan mengelola elemen-elemen permainan seperti logika, grafis, input, dan suara. Kelas ini memegang kendali atas alur permainan secara keseluruhan.

15) Pembuatan def __init__ pada class Game

```
1 def __init__(self):
2     self.WIDTH = 900
3     self.HEIGHT = 1000
4     self.screen = pygame.display.set_mode([self.WIDTH, self.HEIGHT])
5     pygame.display.set_caption('Water Sort PyGame')
6     self.font = pygame.font.Font('freesansbold.ttf', 36)
7     self.large_font = pygame.font.Font('freesansbold.ttf', 72)
8     self.fps = 60
9     self.timer = pygame.time.Clock()
10
11     color_choices = [(255, 0, 0), (255, 165, 0), (173, 216, 230), (0, 0, 255),
12                      (0, 100, 0), (255, 192, 203), (128, 0, 128), (169, 169, 169),
13                      (139, 69, 19), (144, 238, 144), (255, 255, 0), (255, 255, 255)]
14
15     self.sound_manager = SoundManager()
16     self.tube_manager = TubeManager(color_choices, self.WIDTH)
17
18     self.new_game = True
19     self.selected = False
20     self.select_index = -1
21     self.win = False
22     self.lose = False
23     self.score = 0
24     self.message_played = False
```

Penjelasan code

- constructor (`__init__`), yang akan dijalankan pertama kali saat objek dari kelas ini dibuat. Fungsinya untuk menginisialisasi atribut-atribut utama dalam game.
 - **WIDTH** dan **HEIGHT** mengatur ukuran layar game (900x1000 piksel).
 - **screen** membuat jendela game menggunakan `pygame.display.set_mode`.
 - `pygame.display.set_caption` memberikan judul game "Water Sort PyGame" di bagian atas jendela.
 - **font** adalah font utama untuk teks kecil (ukuran 36).
 - **large_font** adalah font untuk teks besar (ukuran 72), biasanya digunakan untuk pesan kemenangan/kekalahan.
 - **fps** menentukan jumlah frame yang akan dirender setiap detik (diatur ke 60 FPS agar animasi lancar).
 - **timer** digunakan untuk mengatur kecepatan frame dengan fungsi `Clock()`.
 - **color_choices** adalah daftar warna dalam format RGB yang digunakan untuk cairan di tabung. Contoh: (255, 0, 0) adalah merah, (0, 0, 255) adalah biru, (255, 192, 203) adalah pink, dll.
 - **sound_manager** adalah objek dari kelas `SoundManager`, yang bertugas mengelola efek suara dalam game (contohnya saat menuang cairan).
 - **new_game**: Menandakan apakah ini adalah game baru (diatur ke `True` saat game baru dimulai).
 - **selected**: Status apakah ada tabung yang sedang dipilih oleh pemain.

- **select_index**: Menyimpan indeks tabung yang dipilih pemain (default -1 berarti tidak ada yang dipilih).
- **win**: Status apakah pemain sudah menang (**True** jika menang).
- **lose**: Status apakah pemain kalah (**True** jika tidak ada langkah yang bisa dilakukan).
- **score**: Menyimpan nilai skor pemain.
- **message_played**: Menandakan apakah pesan kemenangan atau kekalahan sudah ditampilkan di layar.

16) Pembuatan def display_message

```
1 def display_message(self, text, y_offset):
2     message = self.large_font.render(text, True, 'white')
3     self.screen.blit(message, (self.WIDTH // 2 - message.get_width() // 2, self.HEIGHT // 2 + y_offset))
4
```

Penjelasan code

- Fungsi **display_message** bertugas untuk menampilkan pesan teks tertentu di layar game. Dengan parameter:
 - **text**: Pesan teks yang akan ditampilkan di layar (misalnya "You Won!" atau "You Lose!").
 - **y_offset**: Nilai untuk mengatur posisi vertikal pesan di layar.
 - **self.large_font.render**: Membuat objek teks yang bisa ditampilkan di layar.
 - **text**: Teks yang akan ditampilkan.
 - **True**: Mengaktifkan anti-aliasing agar teks terlihat halus.
 - **'white'**: Warna teks (putih dalam format string).
 - **self.screen.blit**: Menempatkan gambar teks ke layar game. Menempatkan teks di tengah layar secara horizontal dengan mengurangi setengah dari lebar teks (**message.get_width()**) dari setengah lebar layar (**self.WIDTH // 2**). Menempatkan teks di tengah layar secara vertikal (**self.HEIGHT // 2**) dan menyesuaikan posisinya dengan nilai offset (**y_offset**).

17) Pembuatan def run(self)


```
1 def run(self):
2     running = True
3     while running:
4         self.screen.fill('black')
5         self.timer.tick(self.fps)
6
7         if self.new_game:
8             self.tube_manager.generate_start()
9             self.new_game = False
10            self.score = 0
11            self.win = False
12            self.lose = False
13            self.message_played = False
14
15            tube_boxes = self.tube_manager.draw_tubes(self.screen, self.select_index)
16
```

Penjelasan code:

- Fungsi **run** ini merupakan loop utama dalam permainan. Fungsi ini menjalankan logika, pembaruan, dan rendering game secara terus-menerus selama permainan berlangsung.
 - **running**: Variabel kontrol untuk loop utama. Selama **running = True**, game akan terus berjalan.
 - **while running**: Memulai **loop utama** game yang berfungsi untuk: Memproses event (input pemain), Memperbarui logika game, Merender ulang tampilan pada layar setiap frame.
 - **self.screen.fill('black')** Mengisi layar game dengan warna hitam (background). Ini dilakukan di setiap frame untuk **menghapus gambar sebelumnya** sehingga hanya elemen terbaru yang ditampilkan.
 - **self.timer.tick(self.fps)**: Membatasi kecepatan frame agar game berjalan pada **frame rate** tertentu (ditentukan oleh **self.fps**, misalnya 60 FPS).
 - **if self.new_game::** Mengecek apakah permainan baru dimulai. Jika **self.new_game** adalah **True**, maka:
 - **self.tube_manager.generate_start()**: Memanggil fungsi **generate_start** dari **self.tube_manager** untuk menghasilkan tabung (tubes) dan warna baru.
 - **self.new_game = False**: Menandakan permainan baru telah dimulai.
 - **self.score = 0**: Reset skor ke 0.
 - **self.win = False**: Menandakan bahwa pemain belum menang.
 - **self.lose = False**: Menandakan bahwa pemain belum kalah.
 - **self.message_played = False**: Menandakan bahwa pesan kemenangan atau kekalahan belum ditampilkan.
 - Memanggil metode **draw_tubes** dari **self.tube_manager** untuk menggambar tabung-tabung pada layar. **Parameter**:
 - **self.screen**: Layar game tempat tabung akan digambar.
 - **self.select_index**: Indeks tabung yang saat ini dipilih oleh pemain (jika ada).

```

1  if not self.win and not self.lose:
2      self.win = self.tube_manager.check_victory()
3      self.lose = self.tube_manager.check_loss() and not self.win
4
5      if self.win and not self.message_played:
6          self.sound_manager.play_win_sound()
7          self.message_played = True
8      elif self.lose and not self.message_played:
9          self.sound_manager.play_lose_sound()
10         self.message_played = True
11
12     if self.win:
13         self.display_message('You Won!', -50)
14     elif self.lose:
15         self.display_message('You Lose!', -50)
16

```

Penjelasan code

- **if not self.win and not self.lose:** Mengecek apakah pemain belum menang (**self.win**) dan belum kalah (**self.lose**). Jika belum ada hasil akhir, maka program akan memeriksa status kemenangan atau kekalahan.
 - **self.win = self.tube_manager.check_victory()** Memanggil fungsi **check_victory()** dari **self.tube_manager** untuk menentukan apakah kondisi kemenangan sudah terpenuhi (misalnya semua tabung hanya berisi warna yang sama).
 - **self.lose = self.tube_manager.check_loss() and not self.win** Memanggil fungsi **check_loss()** untuk memeriksa apakah pemain sudah kalah (misalnya tidak ada gerakan valid yang tersisa). **and not self.win** memastikan kekalahan hanya akan diatur jika pemain belum menang. Hal ini mencegah konflik antara kondisi menang dan kalah.
- **if self.win and not self.message_played:** Mengecek apakah pemain menang (**self.win**) dan efek suara kemenangan belum dimainkan (**self.message_played**). Jika kondisi terpenuhi:
 - **self.sound_manager.play_win_sound():** Memanggil metode **play_win_sound()** dari **self.sound_manager** untuk memainkan efek suara kemenangan.
 - **self.message_played = True:** Menandakan bahwa efek suara telah dimainkan sehingga tidak akan dimainkan lagi.
- **elif self.lose and not self.message_played:** Mengecek apakah pemain kalah (**self.lose**) dan efek suara kekalahan belum dimainkan (**self.message_played**). Jika kondisi terpenuhi:
 - **self.sound_manager.play_lose_sound():** Memanggil metode **play_lose_sound()** dari **self.sound_manager** untuk memainkan efek suara kekalahan.

- **self.message_played = True:** Menandakan bahwa efek suara telah dimainkan.
- **if self.win:** Jika pemain menang (**self.win** adalah **True**):
 - Memanggil metode **self.display_message** untuk menampilkan pesan **"You Won!"** di layar.
 - **-50** adalah offset vertikal untuk posisi pesan (berarti pesan ditampilkan sedikit di atas tengah layar).
- **elif self.lose:** Jika pemain kalah (**self.lose** adalah **True**):
 - Memanggil metode **self.display_message** untuk menampilkan pesan **"You Lose!"** di layar.
 - **-50** adalah offset yang sama untuk posisi pesan.

```

1 score_text = self.font.render(f'Score: {self.score}', True, 'white')
2   self.screen.blit(score_text, (10, 50))
3
4   restart_text = self.font.render('Space: Restart | Enter: New Board', True, 'white')
5   self.screen.blit(restart_text, (self.WIDTH // 2 - restart_text.get_width() // 2, 10))
6

```

Penjelasan code

- kode ini bertanggung jawab untuk menampilkan skor pemain dan instruksi tombol (untuk restart atau memulai papan permainan baru) di layar game.
- **self.font.render(f'Score: {self.score}', True, 'white'):**
 - **self.font:** Mengacu pada font yang sudah didefinisikan sebelumnya dalam inisialisasi game.
 - **f'Score: {self.score}':** Membuat teks dinamis yang menampilkan skor pemain saat ini. Nilai **self.score** adalah variabel yang melacak skor pemain selama permainan.
 - Contoh teks yang dirender: "Score: 50".
 - **True:** Parameter untuk *antialiasing*, memastikan teks terlihat lebih halus.
 - **'white':** Warna teks adalah putih.
- **self.screen.blit(score_text, (10, 50)):**
 - **self.screen.blit** digunakan untuk menampilkan **score_text** di layar.
 - **(10, 50)** adalah koordinat posisi teks di layar: **10:** Posisi horizontal (margin kiri 10 piksel dari tepi layar). **50:** Posisi vertikal (50 piksel dari atas layar).

```

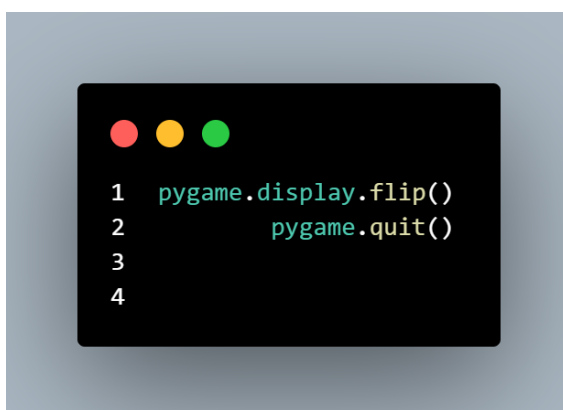
1 for event in pygame.event.get():
2     if event.type == pygame.QUIT:
3         running = False
4     elif event.type == pygame.KEYUP:
5         if event.key == pygame.K_SPACE:
6             self.tube_manager.reset()
7             self.lose = False
8             self.win = False
9             self.score = 0
10            self.message_played = False
11        elif event.key == pygame.K_RETURN:
12            self.new_game = True
13    elif event.type == pygame.MOUSEBUTTONDOWN:
14        if not self.selected:
15            for idx, rect in enumerate(tube_boxes):
16                if rect.collidepoint(event.pos):
17                    self.selected = True
18                    self.select_index = idx
19        else:
20            for idx, rect in enumerate(tube_boxes):
21                if rect.collidepoint(event.pos):
22                    if self.tube_manager.calc_move(self.select_index, idx, self.sound_manager):
23                        self.score += 10
24                    self.selected = False
25                    self.select_index = -1

```

Penjelasan code

- bagian dari loop utama game untuk menangani berbagai event input dari pemain. Fungsinya adalah untuk memproses perintah yang diberikan pemain melalui mouse, keyboard, atau aksi lainnya, dan mengubah status permainan berdasarkan tindakan pemain.
- **pygame.event.get()**: Mengambil semua event yang terjadi selama siklus loop game. Event ini bisa berupa input dari pemain (klik mouse, tombol keyboard) atau sistem (misalnya, jendela game ditutup) **event**: Merepresentasikan satu event yang terjadi.
- Jika pemain menutup jendela game (klik tombol "X"), maka event **pygame.QUIT** dipicu. **running = False**: Menghentikan loop utama game, yang menyebabkan permainan berhenti.
- **elif event.type == pygame.KEYUP**: Mengecek apakah ada tombol keyboard yang dilepaskan (bukan ditekan). Jika tombol **Space** ditekan:
 - **self.tube_manager.reset()**: Mereset papan permainan ke kondisi awal (warna dan posisi tabung dikembalikan seperti awal permainan).
 - **self.lose = False, self.win = False**: Mengatur ulang status kemenangan dan kekalahan menjadi **False**.
 - **self.score = 0**: Skor permainan direset kembali ke **0**.

- **self.message_played = False**: Status pesan kemenangan/kekalahan direset agar dapat dimainkan lagi jika diperlukan.
- **not self.selected**: Mengecek apakah belum ada tabung yang dipilih.
- **for idx, rect in enumerate(tube_boxes)**: Melakukan iterasi untuk memeriksa apakah klik mouse pemain mengenai salah satu tabung.
rect.collidepoint(event.pos): Mengecek apakah posisi klik mouse (koordinat **event.pos**) berada di dalam area salah satu tabung (**rect**). Jika tabung diklik:
 - **self.selected = True**: Menandai bahwa sebuah tabung telah dipilih.
 - **self.select_index = idx**: Menyimpan indeks tabung yang dipilih.
 - **else**: Menangani kondisi ketika ada tabung yang sudah dipilih.
 - **if rect.collidepoint(event.pos)**: Mengecek apakah klik mouse berikutnya berada di salah satu tabung lain.
 - **self.tube_manager.calc_move(self.select_index, idx, self.sound_manager)** : Fungsi ini mencoba untuk memindahkan cairan dari tabung yang dipilih (**self.select_index**) ke tabung target (**idx**).
 - Jika perpindahan berhasil: **self.score += 10**: Menambah skor sebesar **10 poin** setiap kali cairan berhasil dipindahkan.
 - **self.selected = False** dan **self.select_index = -1**:
 - Mengatur ulang status pemilihan tabung setelah perpindahan selesai.



Penjelasan code

- **pygame.display.flip()** Fungsi: Mengupdate layar (display) dengan semua perubahan yang telah dibuat pada frame saat ini.
- **pygame.quit()** Fungsi: Menutup aplikasi Pygame dengan benar.



Penjelasan code

- `if __name__ == "__main__":`: Memastikan kode hanya dijalankan jika file ini adalah file utama, bukan modul impor.
- `game = Game()`: Membuat objek game dengan semua variabel dan elemen awal.
 - `game.run()`: Menjalankan game loop untuk memulai permainan.

3.3 Implementasi OOP Pada Game

Game ini menggunakan konsep OOP agar kode lebih mudah dibaca dan efisien dalam mengelola fitur permainan. Setiap kelas memiliki tugas khusus, seperti mengelola suara, tabung warna, atau logika permainan. Hal ini membuat kode menjadi lebih terstruktur, sehingga lebih mudah untuk dimodifikasi atau dikembangkan. Berikut ini adalah OOP yang digunakan pada pygame ini:

3.3.1 Encapsulation

Encapsulation adalah konsep dalam OOP yang diterapkan dengan cara mengelompokkan data dan fungsi yang saling terkait ke dalam sebuah kelas. Dalam game ini, setiap kelas memiliki peran khusus dengan atribut dan metode yang relevan hanya untuk tugas tertentu.

1. Kelas SoundManager

Berfungsi untuk mengelola suara permainan, termasuk suara gerakan, kemenangan, dan kekalahan.

```
pygame.init()

class SoundManager:
    # Manajer Suara digunakan untuk mengelola efek suara seperti suara gerakan, menang, atau kalah.
    def __init__(self):
        # Memuat file suara untuk gerakan, kemenangan, dan kekalahan.
```

```

    try:
        self.move_sound = pygame.mixer.Sound("suara air.wav") # Suara saat ada
gerakan air
        self.win_sound = pygame.mixer.Sound("menang.mp3") # Suara saat menang
        self.lose_sound = pygame.mixer.Sound("kalah.mp3") # Suara saat kalah
    except pygame.error as e:
        print("File suara tidak ditemukan atau formatnya tidak valid:", e)
        self.move_sound = None
        self.win_sound = None
        self.lose_sound = None

# Memainkan suara gerakan
def play_move_sound(self):
    if self.move_sound:
        self.move_sound.play()

# Memainkan suara kemenangan
def play_win_sound(self):
    if self.win_sound:
        self.win_sound.play()

# Memainkan suara kekalahan
def play_lose_sound(self):
    if self.lose_sound:
        self.lose_sound.play()

```

2. Kelas TubeManager

Berfungsi untuk mengelola tabung warna, termasuk pembuatan awal, penggambaran di layar, serta pengecekan kemenangan atau kekalahan.

```

class TubeManager:
    def __init__(self, color_choices, screen_width):
        self.color_choices = color_choices # Pilihan warna tabung
        self.screen_width = screen_width # Lebar layar untuk tata letak
        self.tubes = [] # Jumlah tabung
        self.tube_colors = [] # Warna dalam tabung
        self.initial_colors = [] # Warna awal untuk reset

    def generate_start(self):
        # Menghasilkan tata letak awal tabung secara acak
        tubes_number = random.randint(10, 14) # Jumlah tabung antara 10 hingga 14
        tubes_colors = [[] for _ in range(tubes_number)] # Membuat daftar warna kosong
untuk setiap tabung
        available_colors = [i for i in range(tubes_number - 2) for _ in range(4)] #
Warna-warna yang akan digunakan

        for i in range(tubes_number - 2):
            for _ in range(4):
                color = random.choice(available_colors)
                tubes_colors[i].append(color)
                available_colors.remove(color)

        self.tubes = tubes_number
        self.tube_colors = tubes_colors
        self.initial_colors = copy.deepcopy(tubes_colors) # Menyimpan warna awal untuk
reset

```

```

def draw_tubes(self, screen, selected_index):
    # Menggambar tabung pada layar
    tube_boxes = []
    tubes_per_row = self.tubes // 2 if self.tubes % 2 == 0 else self.tubes // 2 + 1
    # Tabung per baris
    spacing = self.screen_width / tubes_per_row # Spasi antar tabung

    def draw_single_row(offset, start, end, y_start):
        for i in range(start, end):
            for j in range(len(self.tube_colors[i])):
                # Menggambar blok warna di dalam tabung
                pygame.draw.rect(screen,
self.color_choices[self.tube_colors[i][j]],
                                [offset + 5 + spacing * (i - start), y_start - (50
* j), 65, 50], 0, 3)
                # Menggambar tabung
                box = pygame.draw.rect(screen, 'blue',
                                [offset + 5 + spacing * (i - start), y_start -
150, 65, 200], 5, 5)
                if selected_index == i:
                    # Menyorot tabung yang dipilih
                    pygame.draw.rect(screen, 'green',
                                [offset + 5 + spacing * (i - start), y_start - 150,
65, 200], 3, 5)
                tube_boxes.append(box)

    # Menggambar dua baris tabung
    draw_single_row(0, 0, tubes_per_row, 300)
    if self.tubes % 2 == 0:
        draw_single_row(0, tubes_per_row, self.tubes, 650)
    else:
        draw_single_row(spacing * 0.5, tubes_per_row, self.tubes, 650)

    return tube_boxes

def reset(self):
    # Mereset warna tabung ke tata letak awal
    self.tube_colors = copy.deepcopy(self.initial_colors)

def calc_move(self, selected_index, destination_index, sound_manager):
    # Menghitung gerakan antara dua tabung
    if len(self.tube_colors[selected_index]) > 0:
        color_to_move = self.tube_colors[selected_index][-1] # Warna di bagian atas
tabung
        if len(self.tube_colors[destination_index]) < 4 and \
            (len(self.tube_colors[destination_index]) == 0 or
             self.tube_colors[destination_index][-1] == color_to_move):
            # Memindahkan warna selama syarat terpenuhi
            while len(self.tube_colors[destination_index]) < 4 and \
                len(self.tube_colors[selected_index]) > 0 and \
                self.tube_colors[selected_index][-1] == color_to_move:
self.tube_colors[destination_index].append(self.tube_colors[selected_index].pop())
                sound_manager.play_move_sound()
            return True
        return False

```



```

def check_victory(self):
    # Mengecek apakah semua tabung memiliki warna yang sama atau kosong
    for tube in self.tube_colors:
        if len(tube) > 0:
            if len(tube) != 4 or len(set(tube)) > 1:
                return False
    return True

def check_loss(self):
    # Mengecek apakah tidak ada lagi gerakan yang mungkin dilakukan
    for i in range(len(self.tube_colors)):
        for j in range(len(self.tube_colors)):
            if i != j and len(self.tube_colors[i]) > 0 and len(self.tube_colors[j])
< 4:
                if len(self.tube_colors[j]) == 0 or self.tube_colors[i][-1] ==
self.tube_colors[j][-1]:
                    return False
    return True

```

3. Kelas Game

Berfungsi untuk mengelola logika utama permainan, seperti antarmuka pengguna, logika input, dan status permainan.

```

class Game:
    def __init__(self):
        self.WIDTH = 900 # Lebar layar
        self.HEIGHT = 1000 # Tinggi layar
        self.screen = pygame.display.set_mode([self.WIDTH, self.HEIGHT]) # Membuat
jendela layar
        pygame.display.set_caption('Water Sort PyGame') # Judul permainan
        self.font = pygame.font.Font('freesansbold.ttf', 36) # Font untuk skor
        self.large_font = pygame.font.Font('freesansbold.ttf', 72) # Font untuk pesan
besar
        self.fps = 60 # Kecepatan frame
        self.timer = pygame.time.Clock() # Pengatur waktu

        # Warna-warna yang digunakan dalam permainan
        color_choices = [(255, 0, 0), (255, 165, 0), (173, 216, 230), (0, 0, 255),
                        (0, 100, 0), (255, 192, 203), (128, 0, 128), (169, 169, 169),
                        (139, 69, 19), (144, 238, 144), (255, 255, 0), (255, 255, 255)]

        self.sound_manager = SoundManager() # Mengelola suara
        self.tube_manager = TubeManager(color_choices, self.WIDTH) # Mengelola tabung

        self.new_game = True
        self.selected = False
        self.select_index = -1
        self.win = False
        self.lose = False
        self.score = 0
        self.message_played = False

    def display_message(self, text, y_offset):
        # Menampilkan pesan di tengah layar
        message = self.large_font.render(text, True, 'white')
        self.screen.blit(message, (self.WIDTH // 2 - message.get_width() // 2,
self.HEIGHT // 2 + y_offset))

```

```

def run(self):
    # Menjalankan loop utama permainan
    running = True
    while running:
        self.screen.fill('black') # Membersihkan layar
        self.timer.tick(self.fps) # Menjaga kecepatan frame

        if self.new_game:
            self.tube_manager.generate_start() # Membuat tata letak baru
            self.new_game = False
            self.score = 0
            self.win = False
            self.lose = False
            self.message_played = False

        tube_boxes = self.tube_manager.draw_tubes(self.screen, self.select_index)
# Menggambar tabung

        if not self.win and not self.lose:
            self.win = self.tube_manager.check_victory() # Mengecek kemenangan
            self.lose = self.tube_manager.check_loss() and not self.win # Mengecek
kekalahan

        if self.win and not self.message_played:
            self.sound_manager.play_win_sound() # Memainkan suara kemenangan
            self.message_played = True
        elif self.lose and not self.message_played:
            self.sound_manager.play_lose_sound() # Memainkan suara kekalahan
            self.message_played = True

        if self.win:
            self.display_message('Anda Menang!', -50) # Pesan menang
        elif self.lose:
            self.display_message('Anda Kalah!', -50) # Pesan kalah

        # Menampilkan skor
        score_text = self.font.render(f'Skor: {self.score}', True, 'white')
        self.screen.blit(score_text, (10, 50))

        # Menampilkan instruksi
        restart_text = self.font.render('Spasi: Ulangi | Enter: Tata Letak Baru',
True, 'white')
        self.screen.blit(restart_text, (self.WIDTH // 2 - restart_text.get_width()
// 2, 10))

        for event in pygame.event.get():
            if event.type == pygame.QUIT:
                running = False # Menghentikan permainan
            elif event.type == pygame.KEYUP:
                if event.key == pygame.K_SPACE:
                    self.tube_manager.reset() # Mengulang permainan dengan tata
letak yang sama

                    self.lose = False
                    self.win = False
                    self.score = 0
                    self.message_played = False

```

```

        elif event.key == pygame.K_RETURN:
            self.new_game = True # Membuat tata letak baru
        elif event.type == pygame.MOUSEBUTTONDOWN:
            if not self.selected:
                for idx, rect in enumerate(tube_boxes):
                    if rect.collidepoint(event.pos):
                        self.selected = True # Memilih tabung
                        self.select_index = idx
            else:
                for idx, rect in enumerate(tube_boxes):
                    if rect.collidepoint(event.pos):
                        if self.tube_manager.calc_move(self.select_index, idx,
self.sound_manager):
                            self.score += 10 # Menambah skor saat gerakan
berhasil

                            self.selected = False
                            self.select_index = -1

                pygame.display.flip() # Memperbarui layar
                pygame.quit()

```

3.3.2 Abstraction

Abstraction atau abstraksi dalam OOP adalah konsep yang menyederhanakan sistem dengan menyembunyikan detail implementasi yang rumit dan hanya menampilkan fungsionalitas yang penting kepada pengguna. Dalam game ini, abstraksi terlihat jelas pada metode:

1. Metode `draw_tubes` memungkinkan pemain melihat tabung dan warna di layar tanpa perlu mengetahui proses penghitungan posisi atau rendering warna.

```

1 def draw_tubes(self, screen, selected_index):
2     # Menggambar tabung pada layar
3     tube_boxes = []
4     tubes_per_row = self.tubes // 2 if self.tubes % 2 == 0 else self.tubes // 2 + 1 # Tabung per baris
5     spacing = self.screen_width / tubes_per_row # Spasi antar tabung
6
7     def draw_single_row(offset, start, end, y_start):
8         for i in range(start, end):
9             for j in range(len(self.tube_colors[i])):
10                # Menggambar blok warna di dalam tabung
11                pygame.draw.rect(screen, self.color_choices[self.tube_colors[i][j]],
12                                [offset + 5 + spacing * (i - start), y_start - (50 * j), 65, 50], 0, 3)
13            # Menggambar tabung
14            box = pygame.draw.rect(screen, 'blue',
15                                [offset + 5 + spacing * (i - start), y_start - 150, 65, 200], 5, 5)
16            if selected_index == i:
17                # Menyorot tabung yang dipilih
18                pygame.draw.rect(screen, 'green',
19                                [offset + 5 + spacing * (i - start), y_start - 150, 65, 200], 3, 5)
20            tube_boxes.append(box)

```

2. Metode `check_victory` dan `check_loss` memberi informasi apakah pemain menang atau kalah tanpa mengungkapkan logika di balik pengecekan kondisi permainan.

```

1 def check_victory(self):
2     # Mengecek apakah semua tabung memiliki warna yang sama atau kosong
3     for tube in self.tube_colors:
4         if len(tube) > 0:
5             if len(tube) != 4 or len(set(tube)) > 1:
6                 return False
7     return True
8
9 def check_loss(self):
10    # Mengecek apakah tidak ada lagi gerakan yang mungkin dilakukan
11    for i in range(len(self.tube_colors)):
12        for j in range(len(self.tube_colors)):
13            if i != j and len(self.tube_colors[i]) > 0 and len(self.tube_colors[j]) < 4:
14                if len(self.tube_colors[j]) == 0 or self.tube_colors[i][-1] == self.tube_colors[j][-1]:
15                    return False
16    return True

```

3. Metode `calc_move`, menyembunyikan proses pengecekan syarat perpindahan warna dan pengelolaan data dalam list, sehingga pemain hanya melihat hasil perpindahan warna yang sederhana. Dengan abstraksi, pemain dapat fokus pada permainan tanpa terganggu oleh detail teknis di balik layar.

```

1 def calc_move(self, selected_index, destination_index, sound_manager):
2     # Menghitung gerakan antara dua tabung
3     if len(self.tube_colors[selected_index]) > 0:
4         color_to_move = self.tube_colors[selected_index][-1] # Warna di bagian atas tabung
5         if len(self.tube_colors[destination_index]) < 4 and \
6             (len(self.tube_colors[destination_index]) == 0 or
7              self.tube_colors[destination_index][-1] == color_to_move):
8             # Memindahkan warna selama syarat terpenuhi
9             while len(self.tube_colors[destination_index]) < 4 and \
10                 len(self.tube_colors[selected_index]) > 0 and \
11                 self.tube_colors[selected_index][-1] == color_to_move:
12                 self.tube_colors[destination_index].append(self.tube_colors[selected_index].pop())
13             sound_manager.play_move_sound()
14             return True
15     return False

```

3.3.3 Polymorphism

Polimorfisme adalah kemampuan sebuah entitas untuk memiliki berbagai bentuk atau fungsi tergantung pada konteksnya. Dalam hal ini, `pygame.draw.rect` memanfaatkan prinsip tersebut untuk menggambar elemen visual yang berbeda dalam permainan:

1. Menggambar Blok Warna di Dalam Tabung

Tujuannya menampilkan isi tabung sebagai blok warna berurutan.

```

1 pygame.draw.rect(screen, self.color_choices[self.tube_colors[i][j]],
2                  [offset + 5 + spacing * (i - start), y_start - (50 * j), 65, 50], 0, 3)

```

- Variasi Implementasi: Fungsi digerakkan oleh parameter warna dan posisi:
 - Warna dipilih dari daftar `self.color_choices` berdasarkan elemen tabung.
 - Posisi setiap blok dihitung secara dinamis untuk menciptakan tumpukan dari bawah ke atas.

Polimorfisme Terlihat: Sama-sama menggunakan fungsi `pygame.draw.rect`, tetapi hasilnya berupa blok warna solid yang berbeda sesuai urutan warna di tabung.

2. Menggambar Kerangka Tabung

Tujuan: Menunjukkan batas luar tabung tempat blok warna ditempatkan.

```
1 box = pygame.draw.rect(screen, 'blue',
2                             [offset + 5 + spacing * (i - start), y_start - 150, 65, 200], 5, 5)
3 if selected_index == i:
```

- Variasi Implementasi: Fungsi digerakkan oleh parameter warna (blue) dan ketebalan garis:

- Kerangka tabung terlihat dengan garis tepi biru dan ukuran tetap.
- Posisi dihitung agar tabung tertata rapi secara horizontal dan vertikal.

Polimorfisme Terlihat: Fungsi `pygame.draw.rect` yang sama menghasilkan kerangka dengan garis tepi yang berbeda dibandingkan isi tabung.

3. Menyorot Tabung yang Dipilih

Tujuan: Menunjukkan tabung yang sedang aktif dipilih oleh pemain.

```
1     pygame.draw.rect(screen, 'green',
2                             [offset + 5 + spacing * (i - start), y_start - 150, 65, 200], 3, 5)
3     tube_boxes.append(box)
```

- Variasi Implementasi: Fungsi digerakkan oleh parameter warna (green) dan ketebalan garis:

- Warna hijau digunakan untuk menyorot, dan ketebalan garis disesuaikan agar lebih tipis.
- Posisi tabung yang dipilih ditentukan berdasarkan indeks yang terdeteksi.

Polimorfisme Terlihat: Fungsi yang sama menghasilkan efek sorotan yang mencerminkan interaksi pemain, berbeda dari kerangka biru biasa.

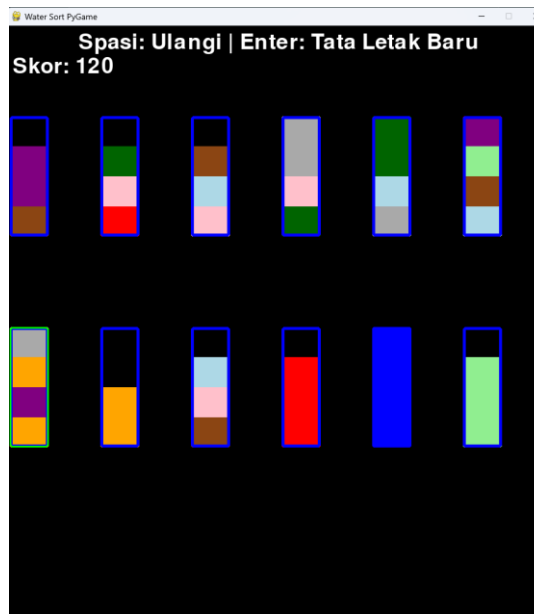
BAB IV

Hasil dan Pembahasan

4.1 Hasil Pengembangan Game

4.1.1 Screenshot dan Demonstrasi Fitur

- Tampilan Saat Bermain



Gambar.2 Tampilan game

Saat bermain, layar menampilkan deretan tabung transparan dengan cairan berwarna acak yang dibagi dalam dua baris. Skor dan instruksi seperti tombol Spasi untuk mengulang serta Enter untuk tata letak baru terlihat di bagian atas. Tabung yang dipilih akan disorot hijau, memudahkan identifikasi. Tampilan sederhana dengan latar gelap membuat warna cairan terlihat mencolok dan permainan lebih menarik.

- Tampilan Saat Menang



Gambar.3 Tampilan menang

Saat pemain menang, layar menampilkan pesan besar "**Anda Menang!**" di tengah dengan latar belakang hitam yang sederhana agar pesan terlihat jelas. Suara kemenangan juga diputar untuk memberikan efek emosional yang memuaskan, menciptakan momen pencapaian yang berkesan.

- Tampilan Saat Kalah



Gambar.4 Tampilan kalah

Jika pemain kalah, layar akan menampilkan pesan besar "**Anda Kalah!**" di tengah dengan latar belakang hitam untuk menonjolkan pesan tersebut. Suara kekalahan diputar, memberikan efek emosional yang menggambarkan kegagalan. Tampilan ini dirancang untuk memberikan umpan balik yang jelas sekaligus mendorong pemain untuk mencoba lagi.

4.2 Cara Bermain

4.2.1 Aturan Permainan:

1. Pemain hanya dapat menuangkan cairan ke tabung kosong atau tabung dengan warna yang sama di bagian atas.
2. Setiap tabung memiliki kapasitas maksimal untuk menampung cairan hingga penuh.

4.2.2 Strategi Bermain:

1. Perhatikan warna di bagian atas setiap tabung untuk merencanakan langkah.
2. Gunakan tabung kosong dengan bijak untuk sementara memindahkan cairan.
3. Hindari gerakan yang membuat warna bercampur atau tidak memungkinkan cairan lain untuk dipindahkan.

4.2.3 Kondisi:

1. Menang: Ketika semua warna berhasil disortir sesuai aturan.
2. Kalah: Jika tidak ada lagi gerakan yang memungkinkan dan cairan belum tersortir.

4.2 Evaluasi Gameplay

4.2.1 Analisis Kelebihan dan Kekurangan

a. Kelebihan Game

1. Aturan permainan mudah dipahami oleh berbagai kalangan, sehingga dapat diakses oleh pemain dengan beragam usia dan latar belakang.
2. Konsep sorting cairan berwarna menawarkan tantangan yang unik dan menyenangkan.
3. Game ini dirancang untuk meningkatkan keterampilan pemain dalam berpikir logis dan menyusun strategi.
4. Pendekatan PBO membuat pengembangan lebih modular, sehingga mudah untuk menambahkan fitur atau memperbaiki bug.
5. Elemen-elemen permainan seperti botol, cairan, dan logika dapat diatur secara terpisah untuk mempermudah pengelolaan.
6. Game ini tidak hanya menghibur, tetapi juga mengedukasi pemain dengan mengasah keterampilan berpikir kritis dan pemecahan masalah.
7. Python adalah bahasa pemrograman yang mudah dipelajari, sehingga game ini bisa menjadi referensi yang baik untuk pemula yang ingin mempelajari pengembangan game.
8. Dengan tingkat kesulitan yang meningkat, game ini memberikan tantangan berkelanjutan yang menjaga minat pemain.

b. Kekurangan Game

1. Gameplay berpotensi menjadi monoton jika tidak ada variasi dalam tantangan atau mekanisme permainan, seperti mode waktu atau mode puzzle khusus.
2. Python memiliki keterbatasan dalam hal performa, sehingga game ini mungkin tidak optimal untuk level yang sangat kompleks atau jumlah objek yang terlalu banyak.
3. Game ini tidak secara langsung mendukung platform seluler, yang menjadi salah satu pasar terbesar untuk game sederhana seperti ini.
4. Pemain yang kurang sabar atau tidak memiliki ketertarikan terhadap permainan logika mungkin kehilangan minat pada game ini.
5. Game ini cenderung fokus pada pemain tunggal, sehingga kurang mendukung elemen sosial seperti kompetisi online atau kolaborasi dengan pemain lain.
6. Pemain mungkin merasa terbatas jika tidak ada opsi untuk kustomisasi, seperti mengganti tema warna atau menambahkan desain botol yang unik.

4.2.2 Evaluasi Keberlanjutan dan Pengembangan

1. Kemudahan Penambahan Fitur Baru

Game ini dirancang dengan pendekatan Pemrograman Berorientasi Objek (PBO), yang memungkinkan pengembang dengan mudah menambahkan fitur baru seperti:

- Tombol undo untuk memperbaiki langkah pemain.
- Sistem hint yang memberikan petunjuk saat pemain menghadapi kesulitan.
- Mode permainan tambahan, seperti mode tantangan waktu.

2. Variasi Tantangan dalam Gameplay

Meskipun game tidak memiliki sistem level, variasi gameplay dapat ditingkatkan dengan:

- Menambahkan jumlah botol atau warna cairan untuk meningkatkan kompleksitas.
- Memperkenalkan aturan atau mekanisme baru, seperti botol khusus yang hanya menerima warna tertentu.
- Menggunakan elemen acak untuk menciptakan tantangan berbeda setiap kali dimainkan.

3. Dukungan Multiplatform

Untuk memperluas jangkauan pengguna, game dapat dikembangkan lebih lanjut agar kompatibel dengan platform lain seperti:

- Perangkat seluler (Android/iOS) menggunakan framework seperti Kivy atau PyGame dengan porting khusus.
- Versi berbasis web dengan konversi menggunakan teknologi seperti Brython atau transpiler Python ke JavaScript.

4. Modularitas dan Skalabilitas Kode

Struktur kode yang modular memudahkan pengembang untuk:

- Mengelola komponen secara terpisah, seperti botol, logika permainan, dan antarmuka.
- Mengintegrasikan fitur baru tanpa memengaruhi bagian lain dari game.
- Melakukan debugging atau pembaruan dengan lebih efisien.

5. Pembaruan Grafis dan Visual

Untuk meningkatkan pengalaman bermain, pengembangan grafis dapat mencakup:

- Animasi cairan saat dituang untuk membuat gameplay lebih dinamis.
- Peningkatan estetika botol dan cairan dengan variasi desain yang menarik.
- Efek transisi atau suara untuk memperkuat interaksi pemain.

6. Kesiapan untuk Pembaruan Berkala

Pembaruan reguler dapat mencakup:

- Penambahan tantangan atau elemen baru untuk menjaga minat pemain.
- Perbaikan bug atau peningkatan performa berdasarkan laporan pengguna.
- Perubahan estetika sesuai tren atau preferensi pengguna.

BAB V

Kesimpulan dan Saran

5.1 Kesimpulan

Dalam pengembangan permainan “Rainbow Riddle”, telah berhasil dirancang dan diimplementasikan sebuah perangkat lunak berbasis Python menggunakan library Pygame. Permainan ini memanfaatkan logika manipulasi tabung warna untuk menciptakan gameplay yang interaktif dan menantang. Berdasarkan analisis kode dan diagram kelas yang telah disusun, beberapa poin penting dapat disimpulkan sebagai berikut:

1. Struktur Perancangan Modular

Sistem permainan dirancang dengan pendekatan berorientasi objek yang modular, di mana setiap kelas memiliki tanggung jawab spesifik, seperti pengelolaan suara (kelas SoundManager), pengelolaan tabung (kelas TubeManager), dan kontrol utama permainan (kelas Game). Pendekatan ini memungkinkan kode yang lebih terstruktur dan mudah untuk dipelihara.

2. Implementasi Prinsip Komposisi dan Ketergantungan

Hubungan antar kelas telah dirancang dengan memperhatikan prinsip komposisi, seperti hubungan antara kelas Game dengan SoundManager dan TubeManager. Selain itu, dependency antar kelas, seperti antara TubeManager dan SoundManager, mempermudah penggunaan kembali kode (reusability).

3. Pemanfaatan Konsep OOP

Kode permainan ini memanfaatkan konsep Object-Oriented Programming (OOP) seperti:

- Polymorphism: Implementasi metode dengan nama yang sama pada berbagai konteks, seperti metode pemutaran suara yang berbeda di SoundManager.
- Encapsulation: Penggunaan atribut dan metode privat dalam kelas untuk menjaga integritas data, seperti pengelolaan warna tabung di TubeManager.
- Abstraction: Penyembunyian detail implementasi tertentu dan hanya menampilkan fungsi penting ke pengguna, seperti antarmuka sederhana untuk memulai dan mengatur ulang permainan pada kelas Game.

4. Fungsi Permainan yang Optimal

Fitur-fitur yang diimplementasikan, seperti efek suara, deteksi kemenangan dan kekalahan, serta pengaturan ulang permainan, berfungsi dengan baik untuk mendukung pengalaman bermain yang menyenangkan dan interaktif.

5. Kesesuaian dengan Tujuan

Permainan ini berhasil memenuhi tujuan utamanya, yaitu memberikan hiburan sambil melatih logika pemain melalui tantangan penyusunan warna di tabung.

5.2 Saran

Meskipun permainan “Rainbow Riddle” telah dirancang dengan baik, terdapat beberapa aspek yang dapat ditingkatkan di masa mendatang untuk memperbaiki kualitas dan memperluas fungsionalitas permainan, yaitu:

1. Peningkatan Visual dan Animasi

- Penggunaan animasi yang lebih halus dan desain antarmuka pengguna yang lebih menarik dapat meningkatkan daya tarik visual permainan.
- Penambahan efek transisi saat memindahkan warna dari satu tabung ke tabung lain akan memberikan pengalaman bermain yang lebih imersif.

2. Pengembangan Tingkat Kesulitan

- Permainan dapat ditingkatkan dengan menambahkan tingkat kesulitan yang lebih kompleks, seperti batas waktu untuk setiap langkah atau pola warna yang lebih menantang.
- Sistem level yang adaptif berdasarkan performa pemain juga dapat meningkatkan replayability.

3. Peningkatan Dukungan Platform

- Mengembangkan permainan agar dapat dimainkan di berbagai platform, seperti perangkat mobile atau melalui browser, akan memperluas jangkauan pengguna.

4. Penambahan Elemen Naratif

- Menambahkan elemen cerita atau karakter yang dapat dimainkan akan meningkatkan daya tarik permainan, khususnya untuk audiens yang menyukai permainan dengan alur cerita.

5. Optimasi Kode

- Melakukan refactoring kode untuk mengurangi kompleksitas dan meningkatkan efisiensi, seperti menggunakan algoritme yang lebih optimal untuk pengecekan kemenangan atau kekalahan.

-

DAFTAR PUSTAKA

1. M. Syafrizal, "Pengantar Jaringan Komputer," Andi, Yogyakarta, 2005.
2. R. Rosandini, "Dampak Permainan Video Game terhadap Kemampuan Kognitif," Jurnal Psikologi, vol. 1, no. 2, pp. 15-20, 2010.
3. A. Giddens, "Runaway World," Profile Books, London, 2003.
4. S. Wiyata, "Kecanduan Game Online di Kalangan Remaja," Wawancara, Jember, 2014.
5. "Water Sort Puzzle - Apps on Google Play," [Online]. Available: https://play.google.com/store/apps/details?id=water.sort.puzzle.liquidsortpuzzle&pcampaignid=web_share

Link Github: <https://github.com/VergiMutiaRahmasyani/UAS-Pemrograman-Berorientasi-Objek>