

ECHO MAZE – GAME LABIRIN BERBASIS PYGAME

UAS

Mata Kuliah
PEMROGRAMAN BERBASIS OBJEK

Oleh
Alya Khoirun Nur Aini
24091397065
2024C



PROGRAM STUDI MANAJEMEN INFORMATIKA
FAKULTAS VOKASI
UNIVERSITAS NEGERI SURABAYA
2025

DAFTAR ISI

DAFTAR ISI	2
BAB I DESKRIPSI APLIKASI	3
1.1 Gambaran Umum	3
1.2 Latar Belakang.....	3
1.3 Konsep dan Tema	3
1.4 Tujuan Aplikasi	3
BAB II KONSEP OOP	4
2.1 Abstraction (Abstraksi)	4
2.2 Encapsulation (Enkapsulasi).....	4
2.3 Inheritance (Pewarisan)	5
2.4 Polymorphism (Polimorfisme)	9
BAB III CLASS DIAGRAM	11
BAB VI FITUR DAN PENGGUNAAN	12
4.1 Fitur Utama Aplikasi <i>Echo Maze</i>	12
4.1.1 Fitur Echo (Sonar)	12
4.1.2 Sistem Tabrakan (Collision Detection)	12
4.1.3 Gate Bergerak (Moving Gate)	13
4.1.4 Sistem Menang dan Kalah	13
4.1.5 Sounds Effects	13
4.2 Informasi UI	14
4.3 Panduan Permainan	14
4.3.1 Menjalankan Aplikasi.....	14
4.3.2 Kontrol Permainan.....	14
4.3.3 Alur Bermain	14
BAB V DOKUMENTASI FUNGSIONALITAS	16

BAB I

DESKRIPSI APLIKASI

1.1 Gambaran Umum

Echo Maze merupakan aplikasi game 2D berbasis Python yang dikembangkan menggunakan library **pygame**. Aplikasi ini dibuat sebagai proyek Ujian Akhir Semester (UAS) mata kuliah **Pemrograman Berorientasi Objek (PBO)** dengan tujuannya menerapkan konsep OOP dalam bentuk aplikasi interaktif.

Game Echo Maze mengusung konsep labirin gelap, di mana pemain harus mencari jalan keluar dengan memanfaatkan kemampuan khusus berupa **echo** atau gelombang suara. Lingkungan permainan tidak sepenuhnya terlihat sejak awal, sehingga pemain dituntut untuk berpikir strategis dalam mengeksplorasi labirin.

1.2 Latar Belakang

Konsep utama permainan ini terinspirasi dari navigasi berbasis suara, seperti cara kerja sonar atau echolocation, jadi pemain harus mengandalkan pantulan suara untuk memahami struktur labirin yang tidak terlihat. Dan penerapannya dalam bentuk labirin terinspirasi dari film labirin bergerak pada serial film fiksi ilmiah distopia *The Maze Runner*.

1.3 Konsep dan Tema

Konsep utama dari Echo Maze adalah navigasi berbasis pantulan suara, menyerupai prinsip **echolocation** atau sonar. Pada awalnya pemain tidak dapat melihat seluruh area labirin secara langsung, melainkan harus mengaktifkan echo (menekan tombol **E**) untuk mengetahui posisi dinding dan jalur sekitarnya. Setiap menabrak dinding labirin juga terdengar suara yang membuat suasa game lebih menantang. Permainan ini juga menerapkan sistem **keterbatasan** baik dalam penggunaan echo maupun tabrakan dinding.

Tema yang diangkat adalah **eksplorasi dan ketegangan**, di mana pemain harus berhati-hati dalam bergerak, mengatur penggunaan echo, serta memperhatikan rintangan dinamis yang dapat berubah seiring waktu.

1.4 Tujuan Aplikasi

Tujuan dari pembuatan aplikasi *Echo Maze* adalah sebagai berikut:

1. Menerapkan konsep **Pemrograman Berorientasi Objek** secara nyata dalam bentuk game
2. Mengembangkan aplikasi sederhana namun memiliki mekanik permainan yang inovatif
3. Melatih kemampuan pengelolaan objek, logika permainan, dan manajemen state.
4. Memberikan pengalaman bermain yang menantang namun mudah dipahami.
5. Mengasah kemampuan mengingat dan mengatur strategi penyelesaian game.

BAB II

KONSEP OOP

Permainan *Echo Maze* dikembangkan menggunakan paradigma *Object Oriented Programming (OOP)*, karena paradigma ini mampu memecah sistem permainan menjadi objek-objek yang saling berinteraksi, sehingga kode menjadi lebih terstruktur, mudah dipahami, serta mudah dikembangkan apabila ada inovasi yang dirasa perlu di tambahkan di kemudian hari. Konsep OOP yang diterapkan meliputi Abstraction, Encapsulation, Inheritance, dan Polymorphism, yaitu ke-empat pilar OOP yang diterapkan secara langsung pada struktur kelas, dan alur program game. Berikut rinciannya:

2.1 Abstraction (Abstraksi)

Abstraction adalah konsep OOP yang bertujuan untuk menyederhanakan sistem dengan cara menampilkan fitur-fitur penting dan menyembunyikan detail implementasi yang kompleks. Contohnya pada method bergerak di class mobil, maka sistem tidak akan menampilkan bagaimana mobil itu bergerak, tetapi hanya mengetahui atau menampilkan bahwa “Mobil bisa bergerak” tanpa peduli detail teknisnya. Jadi dengan abstraksi ini, pengguna dan pengembang tidak perlu mengetahui atau memusingkan bagaimana suatu proses bekerja secara detail, melainkan cukup mengetahui apa fungsi dari objek tersebut.

Dalam game *Echo Maze*, abstraksi diterapkan melalui pembagian peran kelas sebagai berikut:

- Kelas Player → Mengatur seluruh perilaku pemain seperti pergerakan, tabrakan, dan batas kesalahan (hit).
- Kelas Maze → Mengelola struktur labirin, dinding, serta posisi pintu keluar.
- Kelas Echo → Mengatur mekanisme sonar yang digunakan pemain untuk melihat labirin sementara.
- Kelas MovingGate → Mengatur rintangan tambahan berupa gate yang muncul dan menghilang.
- Kelas Game → Mengatur alur permainan secara keseluruhan, mulai dari input, update, hingga kondisi menang atau kalah.

Dengan pembagian ini, setiap kelas hanya fokus pada satu tanggung jawab utama (single responsibility), sehingga kompleksitas sistem dapat dikontrol dengan baik.

2.2 Encapsulation (Enkapsulasi)

Enkapsulasi merupakan konsep OOP yang menggabungkan data dan method yang berkaitan ke dalam satu kelas, serta membatasi akses langsung terhadap data tersebut. Tujuan utamanya adalah untuk menjaga keamanan data dan memastikan **perubahan data hanya terjadi melalui mekanisme yang telah ditentukan.**

Pada permainan *Echo Maze*, enkapsulasi diterapkan dengan cara:

- a. Penyimpanan atribut di dalam kelas
 - **hit_count**, **max_hit**, dan **hit_cooldown** disimpan di dalam kelas **Player**.

- **radius**, **max_radius**, dan **echo_used** disimpan di dalam kelas **Echo**.
- b. Pengubahan data melalui method
- Jumlah tabrakan (**hit_count**) hanya bertambah melalui method **handle_hit()**

```

1 def handle_hit(self): # Method untuk menangani tabrakan player dengan dinding/gate
2     # Hit hanya dihitung jika cooldown sudah habis (untuk menghindari double-hit count)
3     if self.hit_cooldown <= 0:
4         hit_sound.play() # Mainkan sound tabrakan
5         self.hit_count += 1 # Tambah jumlah hit
6         self.hit_cooldown = 0.5 # Set cooldown 0.5 detik

```

- Penggunaan echo hadnya dapat dilakukan melalui method **trigger()**

```

1 def trigger(self, x, y): # Method untuk mengaktifkan echo pada posisi tertentu
2     # Echo hanya bisa dipakai jika belum mencapai batas
3     if self.echo_used < self.max_echo:
4         self.x, self.y = x, y # Set posisi echo ke posisi player
5         self.radius = 0 # Reset radius echo
6         self.active = True # Aktifkan echo
7         self.echo_used += 1 # Tambah jumlah echo yang sudah dipakai
8         echo_sound.play() # Mainkan sound echo

```

Sebagai contoh → pemain tidak dapat menambah jumlah hit secara langsung dari luar kelas **Player**. Karena semua perubahan dikontrol melalui method internal sehingga logika permainan tetap konsisten dan terhindar dari kesalahan manipulasi data.

2.3 Inheritance (Pewarisan)

Inheritance atau pewarisan merupakan konsep OOP yang memungkinkan sebuah kelas mewarisi atribut dan method dari kelas lain. Konsep ini digunakan untuk menghindari duplikasi kode saat menciptakan struktur kelas yang hierarkis. Di dalam konsep pewarisan ini ada dua jenis kelas utama yaitu kelas induk atau superclass yang berperan sebagai kelas utama yang bisa mewariskan atribut dan methodnya ke kelas turunannya, dan kelas anak atau subclass yang merupakan turunan dari kelas induk, kelas ini mewarisi attribute dan method dari kelas induk. Tetapi, subclass juga bisa memiliki kelas turunan dan menurunkan methodnya ke kelas tersebut.

Dalam permainan **Echo Maze**, inheritance diterapkan melalui kelas induk bernama **GameObject**. Kelas ini berfungsi sebagai blueprint umum bagi seluruh objek di dalam game. Berikut rinciannya:

Kelas Induk : GameObject

```

1  class GameObject:
2      """
3      Class induk untuk semua objek dalam game.
4      Class ini menyimpan posisi x dan y yang akan diwarisi oleh class turunan.
5      """
6
7      def __init__(self, x, y):
8          # Constructor untuk menginisialisasi posisi objek
9          self.x = x          # Posisi horizontal (kiri-kanan) objek
10         self.y = y          # Posisi vertikal (atas - bawah) objek
11
12     def update(self, dt):
13         # Method untuk memperbarui state objek setiap frame
14         # dt = delta time (waktu yang berlalu sejak frame terakhir)
15         # Method ini akan di-OVERRIDE oleh class turunan
16         pass
17
18     def draw(self, screen):
19         # Method untuk menggambar objek ke layar
20         # screen = surface pygame tempat objek digambar
21         # Method ini akan di-OVERRIDE oleh class turunan
22         pass

```

Atribut dan method dasar yang dimiliki:

- x dan y → posisi objek
- update () → method dasar untuk logika perframe
- draw() → method dasar untuk menggambar objek

Kelas Turunan dari GameObject

- a. Player

Source code:

```

1 # CLASS PLAYER
2 class Player(GameObject):
3     """
4     Class Player merupakan Turunan dari GameObject (Inheritance)
5     Class Player mengatur pergerakan pemain, tabrakan, dan jumlah hit.
6     """
7
8     def __init__(self, x, y):
9         super().__init__(x, y) # Memanggil constructor GameObject (kelas induk)
10
11         self.size = 30 # Ukuran kotak player
12         self.speed = 150 # Kecepatan gerak player
13         # Membuat rectangle untuk collision detection
14         self.rect = pygame.Rect(x, y, self.size, self.size) # Rect(x, y, width, height) - posisi dan ukuran player
15
16         self.hit_count = 0 # Jumlah tabrakan
17         self.max_hit = 5 # Batas maksimal tabrakan
18         self.hit_cooldown = 0 # Cooldown agar hit tidak berlipat
19
20     def move(self, keys, maze, gates, dt):
21         # Method untuk menggerakkan player berdasarkan input keyboard
22         # keys = status tombol keyboard yang ditekan
23         # maze = objek labirin untuk cek tabrakan dengan dinding
24         # gates = list gate bergerak untuk cek tabrakan
25         # dt = delta time untuk pergerakan yang smooth
26         if self.hit_cooldown > 0:
27             self.hit_cooldown -= dt # --> Mengurangi cooldown hit setiap frame
28
29         # Variabel untuk menyimpan perubahan posisi (delta x, delta y)
30         dx, dy = 0, 0
31
32         # Mengecek tombol panah kiri - gerak ke kiri
33         if keys[pygame.K_LEFT]:
34             dx = -self.speed * dt # Negatif = ke kiri (dikali dt agar smooth)
35
36         # Mengecek tombol panah kanan - gerak ke kanan
37         if keys[pygame.K_RIGHT]:
38             dx = self.speed * dt # Positif = ke kanan
39
40         # Mengecek tombol panah atas - gerak ke atas
41         if keys[pygame.K_UP]:
42             dy = -self.speed * dt # Negatif = ke atas
43
44         # Mengecek tombol panah bawah - gerak ke bawah
45         if keys[pygame.K_DOWN]:
46             dy = self.speed * dt # Positif = ke bawah
47
48         # Menyimpan posisi lama (untuk rollback jika tabrakan)
49         old_x, old_y = self.rect.x, self.rect.y
50
51         # Gerakan horizontal
52         self.rect.x += dx # Tambahkan perubahan posisi horizontal
53         # Cek apakah player menabrak dinding ATAU menabrak salah satu gate
54         if maze.is_wall(self.rect) or any(g.collide(self.rect) for g in gates):
55             self.rect.x = old_x # Kembalikan ke posisi lama (rollback)
56             self.handle_hit() # Proses tabrakan (tambah hit counter + sound)
57
58         # Gerakan vertikal
59         self.rect.y += dy # Tambahkan perubahan posisi vertikal
60         # Cek tabrakan vertikal dengan dinding atau gate
61         if maze.is_wall(self.rect) or any(g.collide(self.rect) for g in gates):
62             self.rect.y = old_y # Kembalikan ke posisi lama
63             self.handle_hit() # Proses tabrakan
64
65         # Sinkronisasi posisi GameObject dengan posisi rect
66         self.x, self.y = self.rect.x, self.rect.y
67
68     def handle_hit(self): # Method untuk menangani tabrakan player dengan dinding/gate
69         # Hit hanya dihitung jika cooldown sudah habis (untuk menghindari double-hit count)
70         if self.hit_cooldown <= 0:
71             hit_sound.play() # Mainkan sound tabrakan
72             self.hit_count += 1 # Tambah jumlah hit
73             self.hit_cooldown = 0.5 # Set cooldown 0.5 detik
74
75     def draw(self, screen):
76         # Method untuk menggambar player sebagai kotak biru di layar
77         # pygame.draw.rect(surface, color, rect) - gambar rectangle
78         pygame.draw.rect(screen, BLUE, self.rect)

```

b. Echo

Source code :

```

1  # CLASS ECHO
2  class Echo(GameObject):
3      """
4      Class Echo turunan dari kelas GameObject (INHERITANCE)
5      mewarisi atribut dan method GameObject
6      --> Class ini mengatur sonar pemain yang memungkinkan player melihat labirin
7      dalam radius tertentu untuk waktu terbatas.
8      """
9
10     def __init__(self, x, y):
11         # Memanggil constructor kelas induk GameObject
12         super().__init__(x, y)
13
14         self.radius = 0          # Radius echo saat ini (awal = 0)
15         self.max_radius = 200    # Radius maksimal echo
16         self.active = False      # Status echo (True = aktif atau False = tidak)
17
18         self.max_echo = 5        # Batas maksimal penggunaan echo
19         self.echo_used = 0       # Jumlah echo yang sudah digunakan
20
21     def trigger(self, x, y):      # Method untuk mengaktifkan echo pada posisi tertentu
22         # Echo hanya bisa dipakai jika belum mencapai batas
23         if self.echo_used < self.max_echo:
24             self.x, self.y = x, y    # Set posisi echo ke posisi player
25             self.radius = 0          # Reset radius echo
26             self.active = True       # Aktifkan echo
27             self.echo_used += 1      # Tambah jumlah echo yang sudah dipakai
28             echo_sound.play()        # Mainkan sound echo
29
30     def update(self, dt):
31         # Method memperbarui radius echo setiap frame
32
33         # Jika echo aktif
34         if self.active:
35             self.radius += 300 * dt  # perbesar radiusnya
36
37         # Jika radius sudah mencapai atau melebihi max_radius
38         if self.radius >= self.max_radius:
39             self.active = False      # Nonaktifkan echo
40
41     def visible(self, x, y, size):
42         # Method untuk mengecek apakah objek berada dalam radius echo
43         # Hitung koordinat pusat objek
44         cx = x + size / 2          # Koordinat x pusat objek
45         cy = y + size / 2          # Koordinat y pusat objek
46         # Return True jika echo aktif DAN jarak <= radius
47         return self.active and math.hypot(cx - self.x, cy - self.y) <= self.radius
48
49     def draw(self, screen):
50         # Method untuk menggambar lingkaran echo
51         if self.active:            # Hanya menggambar jika echo aktif
52             # pygame.draw.circle(surface, color, center, radius, width)
53             # width=1 artinya hanya gambar garis tepi (bukan lingkaran penuh)
54             pygame.draw.circle(screen, WHITE, (int(self.x), int(self.y)), int(self.radius), 1)

```

c. MovingGate

Source code:

```

1 # CLASS MOVING GATE
2
3 class MovingGate(GameObject):
4     """
5     Gate yang muncul (aktif) dan menghilang (inkatif) secara berkala
6     untuk menambah tantangan dengan obstacle dinamis.
7     """
8
9     def __init__(self, x, y, w, h):
10         # Memanggil constructor GameObject
11         super().__init__(x, y)
12         self.rect = pygame.Rect(x, y, w, h) # Membuat rectangle untuk posisi dan ukuran gate
13         self.timer = 0 # Timer untuk mengatur interval aktif/inaktif
14         self.active = True # Status gate (True = terlihat)
15
16     def update(self, dt):
17         # Method untuk memperbarui status gate setiap frame
18         self.timer += dt # Tambah timer dengan delta time
19         if self.timer >= 1.2: # Setiap 1.2 detik
20             self.active = not self.active # Toggle status aktif/inaktif
21             self.timer = 0 # Reset timer
22
23     def collide(self, rect):
24         # Method untuk mengecek tabrakan dengan player
25         # Return True jika gate aktif dan rect bertabrakan dengan gate
26         return self.active and self.rect.colliderect(rect)
27
28     def draw(self, screen, echo):
29         # Method untuk menggambar gate di layar
30         # Gambar gate hanya jika aktif DAN terlihat oleh echo
31         if self.active and echo.visible(self.rect.x, self.rect.y, self.rect.width):
32             # Gambar rectangle merah
33             pygame.draw.rect(screen, RED, self.rect)

```

2.4 Polymorphism (Polimorfisme)

Polimorphism adalah konsep OOP yang memungkinkan satu method dengan nama yang sama tetapi memiliki implementasi yang berbeda tergantung pada objek yang memanggilnya. Konsep ini sangat penting dalam pengembangan game karena banyak objek memiliki fungsi serupa namun perilaku berbeda.

Di dalam game ini, polimorfisme ada pada method:

- **update()**
- **draw()**

Walaupun method tersebut memiliki nama yang sama tetapi implementasinya berbeda di setiap kelas, berikut rinciannya:

- Player.draw()** → menggambar karakter pemain.

```

echo_maze.py > Player
85 class Player(GameObject):
158     def draw(self, screen):
159         # Method untuk menggambar player sebagai kotak biru di layar
160         # pygame.draw.rect(surface, color, rect) - gambar rectangle
161         pygame.draw.rect(screen, BLUE, self.rect)
162

```

- Echo.draw()** → menggambar lingkaran sonar.

```

echo_maze.py > Player
227 class Echo(GameObject):
274     def draw(self, screen):
275         # Method untuk menggambar lingkaran echo
276         if self.active: # Hanya menggambar jika echo aktif
277             # pygame.draw.circle(surface, color, center, radius, width)
278             # width=1 artinya hanya gambar garis tepi (bukan lingkaran penuh)
279             pygame.draw.circle(screen, WHITE, (int(self.x), int(self.y)), int(self.radius), 1)

```

- c. **MovingGate.draw()** → menggambar gate yang aktif.

```

echo_maze.py > Player
283 class MovingGate(GameObject):
307
308     def draw(self, screen, echo):
309         # Method untuk menggambar gate di layar
310         # Gambar gate hanya jika aktif DAN terlihat oleh echo
311         if self.active and echo.visible(self.rect.x, self.rect.y, self.rect.width):
312             # Gambar rectangle merah
313             pygame.draw.rect(screen, RED, self.rect)
314

```

- d. **Game.draw()** → menggambar seluruh elemen permainan.

```

echo_maze.py > Player
317 class Game:
374     def draw(self):
375         # Method untuk menggambar semua elemen game ke layar
376
377         # Isi layar dengan warna hitam (clear screen)
378         screen.fill(BLACK)
379
380         # Gambar maze (dinding dan exit)
381         self.maze.draw(screen, self.echo)
382
383         # Gambar semua gate
384         for g in self.gates:
385             g.draw(screen, self.echo)
386
387         # Gambar player (kotak biru)
388         self.player.draw(screen)
389
390         # Gambar echo (lingkaran putih)
391         self.echo.draw(screen)
392
393         # === GAMBAR UI (HIT COUNTER & ECHO COUNTER) ===
394         # Buat objek font dengan file custom dan ukuran 25
395         font = pygame.font.Font("PixelOperatorSC-Bold.ttf", 25)
396
397         # Render teks hit counter (putih)
398         # f-string untuk format: "Hits: 3/5" (contoh)
399         screen.blit(font.render(f"Hits: {self.player.hit_count}/5", True, WHITE), (10, 10))
400
401         # Render teks echo counter (putih)
402         # Posisi (10, 30) - di bawah hit counter
403         screen.blit(font.render(f"Echo: {self.echo.echo_used}/5", True, WHITE), (10, 30))
404

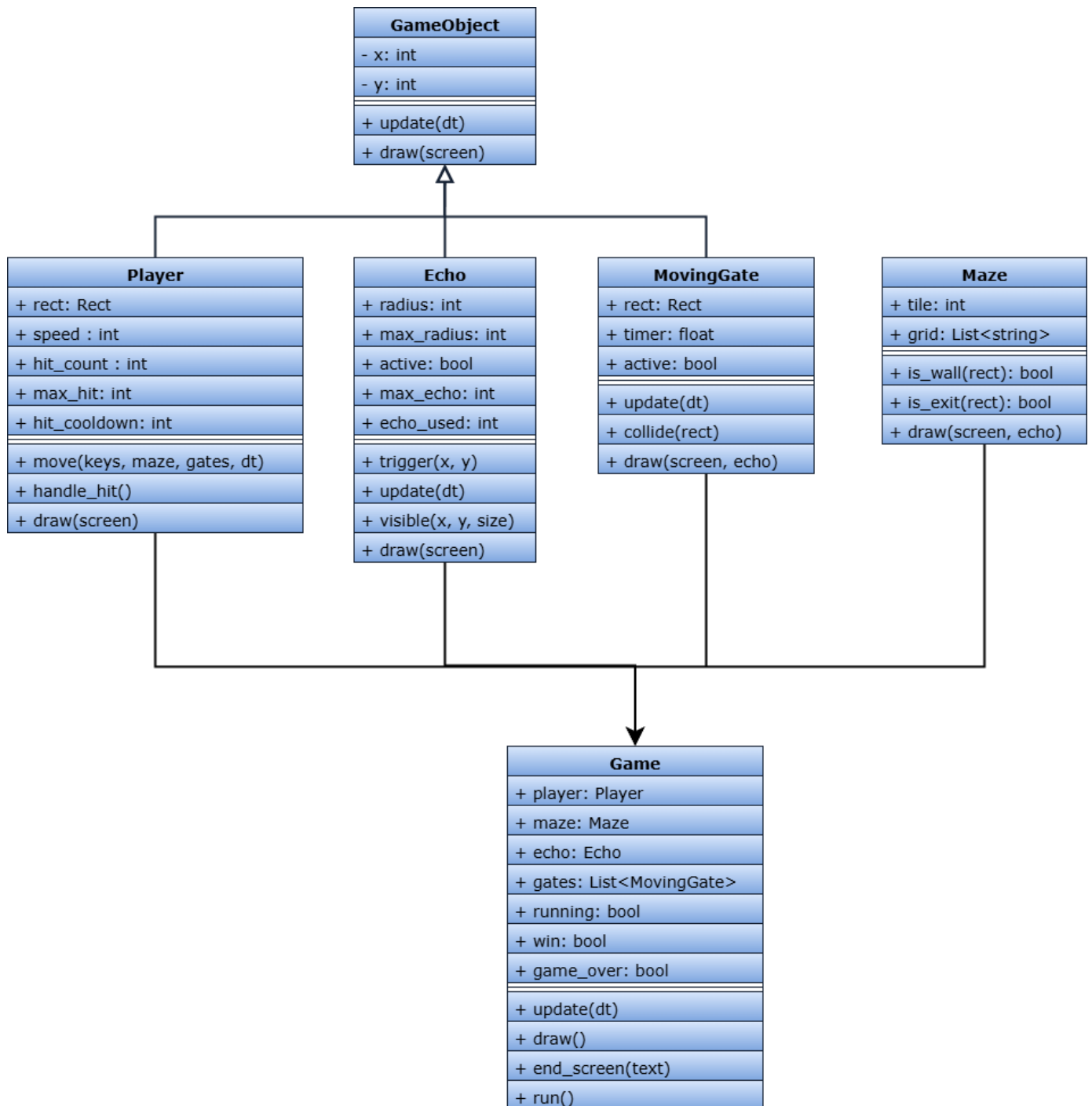
```

Dengan adanya polimorfisme, program dapat memanggil method yang sama tanpa perlu mengetahui jenis objek secara spesifik, sehingga kode menjadi lebih fleksibel dan mudah dikembangkan.

BAB III

CLASS DIAGRAM

Diagram class pada aplikasi *Echo Maze* menggambarkan struktur kelas, atribut, method, serta hubungan antar kelas yang digunakan dalam pengembangan aplikasi. Diagram ini menunjukkan penerapan konsep *Object Oriented Programming* (OOP) secara nyata melalui hubungan inheritance dan asosiasi antar objek.



BAB VI

FITUR DAN PENGGUNAAN

4.1 Fitur Utama Aplikasi *Echo Maze*

Aplikasi *Echo Maze* merupakan permainan labirin berbasis pygame yang mengandalkan mekanisme sonar (*echo*) sebagai fitur utama untuk membantu pemain melihat dinding labirin yang awalnya tidak terlihat. Game ini dirancang tidak hanya sebagai hiburan, tetapi juga sebagai implemementasi nyata konsep OOP, dan untuk mengasah strategi dan melatih daya ingat pemain.

4.1.1 Fitur Echo (Sonar)

Fitur echo memungkinkan pemain untuk “melihat” dinding labirin dalam radius tertentu untuk waktu yang terbatas.

Karakteristik fitur echo:

- Echo diaktifkan dengan menekan tombol E.
- Echo berbentuk lingkaran putih yang menyebar dari posisi pemain.
- Echo hanya dapat digunakan maksimal 5 kali selama permainan.
- Dinding dan pintu keluar hanya akan terlihat ketika berada di dalam radius echo.
- Setiap penggunaan echo disertai efek suara (sound effect)

Fitur ini melatih pemain untuk:

- Mengatur strategi penggunaan echo
- Mengingat pola labirin
- Mengelola sumber daya terbatas

Cara kerja:

- Tekan tombol **E** untuk mengaktifkan echo
- Ketika echo aktif maka suara **echo.mp3** akan terdengar
- Echo akan menyebar dari posisi pemain dalam bentuk lingkaran
- Radius echo akan membesar hingga 200 pixel
- Semua dinding dan objek dalam radius tersebut akan terlihat sementara atau dalam kurun waktu echo aktif.

Batasan:

- Hanya bisa digunakan **5 kali** dalam setiap permainan.
- Harus digunakan dengan bijak untuk mengingat jalur labirin.

Visual: lingkaran putih yang membesar dari posisi player (player akan secara otomatis berada di tengah lingkaran echo yang berupa border bukan lingkaran penuh dan lingkaran ini semakin membesar hingga radius 200 px)

4.1.2 Sistem Tabrakan (Collision Detection)

Game menerapkan sistem deteksi tabrakan antara pemain dengan:

- Dinding labirin
- Gate (pintu) bergerak

Aturan tabrakan:

- Setiap tabrakan akan menambah hit counter +1 dan sound effect **hit.mp3** akan otomatis dimainkan.
- Player akan berhenti (tidak bisa menembus)
- Terdapat cooldown hit selama 0.5 detik agar satu tabrakan tidak dihitung berkali-kali.
- Maksimal tabrakan yang diperbolehkan adalah 5 kali.
- Jika jumlah hit melebihi batas, maka permainan berakhir (Game Over).

Visibility: hanya terlihat ketika terkena echo

4.1.3 Gate Bergerak (Moving Gate)

Gate bergerak merupakan tantangan tambahan yang bersifat dinamis..

Karakteristik gate bergerak:

- Gate akan muncul dan menghilang secara berkala (setiap ± 1.2 detik).
- Gate dapat menutup jalur yang sebelumnya terbuka.
- Gate hanya terlihat ketika terkena echo.
- Gate dapat menyebabkan tabrakan jika aktif dan disentuh pemain.

Fitur ini meningkatkan tingkat kesulitan dan membuat permainan lebih menantang dibandingkan labirin statis.

Strategi yang diharapkan: memperhatikan timing untuk melewati jalan ketika gate sedang hilang.

4.1.4 Sistem Menang dan Kalah

Game memiliki dua kondisi akhir permainan:

- Kondisi Menang (Win Condition)
 - Pemain berhasil mencapai pintu keluar yang ditandai dengan huruf E (di code) dan kotak berwarna hijau (di labirin dan hanya bisa terlihat jika terkena echo).
 - Sistem akan menampilkan pesan: "YOU ESCAPED THE MAZE!"
 - Efek suara kemenangan (win sound) akan diputar.
- Kondisi Kalah (Game Over)
 - Pemain menabrak dinding atau gate lebih dari 5 kali.
 - Sistem akan menampilkan pesan: "GAME OVER – TOO MANY HITS"
 - Efek suara game over akan diputar.

4.1.5 Sounds Effects

- Echo Sound: Suara saat mengaktifkan echo (50% volume)
- Hit Sound: Suara saat menabrak dinding/gate (60% volume)
- Win Sound: Suara saat berhasil menang (70% volume)
- Game Over Sound: Suara saat kalah (90% volume)

4.2 Informasi UI

a. Di layar game

- Hits: 3/5 ← Jumlah tabrakan
- Echo: 2/5 ← Jumlah echo yang tersisa
- Area labirin akan terlihat jalurnya jika menekan tombol E (mengaktifkan echo)

b. End Screen

- Menang → kondisi ketika player berhasil mencapai pintu keluar yaitu kotak hijau: Teks hijau "YOU ESCAPED THE MAZE!"
- Kalah → kondisi ketika user melebihi batas maksimal menabrak dinding yaitu 5 kali: Teks merah "GAME OVER - TOO MANY HITS"
- Sistem akan otomatis quit atau mati ketika melebihi 10 dari kondisi akhir, tetapi jika user mengklik tombol esc maka aplikasi akan langsung mati.

4.3 Panduan Permainan

4.3.1 Menjalankan Aplikasi

Langkah-langkah menjalankan game:

1. Pastikan Python dan Pygame sudah terinstal.
2. Pastikan file suara (echo.mp3, hit.mp3, win.mp3, game_over.mp3) dan font berada dalam satu folder dengan file .py.
3. Jalankan program
4. Jendela game akan otomatis muncul dan siap dijalankan.

4.3.2 Kontrol Permainan

Tombol	Fungsi
↑ (Panah Atas)	Gerak ke atas
↓ (Panah Bawah)	Gerak ke bawah
← (Panah Kiri)	Gerak ke kiri
→ (Panah Kanan)	Gerak ke kanan
E	Aktifkan Echo (Sonar)
ESC	Keluar dari end screen
X (Close Button)	Tutup game

4.3.3 Alur Bermain

1. Pemain memulai permainan di posisi awal labirin.
2. Layar gelap dan dinding tidak terlihat secara langsung.
3. Pemain menekan tombol E untuk memunculkan echo dan melihat struktur labirin.
4. Pemain mengingat jalur yang terlihat dan bergerak menuju pintu keluar.

5. Pemain harus menghindari dinding dan gate bergerak.
6. Permainan berakhir ketika:
 - Pemain berhasil mencapai pintu keluar (menang), atau
 - Pemain terlalu sering menabrak dinding/gate (kalah).

Flownya seperti berikut:

START



GAME DIMULAI



Player spawn di (60, 60)



[LOOP GAME]

- |— Gerakkan player dengan arrow keys
- |— Gunakan echo (E) untuk melihat labirin
- |— Hindari dinding dan moving gate
- |— Cari pintu exit (kotak hijau)
- |— Pantau hit counter (max 5)



KONDISI AKHIR?

- |— [Sampai Exit] → YOU WIN : "YOU ESCAPED THE MAZE!"
- |— [5 Hits] → GAME OVER : "GAME OVER – TOO MANY HITS"



END SCREEN (10 detik atau tekan ESC)

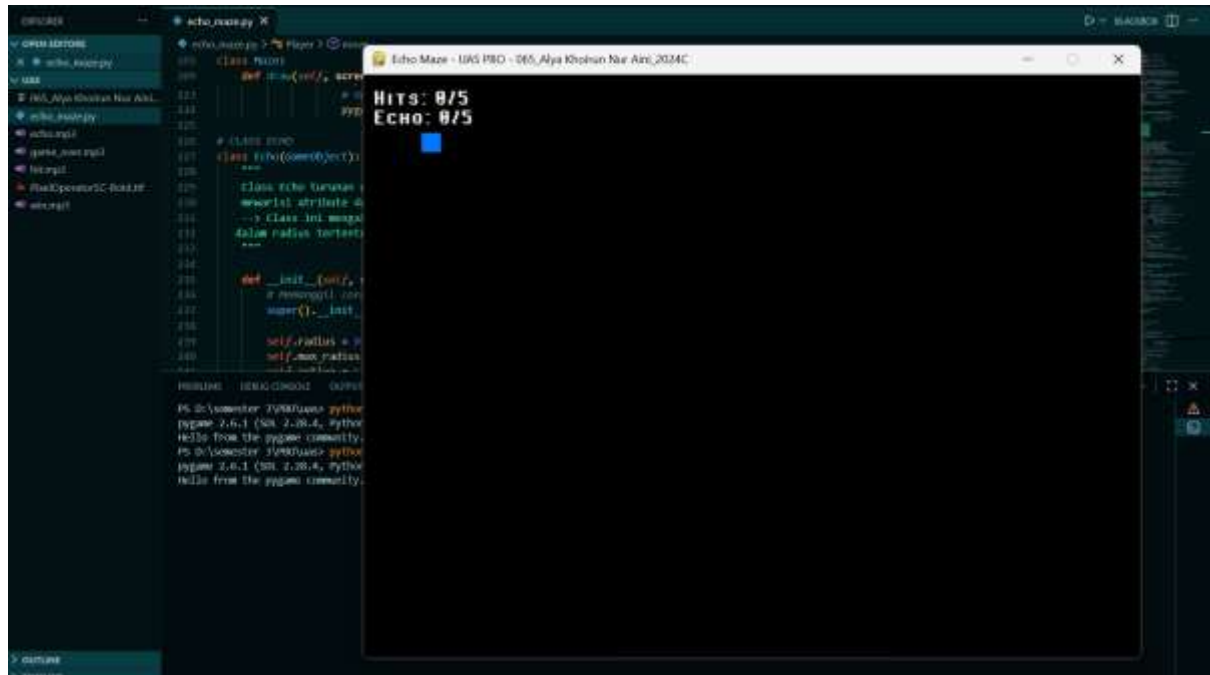


GAME SELESAI

BAB V

DOKUMENTASI FUNGSIONALITAS

- Tampilan Awal

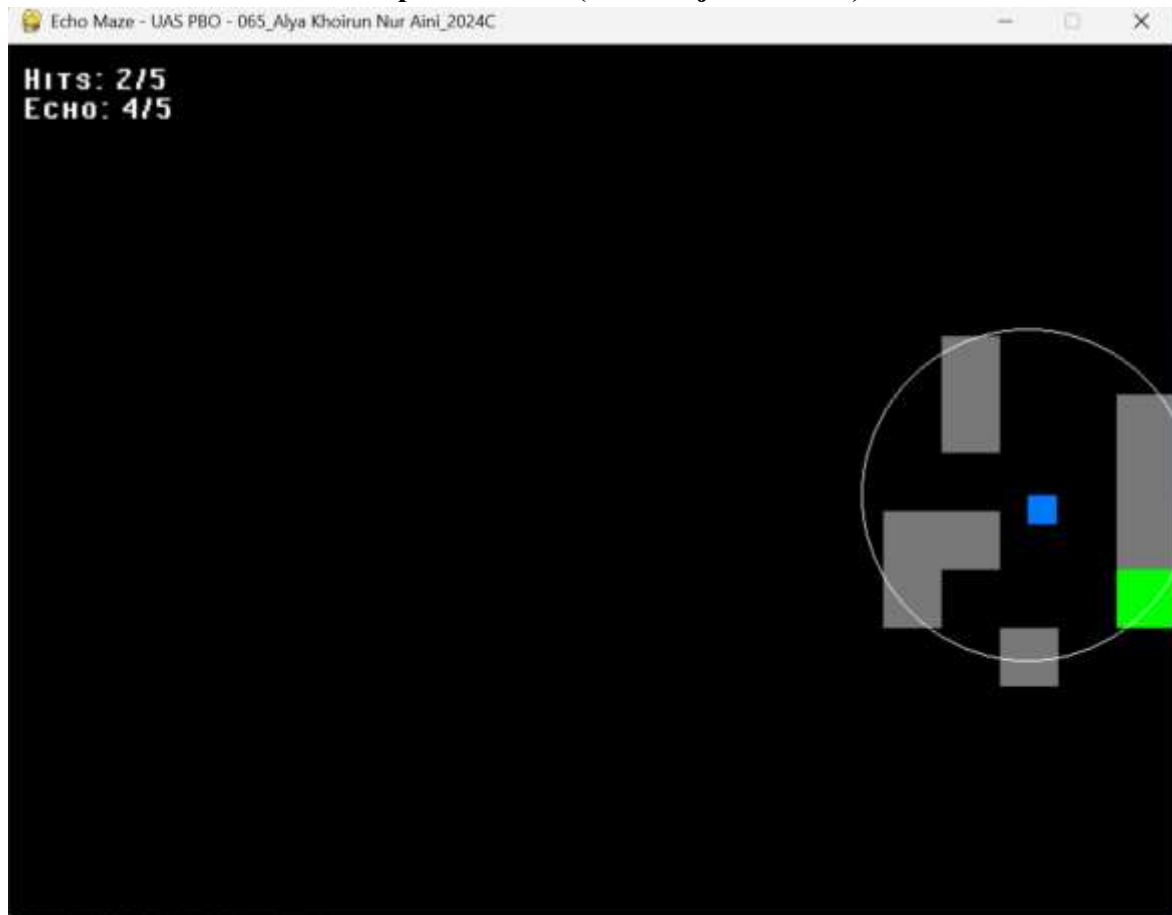


- Ketika menabrak dan di tekan E (echo diaktifkan)



Yang berwarna merah merupakan gate yang dapat bergerak (MovingGate) sedangkan yang berwarna abu-abu adalah dinding dtatis labirin.

- **Kondisi membuka echo dan pintu keluar (kotak hijau terlihat)**



- **Ketika game over → kondisi menabrak lebih dari 5**



- Kondisi ketika win → user berhasil mencapai pintu keluar (kotak warna hijau di labirin)

