

TUGAS AKHIR UAS
GHOST - ESCAPE THE LAB

Mata Kuliah:
Pemrograman Berbasis Objek



Disusun Oleh Kelompok 3 (2024A):

- | | |
|------------------------------|-------------|
| 1. MOHAMMAD MAULANA RIYAN W. | 24091397011 |
| 2. DIMAS ADITYA PRATAMA | 24091397014 |
| 3. MILA CAHAYA RANI | 24091397072 |

PROGRAM STUDI D4 MANAJEMEN INFORMATIKA
FAKULTAS VOKASI
UNIVERSITAS NEGERI SURABAYA
TAHUN 2025

DAFTAR ISI

DAFTAR ISI.....	II
BAB I PENDAHULUAN	1
1.1 Latar Belakang Proyek.....	1
1.2 Deskripsi Game	1
1.3 Tujuan Permainan.....	1
1.4 Fitur-Fitur Utama	2
1.5 Penerapan Konsep OOP	3
1.6 Keunikan dan Kreativitas Game	3
BAB II PERANCANGAN DAN DOKUMENTASI SISTEM.....	4
2.1 Perancangan	4
2.2 Implementasi Sistem (Source Code).....	10
BAB III HASIL PROYEK.....	33
3.1 Tampilan Game	33
BAB IV PENUTUP	38
4.1 Kesimpulan.....	38
4.2 Saran dan Pengembangan Lanjutan	38

BAB I

PENDAHULUAN

1.1 Latar Belakang Proyek

Dalam era digital saat ini, pengembangan aplikasi berbasis permainan (game) menjadi salah satu media yang efektif untuk mempelajari dan mengimplementasikan konsep-konsep pemrograman, khususnya Pemrograman Berorientasi Objek (OOP). Game tidak hanya menawarkan pengalaman interaktif yang menarik, tetapi juga memungkinkan pengembang untuk merancang sistem yang kompleks dengan struktur kode yang rapi dan mudah untuk dikembangkan. Proyek “Ghost - Escape The Lab” ini dibuat sebagai tugas akhir mata kuliah Pemrograman Berbasis Objek dengan tujuan utama untuk menerapkan prinsip-prinsip OOP secara nyata dalam sebuah aplikasi game 2D yang utuh.

Pemilihan tema laboratorium futuristik yang terbengkalai dengan elemen escape dan survival dipilih karena mampu menggabungkan mekanisme maze-runner seperti Pac-Man dengan nuansa modern sci-fi. Tema ini memberikan ruang yang luas untuk mengimplementasikan berbagai fitur menarik seperti navigasi labirin, musuh yang berbasis kecerdasan buatan, sistem pengumpulan item, serta power-up sementara. Selain itu, pengembangan game ini juga menjadi wadah untuk mengeksplorasi fitur tambahan seperti sistem penyimpanan progres, toko kustomisasi skin, leaderboard, dan narasi lore melalui jurnal rahasia, sehingga proyek tidak hanya memenuhi persyaratan akademis, tetapi juga menghasilkan sebuah permainan yang lengkap.

1.2 Deskripsi Game

Judul : Ghost - Escape The Lab
Jenis Aplikasi : Game 2D Top-Down berbasis Python dan Pygame
Tema : Eksperimen laboratorium futuristik
Genre : Maze-runner

Escape The Lab adalah sebuah game 2D yang berfokus pada navigasi dalam sebuah lab penelitian yang sudah terbengkalai. Pemain berperan sebagai subjek eksperimen yang mencoba melarikan diri dengan mengumpulkan tiga buah Energy Chips yang tersebar di dalam maze. Sepanjang permainan, pemain harus menghindari robot penjaga yang berpatroli dan dapat mengejar pemain jika terdeteksi. Game ini dirancang untuk menampilkan penerapan konsep OOP seperti encapsulation, inheritance, dan polymorphism secara terstruktur.

1.3 Tujuan Permainan

- Mengumpulkan seluruh Energy Chips yang terdapat dalam area permainan.
- Menghindari musuh yang berpatroli dan mengejar pemain.
- Mencapai titik akhir dengan kondisi hidup dan semua item terkumpul.

1.4 Fitur-Fitur Utama

1. Player Movement

Pemain dapat bergerak ke empat arah (atas, bawah, kiri, kanan). Terdapat mekanisme collision terhadap dinding maze untuk mencegah pemain bergerak ke area terlarang. Kecepatan pemain dapat berubah tergantung kondisi power-up.

2. Maze System

- Maze dibangun menggunakan struktur grid 2D berbasis list.
- Setiap tile memiliki jenis tertentu: tembok, jalur, atau titik spawn.
- Maze dirancang agar mendukung sistem navigasi musuh dan jalur pemain.

3. Enemy AI

Musuh memiliki dua mode utama:

a. Patrol Mode

Musuh bergerak mengikuti rangkaian waypoint secara berulang.

b. Chase Mode

Musuh memasuki mode pengejaran apabila pemain berada dalam radius deteksi tertentu. Ketika pemain berhasil menjauh, musuh kembali ke mode patroli.

Enemy AI diimplementasikan menggunakan konsep inheritance dan polymorphism untuk membedakan perilaku antar kelas musuh.

4. Collectible Items

Terdapat tiga buah Energy Chips yang ditempatkan pada titik tetap.

Saat pemain mengambil item ini:

- Skor pemain bertambah.
- Menjadi salah satu syarat kemenangan.

5. Power-Up System

Power-up muncul pada titik tetap dalam maze dan dapat diambil oleh pemain untuk memberikan efek sementara.

Beberapa jenis power-up:

a. Speed Boost

Meningkatkan kecepatan pemain selama durasi tertentu.

b. Immune Shield

Membuat pemain kebal terhadap serangan musuh dalam waktu singkat.

c. (Opsional) Time Slow

Mengurangi kecepatan gerakan musuh.

Setiap power-up diimplementasikan sebagai subclass yang mengoverride method apply() dan update(), sehingga memanfaatkan polymorphism pada OOP.

6. Win/Lose Condition

- Menang: Semua Energy Chips berhasil dikumpulkan.
- Kalah: Pemain tertangkap musuh ketika tidak memiliki efek Immune Shield.\\"

7. HUD dan User Interface

- Menampilkan jumlah Energy Chips yang telah dikumpulkan.
- Menampilkan indikator power-up aktif.
- Menyediakan tampilan menu awal, gameplay, serta layar kemenangan dan kekalahan.

1.5 Penerapan Konsep OOP

- a. Encapsulation
 - Variabel penting seperti posisi, kecepatan, status immune, dan timer power-up menggunakan akses private.
 - Atribut diakses dan dimodifikasi melalui getter dan setter sesuai kebutuhan desain.
- b. Inheritance
 - Kelas dasar Entity diturunkan menjadi Player, Enemy, dan Item.
 - Enemy diturunkan lagi menjadi PatrolEnemy dan (opsional) SmartEnemy.
 - PowerUp diturunkan lagi menjadi SpeedBoost, ImmuneShield, dan lainnya.
- c. Polymorphism
 - Method update() pada power-up memiliki implementasi berbeda pada setiap subclass.
 - Method update_state() pada musuh berperilaku berbeda antara PatrolEnemy dan SmartEnemy.
 - Method move() di-override oleh Player dan Enemy dengan logika masing-masing.

1.6 Keunikan dan Kreativitas Game

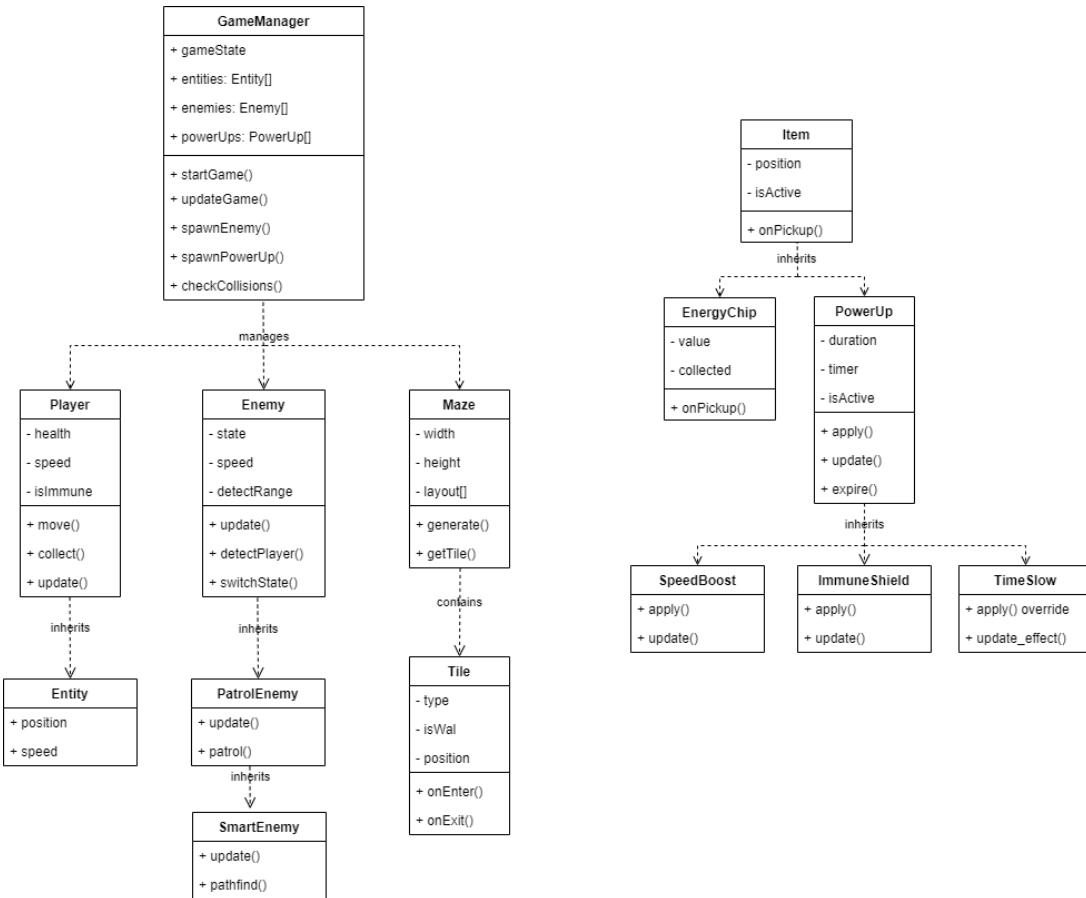
- Memadukan konsep Pacman klasik dengan tema laboratorium futuristik.
- Sistem AI menggunakan pola patrol-chase yang memberikan tantangan strategis.
- Penambahan power-up memberikan variasi gameplay dan meningkatkan replayability.
- Struktur OOP yang modular memudahkan pengembangan fitur tambahan.

BAB II

PERANCANGAN DAN DOKUMENTASI SISTEM

2.1 Perancangan

a. Diagram Class



1. Kelas: GameManager

Peran Utama:

- Mengatur seluruh alur permainan dari awal hingga selesai.
- Mengelola objek penting seperti Player, Enemy, Maze, dan Power-Up.
- Menjalankan update game setiap frame dan menangani spawn musuh/item.
- Mengecek tabrakan antara pemain, musuh, dan item.

Atribut

- **gameState** > status game (play, pause, win, lose).
- **entities: Entity[]** > daftar seluruh objek bergerak dalam game.
- **enemies: Enemy[]** > daftar musuh yang aktif.
- **powerUps: PowerUp[]** > daftar power-up yang muncul atau sedang aktif.

Method

- startGame() > Menginisialisasi player, maze, dan musuh awal.
- updateGame() > Menjalankan update per frame untuk player, musuh, dan item.
- spawnEnemy() > Membuat musuh baru pada lokasi tertentu.
- spawnPowerUp() > Menghasilkan power-up secara acak atau terjadwal.
- checkCollisions() > Mengecek tabrakan player dengan musuh atau item dan mengeksekusi efeknya.

2. Kelas: Player

Peran Utama:

- Mewakili karakter utama yang dikendalikan pemain.
- Dapat bergerak, mengambil item, dan menerima efek power-up.

Atribut

- Health > jumlah nyawa pemain.
- speed > kecepatan gerak yang dapat berubah oleh power-up.
- isImmune > status kebal dari serangan musuh.

Method

- move() > Menggerakkan player sesuai input.
- collect() > Mengambil item dan memicu efeknya.
- update() > Memperbarui status player setiap frame.

3. Kelas: Entity

Peran Utama:

- Menjadi kelas dasar bagi semua objek yang memiliki posisi dan kecepatan.
- Menyatukan properti umum Player dan Enemy

Atribut

- Position > koordinat objek pada map.
- Speed > kecepatan gerakan objek

4. Kelas: Player

Peran Utama:

- Mewakili karakter utama yang dikendalikan pemain.
- Dapat bergerak, mengambil item, dan menerima efek power-up.

Atribut

- health > jumlah nyawa pemain.
- speed > kecepatan gerak yang dapat berubah oleh power-up.
- isImmune > status kebal dari serangan musuh

Method

- move() > Menggerakkan player sesuai input.
- collect() > Mengambil item dan memicu efeknya.
- update() > Memperbarui status player setiap frame.

5. Kelas: Enemy

Peran Utama:

- Menjadi dasar untuk semua musuh dalam permainan.
- Memiliki kemampuan mendeteksi player dan berpindah state.

Atribut

- state > kondisi musuh (patrol/chase).
- speed > kecepatan musuh.
- detectRange > jarak untuk mendeteksi player.

Method

- update() > Mengatur perilaku berdasarkan state.
- detectPlayer() > Mengecek apakah player masuk jangkauan.
- switchState() > Mengubah mode musuh (patrol ↔ chase).

6. Kelas: PatrolEnemy (turunan Enemy)

Peran Utama:

Musuh yang bergerak secara berulang mengikuti jalur patroli.

Atribut

(mewarisi atribut Enemy)

Method

- update() > Menjalankan logika patrol.
- patrol() > Bergerak mengikuti rute patroli.

7. Kelas: SmartEnemy (turunan PatrolEnemy)

Peran Utama:

Musuh yang dapat menghitung jalur (pathfinding) untuk mengejar player.

Method

- update() > Logika enemy cerdas.
- pathfind() > Mencari rute menuju player.

8. Kelas: Maze

Peran Utama:

- Mewakili peta labirin tempat game berlangsung.
- Menyimpan kumpulan Tile (dinding, jalan, tujuan).

Atribut

- Width > lebar maze.
- height > int tinggi maze.
- layout[] > Tile matriks tile.

Method

- generate() > Membentuk labirin otomatis.
- getTile() > Mengambil tile berdasarkan koordinat.

9. Kelas: Tile

Peran Utama:

- Unit terkecil dalam maze seperti dinding atau jalur.

- Mengendalikan perilaku saat player memasuki atau meninggalkan tile.

Atribut

- type > tipe tile (wall/path/goal).
- isWall > cek apakah tile adalah dinding.
- position > lokasi tile.

Method

- onEnter() > Efek saat pemain masuk.
- onExit() > Efek saat keluar.

10. Kelas: Item

Peran Utama:

- Menjadi dasar semua objek yang dapat diambil player.
- Mengatur logika pickup sebelum efek diterapkan.

Atribut

- position
- isActive

Method

onPickup() > Dipanggil ketika item diambil.

11. Kelas: EnergyChip (turunan Item)

Peran Utama:

Item yang menambah skor saat diambil.

Atribut

- value
- collected

Method

onPickup() menambahkan skor player.

12. Kelas: PowerUp (turunan Item)

Peran Utama:

Memberikan efek sementara kepada player (buff).

Atribut

- Duration
- timer
- isActive

Method

- apply() > Menerapkan buff.
- update() > Menghitung durasi buff.
- expire() > Mengakhiri efek buff.

13. Turunan PowerUp

a. SpeedBoost

Peran: Meningkatkan kecepatan pemain.

Metode:

- apply()
- update()

b. ImmuneShield

Peran: Membuat pemain kebal sementara.

Metode:

- apply()

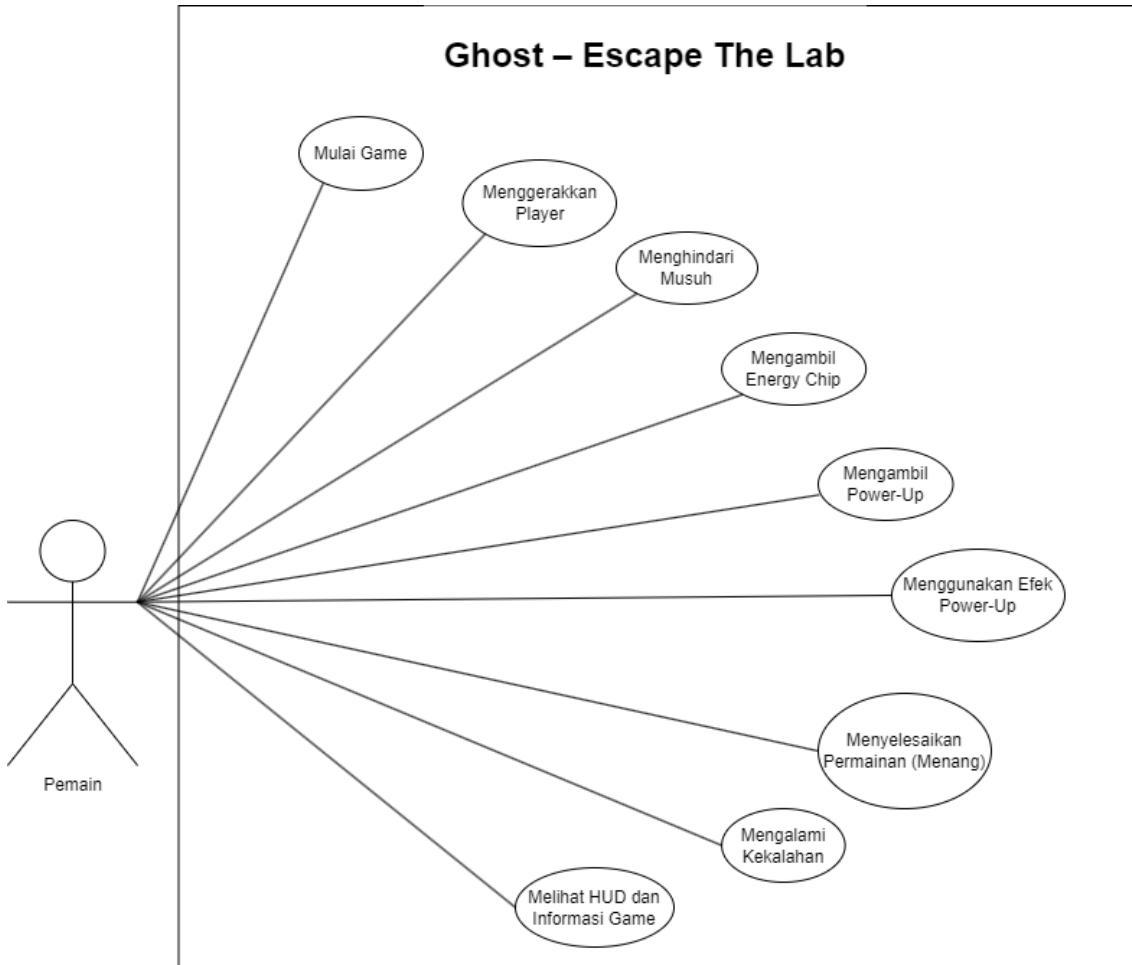
c. TimeSlow

Peran: Memperlambat semua musuh.

Metode:

- apply() override
- update_effect()

b. Alur Aplikasi



- Aktor Utama: Pemain.
- Interaksi Pemain dengan Sistem:
 - Mulai Game: Memulai permainan baru.
 - Menggerakkan Player: Mengontrol pergerakan karakter pemain.
 - Menghindari Musuh: Menghindari atau melarikan diri dari musuh untuk bertahan hidup.
 - Mengambil Energy Chip: Mengumpulkan chip energi untuk menambah sumber daya (seperti stamina atau poin).
 - Mengambil Power-Up: Mengumpulkan item power-up untuk keuntungan sementara.
 - Menggunakan Efek Power-Up: Mengaktifkan efek dari power-up yang telah diambil.
 - Menyelesaikan Permainan (Menang): Mencapai kemenangan dengan menyelesaikan tujuan utama (escape lab).
 - Mengalami Kekalahan: Gagal bertahan, menyebabkan game over.
 - Melihat HUD dan Informasi Game: Memantau tampilan Heads-Up Display (HUD) seperti skor, kesehatan, dan info penting lainnya.

2.2 Implementasi Sistem (Source Code)

a. Entities

1. boss_enemy.py

```
boss_enemy.py X
entities > boss_enemy.py > BossEnemy > update
1 # FILE: entities/boss_enemy.py
2 import pygame
3 import math
4 from entities.smart_enemy import SmartEnemy
5 from settings import *
6
7 # FILE: entities/boss_enemy.py
8 import pygame
9 import math
10 from entities.enemy import Enemy
11 from settings import *
12 import os
13
14 class BossEnemy(Enemy):
15     def __init__(self, x, y, color_override=None, image_path="boss.png"):
16         # Gunakan image_path yang dikirim dari shop, fallback ke boss.png
17         super().__init__(x, y, MAGENTA, image_path)
18
19         # Resize boss agar lebih besar
20         target_size = 60
21         self.frames = [
22             pygame.transform.scale(f, (target_size, target_size))
23             for f in self.frames
24         ]
25         self.image = self.frames[0]
26
27         # Posisikan di tengah tile
28         center_x = x + TILE_SIZE // 2
29         center_y = y + TILE_SIZE // 2
30         self.rect = self.image.get_rect(center=(center_x, center_y))
31
32         self.speed = BOSS_SPEED
33         self.name = "BOSS"
34
35         self.zigzag_timer = 0
36
37     def update(self, walls, player):
38         # Boss selalu mengejar player
39         dx = player.rect.centerx - self.rect.centerx
40         dy = player.rect.centery - self.rect.centery
41         length = math.hypot(dx, dy)
42
43         if length == 0:
44             return
45
46         dir_x = dx / length
47         dir_y = dy / length
```

```

49         # Zig-zag movement
50         self.zigzag_timer += 0.2
51         offset = math.sin(self.zigzag_timer) * 0.6
52
53         self.rect.x += (dir_x + offset) * self.speed
54         self.check_collision(walls, 'x')
55
56         self.rect.y += (dir_y - offset) * self.speed
57         self.check_collision(walls, 'y')
58

```

2. enemy.py

```

enemy.py  X
entities > enemy.py > ...
1 import pygame
2 from entities.entity import Entity
3 from settings import *
4
5 class Enemy(Entity):
6     # KITA TAMBAHKAN image_path=None DISINI
7     def __init__(self, x, y, color=RED, image_path=None):
8         # LALU TERUSKAN image_path KE INDUKNYA (Entity)
9         super().__init__(x, y, color, "Enemy", image_path)
10        self.speed = ENEMY_SPEED
11
12    def update(self, walls, player):
13        # Base method, to be overridden (Polymorphism)
14        pass
15
16    def apply_skin(self, image_filename):
17        import os
18        path = os.path.join("assets", "images", image_filename)
19
20        if os.path.exists(path):
21            img = pygame.image.load(path).convert_alpha()
22            img = pygame.transform.scale(img, (TILE_SIZE-4, TILE_SIZE-4))
23            self.frames = [img]
24            self.image = img
25

```

3. entity.py

```
entity.py  X
entities > entity.py > ...
1  # FILE: entities/entity.py
2  import pygame
3  import os
4  from settings import *
5
6  class Entity(pygame.sprite.Sprite):
7      def __init__(self, x, y, color, name="Entity", image_path=None):
8          super().__init__()
9
10         # --- ANIMATION SYSTEM ---
11         self.frames = []
12         self.frame_index = 0
13         self.animation_speed = 0.1
14         self.last_update = 0
15
16         self.image = None
17
18         # --- DEBUGGING IMAGE LOAD ---
19         if image_path:
20             # 1. Tentukan path lengkap
21             full_path = os.path.join("assets", "images", image_path)
22
23             # 2. Cek apakah file ada secara fisik
24             if os.path.exists(full_path):
25                 try:
26                     # Coba load
27                     loaded = pygame.image.load(full_path).convert_alpha()
28                     self.image = pygame.transform.scale(loaded, (TILE_SIZE-4, TILE_SIZE-4))
29                     self.frames.append(self.image)
30                     # Jika berhasil, print info sukses (Opsional, bisa dihapus nanti)
31                     print(f"[SUCCESS] Loaded: {full_path}")
32                 except Exception as e:
33                     # Jika file ada tapi error saat load (misal corrupt)
34                     print(f"[ERROR] Gagal load gambar {full_path}: {e}")
35                 else:
36                     # Jika file tidak ditemukan
37                     # PENTING: Cek path ini di terminalmu nanti!
38                     print(f"[MISSING] File tidak ditemukan di: {os.path.abspath(full_path)}")
39
40             # --- FALBACK (JIKA GAMBAR GAGAL) ---
41             if not self.frames:
42                 if image_path:
43                     print(f"[FALLBACK] Menggunakan kotak warna untuk {name} karena gambar gagal.")
44
```

```

45     # Buat 2 frame warna (kedip dikit) untuk animasi dummy
46     surf1 = pygame.Surface((TILE_SIZE - 4, TILE_SIZE - 4))
47     surf1.fill(color)
48     surf2 = pygame.Surface((TILE_SIZE - 4, TILE_SIZE - 4))
49     c2 = (min(255, color[0]+30), min(255, color[1]+30), min(255, color[2]+30))
50     surf2.fill(c2)
51     self.frames = [surf1, surf2]
52
53     self.image = self.frames[0]
54     self.rect = self.image.get_rect(topleft=(x + 2, y + 2))
55
56     self.direction = pygame.math.Vector2(0, 0)
57     self.speed = 0
58     self.name = name
59
60     def animate(self):
61         self.frame_index += self.animation_speed
62         if self.frame_index >= len(self.frames):
63             self.frame_index = 0
64         self.image = self.frames[int(self.frame_index)]
65
66     def move(self, walls):
67         self.rect.x += self.direction.x * self.speed
68         self.check_collision(walls, 'x')
69         self.rect.y += self.direction.y * self.speed
70         self.check_collision(walls, 'y')
71
72         if self.direction.magnitude() != 0:
73             self.animate()
74
75     def check_collision(self, walls, axis):
76         for wall in walls:
77             if self.rect.colliderect(wall):
78                 if axis == 'x':
79                     if self.direction.x > 0: self.rect.right = wall.left
80                     elif self.direction.x < 0: self.rect.left = wall.right
81                 elif axis == 'y':
82                     if self.direction.y > 0: self.rect.bottom = wall.top
83                     elif self.direction.y < 0: self.rect.top = wall.bottom
84
85     def draw(self, surface):
86         surface.blit(self.image, self.rect)

```

4. patrol_enemy.py

```
patrol_enemy.py X
entities > patrol_enemy.py > PatrolEnemy
1 import pygame
2 import random
3 from entities.enemy import Enemy
4 from settings import *
5
6 class PatrolEnemy(Enemy):
7     def __init__(self, x, y, image_path="enemy_patrol.png"):
8         super().__init__(x, y, RED, image_path=image_path)
9         self.direction = pygame.math.Vector2(1, 0)
10        self.move_timer = 0
11
12    def update(self, walls, player):
13        # Simple Logic: Move until hit wall, then random direction
14        old_x, old_y = self.rect.x, self.rect.y
15        self.move(walls)
16
17        # If position didn't change, we hit a wall
18        if self.rect.x == old_x and self.rect.y == old_y:
19            self.change_direction()
20
21        # Optional: Randomly change direction occasionally
22        if pygame.time.get_ticks() - self.move_timer > 2000:
23            self.change_direction()
24            self.move_timer = pygame.time.get_ticks()
25
26    def change_direction(self):
27        directions = [pygame.math.Vector2(1, 0), pygame.math.Vector2(-1, 0),
28                      pygame.math.Vector2(0, 1), pygame.math.Vector2(0, -1)]
29        self.direction = random.choice(directions)
```

5. player.py

```
player.py X
entities > player.py > ...
1 import pygame
2 from entities.entity import Entity
3 from settings import *
4
5 class Player(Entity):
6     def __init__(self, x, y):
7         # 1. Panggil Parent (Ini akan memuat gambar ke dalam self.frames)
8         super().__init__(x, y, BLUE, "Player", image_path="player.png")
9
10        # --- PERBAIKAN LOGIKA UKURAN (AGAR TETAP KECIL SAAT ANIMASI) ---
11        target_size = 28
12
13        # Masalah sebelumnya: Kita hanya resize self.image, tapi self.frames masih besar.
14        # Solusi: Kita resize SEMUA gambar di dalam self.frames.
15        new_frames = []
16        for frame in self.frames:
17            resized_frame = pygame.transform.scale(frame, (target_size, target_size))
18            new_frames.append(resized_frame)
19
20        # Timpa frames lama dengan frames yang sudah kecil
21        self.frames = new_frames
22
23        # Set image awal ke frame pertama yang sudah kecil
24        self.image = self.frames[0]
25
```

```

26     # 2. Update Rect (Hitbox) agar di Tengah Tile
27     center_x = x + TILE_SIZE // 2
28     center_y = y + TILE_SIZE // 2
29     self.rect = self.image.get_rect(center=(center_x, center_y))
30     # -----
31
32     self.base_speed = PLAYER_SPEED
33     self.speed = self.base_speed
34
35     # Private variables
36     self._is_immune = False
37     self._immune_timer = 0
38     self._speed_timer = 0
39     self._immune_duration = 0
40     self.original_color = BLUE
41
42     def apply_skin(self, image_name):
43         import os
44
45         if not image_name:
46             print("[SKIN] Tidak ada nama file skin.")
47             return
48
49         path = os.path.join("assets", "images", image_name)
50
51         if not os.path.exists(path):
52             print(f"[SKIN ERROR] File tidak ditemukan: {path}")
53             return
54
55         try:
56             img = pygame.image.load(path).convert_alpha()
57
58             # Resize mengikuti ukuran player saat ini
59             img = pygame.transform.scale(img, (self.rect.width, self.rect.height))
60
61             # Replace frame & image
62             self.frames = [img]
63             self.image = img
64
65             # Update rect center biar tidak geser
66             cx, cy = self.rect.center
67             self.rect = self.image.get_rect(center=(cx, cy))
68
69             print(f"[SKIN] Skin applied: {image_name}")
70
71         except Exception as e:
72             print(f"[SKIN ERROR] {e}")
73

```

```

74     def input(self):
75         keys = pygame.key.get_pressed()
76         self.direction.x = 0
77         self.direction.y = 0
78
79         # Movement Logic
80         if keys[pygame.K_w]:
81             self.direction.y = -1
82         elif keys[pygame.K_s]:
83             self.direction.y = 1
84         elif keys[pygame.K_a]:
85             self.direction.x = -1
86         elif keys[pygame.K_d]:
87             self.direction.x = 1
88
89         # Normalisasi vector
90         if self.direction.magnitude() != 0:
91             self.direction = self.direction.normalize()
92
93     def update(self, walls):
94         self.input()
95         # move() ada di parent class Entity, dia otomatis panggil animate()
96         self.move(walls)
97         self.handle_powerup_timers()
98
99     def apply_speed_boost(self, duration_sec):
100        self.speed = self.base_speed * 1.5
101        self._speed_timer = pygame.time.get_ticks() + (duration_sec * 1000)
102        # Visual feedback
103        if not self._is_immune:
104            self.image.set_alpha(150)
105
106    def apply_immune_shield(self, duration_sec):
107        self._is_immune = True
108        self._immune_duration = duration_sec * 1000
109        self._immune_timer = pygame.time.get_ticks() + self._immune_duration
110
111    def handle_powerup_timers(self):
112        current_time = pygame.time.get_ticks()
113
114        # Check Speed Boost
115        if self.speed != self.base_speed and current_time > self._speed_timer:
116            self.speed = self.base_speed
117            self.image.set_alpha(255) # Kembali normal
118
119        # Check Immune Shield
120        if self._is_immune and current_time > self._immune_timer:
121            self._is_immune = False
122
123    def is_invincible(self):
124        return self._is_immune
125
126    def draw(self, surface):
127        super().draw(surface)
128        if self._is_immune:
129            # Gambar lingkaran pelindung
130            pygame.draw.circle(surface, ORANGE, self.rect.center, 20, 2)

```

6. smart_enemy.py

```
# FILE: entities/smart_enemy.py
import pygame
import math
import random
from entities.enemy import Enemy
from settings import *

class SmartEnemy(Enemy):
    def __init__(self, x, y, image_path="enemy_smart.png"):
        super().__init__(x, y, PURPLE, image_path=image_path)

    # Resize
    target = 28
    self.frames = [pygame.transform.scale(f, (target, target)) for f in self.frames]
    self.image = self.frames[0]

    center_x = x + TILE_SIZE//2
    center_y = y + TILE_SIZE//2
    self.rect = self.image.get_rect(center=(center_x, center_y))

    self.original_frames = [f.copy() for f in self.frames]
    self.state = "PATROL"
    self.speed = ENEMY_SPEED

    self.patrol_dir = pygame.math.Vector2(0, 1)
    self.patrol_timer = 0

    # Variabel Warna Original untuk reset
    self.original_frames = [f.copy() for f in self.frames]

def update(self, walls, player):
    dist_to_player = math.hypot(player.rect.centerx - self.rect.centerx,
                                player.rect.centery - self.rect.centery)

    if dist_to_player < DETECT_RADIUS:
        if self.state != "CHASE":
            self.state = "CHASE"
            # EFFECT: Berubah Merah saat Chase (Behavior Upgrade)
            for f in self.frames:
                f.fill((255, 50, 50), special_flags=pygame.BLEND_MULT)
        else:
            if self.state == "CHASE":
                self.state = "PATROL"
                # Reset Warna
                self.frames = [f.copy() for f in self.original_frames]

    if self.state == "CHASE":
        self.speed = CHASE_SPEED
        self.chase(player, walls)
    else:
        self.speed = ENEMY_SPEED
        self.patrol(walls)

    # Animasi kedip saat chase
    if self.state == "CHASE":
        self.animate()
```

```

58     def chase(self, player, walls):
59         dx = player.rect.centerx - self.rect.centerx
60         dy = player.rect.centery - self.rect.centery
61         length = math.hypot(dx, dy)
62         if length == 0: return
63
64         dir_x = dx / length
65         dir_y = dy / length
66
67         self.rect.x += dir_x * self.speed
68         self.check_collision(walls, 'x')
69         self.rect.y += dir_y * self.speed
70         self.check_collision(walls, 'y')
71
72     def patrol(self, walls):
73         self.rect.x += self.patrol_dir.x * self.speed
74         self.check_collision(walls, 'x')
75         self.rect.y += self.patrol_dir.y * self.speed
76         self.check_collision(walls, 'y')
77
78         if pygame.time.get_ticks() - self.patrol_timer > 1000:
79             if random.randint(0, 50) == 0:
80                 self.change_dir()
81
82     def check_collision(self, walls, axis):
83         for wall in walls:
84             if self.rect.colliderect(wall):
85                 if axis == 'x':
86                     if self.rect.centerx < wall.centerx: self.rect.right = wall.left
87                     else: self.rect.left = wall.right
88                     if self.state == "PATROL": self.change_dir()
89                 elif axis == 'y':
90                     if self.rect.centery < wall.centery: self.rect.bottom = wall.top
91                     else: self.rect.top = wall.bottom
92                     if self.state == "PATROL": self.change_dir()
93
94     def change_dir(self):
95         dirs = [pygame.math.Vector2(1,0), pygame.math.Vector2(-1,0),
96                 pygame.math.Vector2(0,1), pygame.math.Vector2(0,-1)]
97         self.patrol_dir = random.choice(dirs)
98         self.patrol_timer = pygame.time.get_ticks()

```

b. Items

1. energy_chip.py

```
�能 energy_chip.py X  
items > 能 energy_chip.py > ...  
1  from items.item import Item  
2  from settings import *  
3  
4  class EnergyChip(Item):  
5      def __init__(self, x, y):  
6          # Memanggil gambar "chip.png"  
7          super().__init__(x, y, GREEN, image_path="chip.png")  
8  
9      def apply(self, player, game_manager):  
10         game_manager.add_score(1)
```

2. immune_shield.py

```
能 immune_shield.py X  
items > 能 immune_shield.py > ...  
1  from items.powerup import PowerUp  
2  from settings import *  
3  
4  class ImmuneShield(PowerUp):  
5      def __init__(self, x, y):  
6          # Memanggil gambar "powerup_shield.png"  
7          super().__init__(x, y, ORANGE, image_path="powerup_shield.png")  
8  
9      def apply(self, player, game_manager):  
10         game_manager.active_message = "SHIELD ACTIVE!"  
11         game_manager.message_timer = pygame.time.get_ticks() + 2000  
12         player.apply_immune_shield(5)
```

3. item.py

```
item.py X
items > item.py > Item > __init__
1 # FILE: items/item.py
2 import pygame
3 import os
4 from settings import *
5
6 class Item(pygame.sprite.Sprite):
7     def __init__(self, x, y, color, image_path=None):
8         super().__init__()
9         self.image = None
10
11     # --- LOGIKA LOAD GAMBAR ---
12     if image_path:
13         full_path = os.path.join("assets", "images", image_path)
14         if os.path.exists(full_path):
15             try:
16                 loaded = pygame.image.load(full_path).convert_alpha()
17                 # Resize item agar pas (misal 24x24 pixel)
18                 self.image = pygame.transform.scale(loaded, (36, 36))
19             except:
20                 pass
21
22     # Fallback jika gambar tidak ada -> Pakai Kotak Warna
23     if self.image is None:
24         self.image = pygame.Surface((TILE_SIZE//2, TILE_SIZE//2))
25         self.image.fill(color)
26         self.rect = self.image.get_rect(center=(x + TILE_SIZE//2, y + TILE_SIZE//2))
27
28     def apply(self, player, game_manager):
29         pass
```

4. powerup.py

```
powerup.py X
items C:\Users\Mentari\Documents\PBO\UAS\items\powerup.py (preview ⚡)
1 # FILE: items/powerup.py
2 from items.item import Item
3
4 class PowerUp(Item):
5     # Tambahkan parameter image_path=None
6     def __init__(self, x, y, color, image_path=None):
7         # Teruskan image_path ke induk (Item)
8         super().__init__(x, y, color, image_path)
9
10    def apply(self, player, game_manager):
11        pass # Overridden by children
```

5. speed_boost.py

```
speed_boost.py X
items > speed_boost.py > ...
1  from items.powerup import PowerUp
2  from settings import *
3
4  class SpeedBoost(PowerUp):
5      def __init__(self, x, y):
6          # Memanggil gambar "powerup_speed.png"
7          super().__init__(x, y, CYAN, image_path="powerup_speed.png")
8
9      def apply(self, player, game_manager):
10         game_manager.active_message = "SPEED BOOST!"
11         game_manager.message_timer = pygame.time.get_ticks() + 4000
12         player.apply_speed_boost(3)
```

c. leaderboard.py

```
leaderboard_manager.py X
leaderboard_manager.py > ...
1  # FILE: leaderboard_manager.py
2  import json
3  import os
4
5  LEADERBOARD_FILE = "leaderboard.json"
6
7  class LeaderboardManager:
8      def __init__(self):
9          self.scores = []
10         self.load_scores()
11     def load_scores(self):
12         if os.path.exists(LEADERBOARD_FILE):
13             try:
14                 with open(LEADERBOARD_FILE, "r") as f:
15                     self.scores = json.load(f)
16             except:
17                 self.scores = []
18             else:
19                 self.scores = []
20     def save_score(self, name, score):
21         # Tambah skor baru
22         self.scores.append({"name": name, "score": score})
23         # Urutkan dari tertinggi ke terendah
24         self.scores.sort(key=lambda x: x["score"], reverse=True)
25         # Ambil Top 5
26         self.scores = self.scores[:5]
27
28         with open(LEADERBOARD_FILE, "w") as f:
29             json.dump(self.scores, f, indent=4)
```

d. main.py

```
main.py  X  
main.py > ...  
1  # FILE: main.py  
2  import pygame  
3  import sys  
4  import random  
5  import os  
6  from settings import *  
7  from save_manager import SaveManager  
8  from leaderboard_manager import LeaderboardManager  
9  from vfx import ParticleManager  
10 from maze import Maze, LEVELS  
11 from entities.player import Player  
12 from entities.patrol_enemy import PatrolEnemy  
13 from entities.smart_enemy import SmartEnemy  
14 from entities.boss_enemy import BossEnemy  
15 from items.energy_chip import EnergyChip  
16 from items.speed_boost import SpeedBoost  
17 from items.immune_shield import ImmuneShield  
18
```

```
42 class GameManager:  
43     def __init__(self):  
44         pygame.init()  
45         pygame.mixer.init()  
46  
47         self.screen = pygame.display.set_mode((SCREEN_WIDTH, SCREEN_HEIGHT))  
48         pygame.display.set_caption("Subject: Eliminated")  
49         self.clock = pygame.time.Clock()  
50  
51     # Managers  
52     self.save_manager = SaveManager()  
53     self.leaderboard = LeaderboardManager()  
54     self.particles = ParticleManager()  
55  
56     # Fonts  
57     self.font = pygame.font.SysFont("Arial", 20)  
58     self.book_font = pygame.font.SysFont("Courier New", 24, bold=True)  
59     self.title_font = pygame.font.SysFont("Arial", 64, bold=True)  
60     self.info_font = pygame.font.SysFont("Arial", 32)  
61  
62     # Game State  
63     self.game_state = "MENU"  
64     self.current_level_index = 0  
65     self.score = 0  
66     self.target_score = 0  
67     self.active_message = ""  
68     self.message_timer = 0  
69
```

```

70     self.shop_category = "player"
71     self.current_book_id = None
72     self.current_page = 0
73
74     # Transition
75     self.transition_alpha = 0
76     self.transition_mode = None
77     self.transition_surface = pygame.Surface((SCREEN_WIDTH, SCREEN_HEIGHT))
78     self.transition_surface.fill(BLACK)
79
80     # --- LOAD ASSETS (SOUNDS & BACKGROUND) ---
81     self.sounds = {}
82     self.load_sounds()
83     self.background = None
84     self.load_background()
85
86     self.all_sprites = pygame.sprite.Group()
87     self.enemies = pygame.sprite.Group()
88     self.items = pygame.sprite.Group()
89

```

```

135     def setup_level(self, level_index):
136         self.all_sprites.empty()
137         self.enemies.empty()
138         self.items.empty()
139         self.score = 0
140
141         level_data = LEVELS[level_index]
142
143         tile_skin_id = self.save_manager.data['equipped']['tile']
144         tile_color = self.get_color_from_id(tile_skin_id, BLUE)
145         tile_texture = self.get_texture_from_id(tile_skin_id, "wall_default.png")
146         self.maze = Maze(level_data, wall_texture=tile_texture, wall_color=tile_color)
147
148         player_skin_id = self.save_manager.data['equipped']['player']
149         skin_file = self.get_texture_from_id(player_skin_id)
150
151         player_spawned = False
152
153         for r, row in enumerate(level_data):
154             for c, tile in enumerate(row):
155                 if tile == PLAYER_START:
156                     self.player = Player(c*TILE_SIZE, r*TILE_SIZE)
157                     self.all_sprites.add(self.player)
158                     player_spawned = True
159

```

```

160             # APPLY SKIN SETELAH PLAYER ADA
161             if skin_file:
162                 self.player.apply_skin(skin_file)
163
164             shield_x = (c + 1) * TILE_SIZE
165             shield_y = r * TILE_SIZE
166             start_shield = ImmuneShield(shield_x, shield_y)
167             self.all_sprites.add(start_shield)
168             self.items.add(start_shield)
169
170         if not player_spawned:
171             self.player = Player(40, 40)
172             self.all_sprites.add(self.player)
173
174         is_boss_level = (level_index == 9)
175         num_chips = 3 + level_index
176         num_enemies = 2 + int(level_index * 1.5)
177         num_powerups = 3
178
179         self.target_score = num_chips
180         if not self.maze.path_tiles: return
181
182         self.spawn_random_object(EnergyChip, num_chips)
183

```

```

184     if is_boss_level:
185         boss_skin_id = self.save_manager.data['equipped']['boss']
186         boss_skin_file = self.get_texture_from_id(boss_skin_id, "boss.png")
187
188         self.spawn_boss_enemy(BossEnemy, boss_skin_file)
189
190         self.spawn_random_enemies(SmartEnemy, 2, PURPLE)
191     else:
192         smart_skin = self.save_manager.data['equipped']['enemy']
193         patrol_skin = self.save_manager.data['equipped']['patrol']
194
195
196         smart_count = level_index // 3
197         patrol_count = num_enemies - smart_count
198         self.spawn_random_enemies(PatrolEnemy, patrol_count, patrol_skin)
199         self.spawn_random_enemies(SmartEnemy, smart_count, smart_skin)
200
201         self.spawn_random_object(SpeedBoost, num_powerups // 2)
202         self.spawn_random_object(ImmuneShield, num_powerups - (num_powerups // 2))
203

```

```
204     def spawn_boss_enemy(self, obj_class, image_file):
205         if not self.maze.path_tiles:
206             return
207         spawn_pos = random.choice(self.maze.path_tiles)
208         boss = obj_class(spawn_pos[0], spawn_pos[1], image_path=image_file)
209         self.all_sprites.add(boss)
210         self.enemies.add(boss)
211
212     def spawn_random_object(self, obj_class, count):
213         for _ in range(count):
214             if not self.maze.path_tiles: break
215             spawn_pos = random.choice(self.maze.path_tiles)
216             new_obj = obj_class(spawn_pos[0], spawn_pos[1])
217             self.all_sprites.add(new_obj)
218             self.items.add(new_obj)
219
220     def spawn_random_enemies(self, obj_class, count, skin_id):
221         for _ in range(count):
222             if not self.maze.path_tiles:
223                 break
224
225             spawn_pos = random.choice([self.maze.path_tiles])
226             # DUIT ITEM ↓
227             skin_file = self.get_texture_from_id(skin_id, None)
228             enemy = obj_class(spawn_pos[0], spawn_pos[1], image_path=skin_file)
229
230             self.enemies.add(enemy)
231             self.all_sprites.add(enemy)
```

```
233     def run(self):
234         while True:
235             self.events()
236             self.update()
237             self.draw()
238             self.clock.tick(FPS)
239
```

```

378     if self.game_state == "PLAYING" and not self.transition_mode:
379         self.player.update(self.maze.walls)
380
381         for enemy in self.enemies:
382             enemy.update(self.maze.walls, self.player)
383
384         hit_list = pygame.sprite.spritecollide(self.player, self.items, True)
385         for item in hit_list:
386             item.apply(self.player, self)
387             color = GREEN
388             if isinstance(item, SpeedBoost): color = CYAN
389             elif isinstance(item, ImmuneShield): color = ORANGE
390             self.particles.emit(item.rect.centerx, item.rect.centery, color)
391
392             if isinstance(item, EnergyChip):
393                 self.play_sound('collect')
394                 self.save_manager.add_coins(10)
395             else:
396                 self.play_sound('powerup')
397
398             if not self.player.is_invincible():
399                 if pygame.sprite.spritecollide(self.player, self.enemies, False):
400                     self.game_state = "LOSE"
401                     self.play_sound('lose')
402                     self.leaderboard.save_score("Player", self.score * 10)
403

```

```

404         if pygame.time.get_ticks() > self.message_timer:
405             self.active_message = ""
406
407     def add_score(self, amount):
408         self.score += amount
409         if self.score >= self.target_score:
410             if self.current_level_index < len(LEVELS) - 1:
411                 self.play_sound('win')
412                 next_level = self.current_level_index + 1
413                 self.start_transition("OUT", lambda: self.start_game(next_level))
414             else:
415                 self.game_state = "WIN"
416                 self.play_sound('win')
417                 self.leaderboard.save_score("Player", self.score * 100)
418

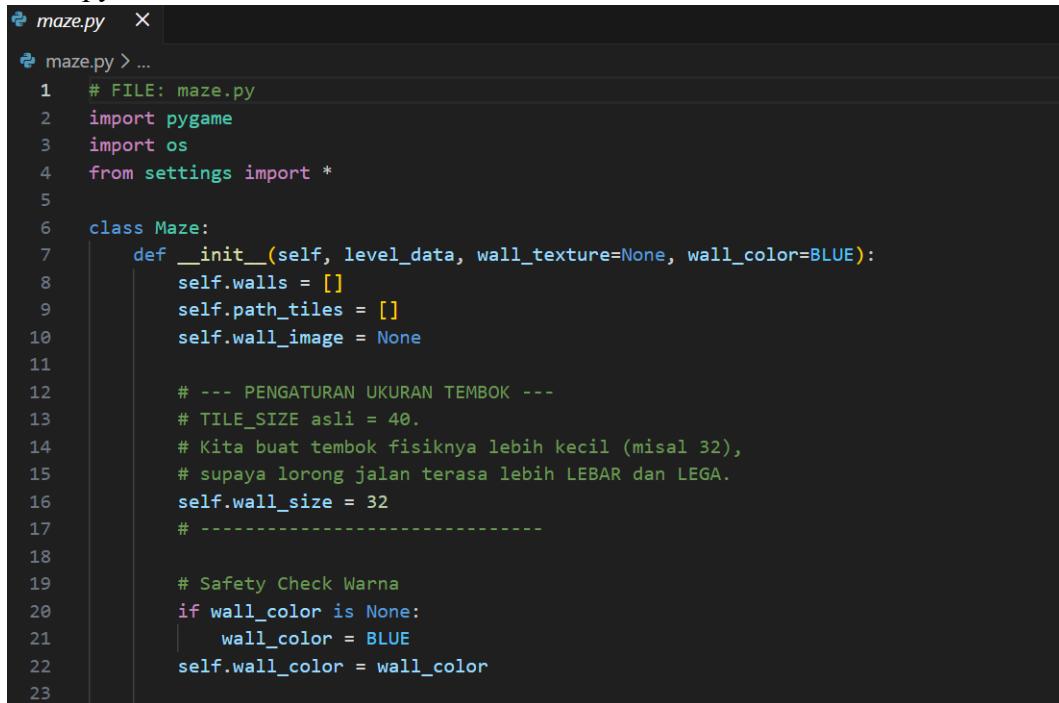
```

```

463     def draw_hud(self):
464         hud_y = 0
465         hud_bg = pygame.Surface((SCREEN_WIDTH, 30))
466         hud_bg.set_alpha(200)
467         hud_bg.fill((20, 20, 30))
468         self.screen.blit(hud_bg, (0, hud_y))
469
470         pygame.draw.line(self.screen, GRAY, (0, hud_y), (SCREEN_WIDTH, hud_y), 2)
471
472         info_left = f"LVL: {self.current_level_index + 1} | CHIPS: {self.score}/{self.target_score}"
473         txt_left = self.font.render(info_left, True, WHITE)
474         self.screen.blit(txt_left, (20, hud_y + 15))
475
476         info_coin = f"COINS: {self.save_manager.data['coins']}"
477         txt_coin = self.font.render(info_coin, True, GOLD)
478         self.screen.blit(txt_coin, (SCREEN_WIDTH - 350, hud_y + 15))
479
480         x_offset = SCREEN_WIDTH - 150
481         current_time = pygame.time.get_ticks()
482
483         if self.player.is_invincible():
484             time_left = max(0, (self.player._immune_timer - current_time) // 100)
485             pygame.draw.circle(self.screen, ORANGE, (x_offset, hud_y + 25), 15)
486             t_txt = self.font.render(str(time_left), True, WHITE)
487             self.screen.blit(t_txt, (x_offset-5, hud_y + 15))
488             x_offset += 50
489
490
491         if self.player.speed > PLAYER_SPEED:
492             time_left = max(0, (self.player._speed_timer - current_time) // 100)
493             pygame.draw.circle(self.screen, CYAN, (x_offset, hud_y + 25), 15)
494             t_txt = self.font.render(str(time_left), True, WHITE)
495             self.screen.blit(t_txt, (x_offset-5, hud_y + 15))

```

e. maze.py



```

1  # FILE: maze.py
2  import pygame
3  import os
4  from settings import *
5
6  class Maze:
7      def __init__(self, level_data, wall_texture=None, wall_color=BLUE):
8          self.walls = []
9          self.path_tiles = []
10         self.wall_image = None
11
12         # --- PENGATURAN UKURAN TEMBOK ---
13         # TILE_SIZE asli = 40.
14         # Kita buat tembok fisiknya lebih kecil (misal 32),
15         # supaya lorong jalan terasa lebih LEBAR dan LEGA.
16         self.wall_size = 32
17         # -----
18
19         # Safety Check Warna
20         if wall_color is None:
21             wall_color = BLUE
22         self.wall_color = wall_color

```

```

24     # --- LOGIKA LOAD TEXTURE ---
25     if wall_texture:
26         path = os.path.join("assets", "images", wall_texture)
27         if os.path.exists(path):
28             try:
29                 img = pygame.image.load(path).convert_alpha()
30                 # Resize gambar sesuai ukuran tembok yang sudah dikecilkan
31                 self.wall_image = pygame.transform.scale(img, (self.wall_size, self.wall_size))
32             except:
33                 self.wall_image = None
34
35     # Shadow color
36     self.shadow_color = (max(0, self.wall_color[0]-40),
37                          max(0, self.wall_color[1]-40),
38                          max(0, self.wall_color[2]-40))
39
40     self.create_walls(level_data)
41
42 def create_walls(self, level_data):
43     self.walls = []
44     self.path_tiles = []
45
46     # Hitung offset (geser ke tengah) supaya tembok ada di tengah-tengah kotak grid
47     offset = (TILE_SIZE - self.wall_size) // 2
48
49     for row_index, row in enumerate(level_data):
50         for col_index, tile in enumerate(row):
51             # Koordinat Grid Asli
52             grid_x = col_index * TILE_SIZE
53             grid_y = row_index * TILE_SIZE
54
55             if tile == WALL:
56                 # Buat tembok lebih kecil dan posisikan di tengah grid
57                 # Ini fisik collision-nya yang mengelil
58                 self.walls.append(pygame.Rect(grid_x + offset, grid_y + offset, self.wall_size, self.wall_size))
59             else:
60                 self.path_tiles.append((grid_x, grid_y))
61
62     def draw(self, surface):
63         for wall in self.walls:
64             if self.wall_image:
65                 # Gambar Texture pas di kotak wall
66                 surface.blit(self.wall_image, wall)
67             else:
68                 # Fallback Warna
69                 pygame.draw.rect(surface, self.wall_color, wall)
70                 # Shadow kecil
71                 pygame.draw.rect(surface, self.shadow_color,
72                                  (wall.x, wall.bottom - 4, wall.width, 4))
73                 pygame.draw.rect(surface, self.shadow_color,
74                                  (wall.right - 4, wall.y, 4, wall.height))

```

f. save_manager.py

```
save_manager.py X
save_manager.py > ...
1 # FILE: save_manager.py
2 import json
3 import os
4
5 SAVE_FILE = "save_data.json"
6
7 class SaveManager:
8     def __init__(self):
9         # Default data jika file tidak ada
10        self.data = {
11            "coins": 0,
12            "inventory": [], # List ID item yang sudah dibeli
13            "equipped": {    # Item yang sedang dipakai
14                "player": "default",
15                "enemy": "default",
16                "patrol": "default",
17                "tile": "default",
18                "boss": "default_boss",
19                "story": None
20            },
21            "unlocked_levels": 1
22        }
23        self.load()
24
25    def load(self):
26        if os.path.exists(SAVE_FILE):
27            try:
28                with open(SAVE_FILE, "r") as f:
29                    self.data = json.load(f)
30            except:
31                print("Save file corrupted, creating new one.")
32                self.save()
33        else:
34            self.save()
35
36    def save(self):
37        with open(SAVE_FILE, "w") as f:
38            json.dump(self.data, f, indent=4)
39
40    def reset_data(self):
41        # Hapus save (Fitur request no 4)
42        if os.path.exists(SAVE_FILE):
43            os.remove(SAVE_FILE)
44        self.__init__() # Re-init default data
45        print("Data Reset Successful")
46
47    def add_coins(self, amount):
48        self.data["coins"] += amount
49        self.save()
50
```

```

51     def buy_item(self, item_id, price):
52         if self.data["coins"] >= price and item_id not in self.data["inventory"]:
53             self.data["coins"] -= price
54             self.data["inventory"].append(item_id)
55             self.save()
56             return True
57         return False
58
59     def equip_item(self, category, item_id):
60         # Pastikan item sudah dibeli atau itu default
61         if item_id == "default" or item_id in self.data["inventory"]:
62             self.data["equipped"][category] = item_id
63             self.save()

```

g. settings.py

```

settings.py X
settings.py > ...
1  # FILE: settings.py
2  import pygame
3
4  # Screen Settings
5  SCREEN_WIDTH = 1280
6  SCREEN_HEIGHT = 720
7  TILE_SIZE = 40
8  FPS = 60
9
10 # Colors (Basic)
11 WHITE = (255, 255, 255)
12 BLACK = (0, 0, 0)
13 GRAY = (100, 100, 100)
14 DARK_GRAY = (40, 40, 40)
15 RED = (200, 50, 50)
16 GREEN = (50, 200, 50)
17 BLUE = (50, 50, 200)
18 YELLOW = (255, 215, 0)
19 CYAN = (0, 255, 255)
20 PURPLE = (150, 0, 150)
21 ORANGE = (255, 165, 0)
22 GOLD = (218, 165, 32)
23 MAGENTA = (255, 0, 255)
24
25 # --- SHOP DATA (DATABASE BARANG) ---
26
27 SHOP_ITEMS = [
28     # --- DEFAULT SKINS (ID UNIK) ---
29     {"id": "default_player", "cat": "player", "name": "Default Suit", "price": 0, "color": BLUE, "image": "player.png"}, 
30     {"id": "default_smart", "cat": "enemy", "name": "Default Smart", "price": 0, "color": PURPLE, "image": "enemy_smart.png"}, 
31     {"id": "default_patrol", "cat": "patrol", "name": "Default Patrol", "price": 0, "color": RED, "image": "enemy_patrol.png"}, 
32     {"id": "default_tile", "cat": "tile", "name": "Standard Lab", "price": 0, "color": (100, 100, 200), "image": "wall_default.png"}, 
33
34     # Player skins
35     {"id": "p_neon", "cat": "player", "name": "Neon Armor", "price": 50, "color": CYAN, "image": "player_neon.png"}, 
36
37     # Smart enemy skins
38     {"id": "e_dark", "cat": "enemy", "name": "Dark Hunter", "price": 50, "color": BLUE, "image": "enemy_dark.png"}, 
39
40     # Patrol skins
41     {"id": "pt_police", "cat": "patrol", "name": "Police Bot", "price": 50, "color": BLUE, "image": "enemy_police.png"}, 
42
43     # Boss Skins
44     {"id": "default_boss", "cat": "boss", "name": "Default Boss", "price": 0, "color": MAGENTA, "image": "boss.png"}, 
45     {"id": "boss_dark", "cat": "boss", "name": "Dark Overlord", "price": 100, "color": MAGENTA, "image": "boss_dark.png"}, 
46
47     # Tile themes
48     {"id": "t_forest", "cat": "tile", "name": "Forest Lab", "price": 50, "color": (34, 139, 34), "image": "wall_forest.png"}, 
49
50     # Story
51     {"id": "sty_journal", "cat": "story", "name": "The Lost Journal", "price": 200, "color": WHITE}, 
52 ]

```

```

85     # Gameplay
86     PLAYER_SPEED = 5
87     ENEMY_SPEED = 2
88     CHASE_SPEED = 3.5
89     DETECT_RADIUS = 100
90     BOSS_SPEED = 4          # Speed Boss
91     BOSS_DETECT_RADIUS = 500 # Boss deteksi jauh
92
93     # Map Tags
94     WALL = '1'
95     PLAYER_START = 'P'
96
97     # --- 10 NEW FULLY CONNECTED LEVELS ---
98     # Ukuran Grid: 32 Kolom x 18 Baris
99     # Pastikan semua '0' terhubung ke 'P'
100
101    # LEVEL 1: The Training Hall (Ruang luas, banyak pilar)
102    LEVEL_1 = [
103        "11111111111111111111111111111111",
104        "1P000000000000000000000000000001",
105        "10110110110110110110110110110101",
106        "10000000000000000000000000000001",
107        "10110110110110110110110110110101",
108        "10000000000000000000000000000001",
109        "10110110110110110110110110110101",
110        "10000000000000000000000000000001",
111        "10110110110110110110110110110101",
112        "10000000000000000000000000000001",
113        "10110110110110110110110110110101",
114        "10000000000000000000000000000001",
115        "10110110110110110110110110110101",
116        "10000000000000000000000000000001",
117        "10110110110110110110110110110101",
118        "10000000000000000000000000000001",
119        "10000000000000000000000000000001",
120        "11111111111111111111111111111111",
121    ]
122
123    # LEVEL 2: The Loop (Dua jalur utama melingkar dengan penghubung)

```

```

299    # LEVEL 10: BOSS ARENA (Ruang terbuka lebar)
300    LEVEL_10 = [
301        "11111111111111111111111111111111",
302        "1P000000000000000000000000000001",
303        "10000000000000000000000000000001",
304        "10000000000000000000000000000001",
305        "10001110000000000000000011100001",
306        "10001110000000000000000011100001",
307        "10000000000000000000000000000001",
308        "10000000000000000000000000000001",
309        "10000000000000000000000000000001",
310        "10000000000000000000000000000001",
311        "10000000000000000000000000000001",
312        "10000000000000000000000000000001",
313        "10001110000000000000000011100001",
314        "10001110000000000000000011100001",
315        "10000000000000000000000000000001",
316        "10000000000000000000000000000001",
317        "10000000000000000000000000000001",
318        "11111111111111111111111111111111",
319    ]
320
321    LEVELS = [
322        LEVEL_1, LEVEL_2, LEVEL_3, LEVEL_4, LEVEL_5,
323        LEVEL_6, LEVEL_7, LEVEL_8, LEVEL_9, LEVEL_10
324    ]

```

h. vfx.py

```
vfxpy  X
vfx.py > ...
1  # FILE: vfx.py
2  import pygame
3  import random
4
5  class Particle(pygame.sprite.Sprite):
6      def __init__(self, x, y, color):
7          super().__init__()
8          self.image = pygame.Surface((random.randint(3, 6), random.randint(3, 6)))
9          self.image.fill(color)
10         self.rect = self.image.get_rect(center=(x, y))
11
12         # Physics sederhana
13         self.vx = random.uniform(-3, 3)
14         self.vy = random.uniform(-3, 3)
15         self.life = 255 # Opacity / Durasi
16         self.decay = random.randint(5, 10) # Kecepatan hilang
17
18     def update(self):
19         self.rect.x += self.vx
20         self.rect.y += self.vy
21         self.life -= self.decay
22
23         if self.life <= 0:
24             self.kill()
25         else:
26             self.image.set_alpha(self.life)
27
28 class ParticleManager:
29     def __init__(self):
30         self.group = pygame.sprite.Group()
31
32     def emit(self, x, y, color, amount=10):
33         for _ in range(amount):
34             p = Particle(x, y, color)
35             self.group.add(p)
36
37     def update(self):
38         self.group.update()
39
40     def draw(self, surface):
41         self.group.draw(surface)
```

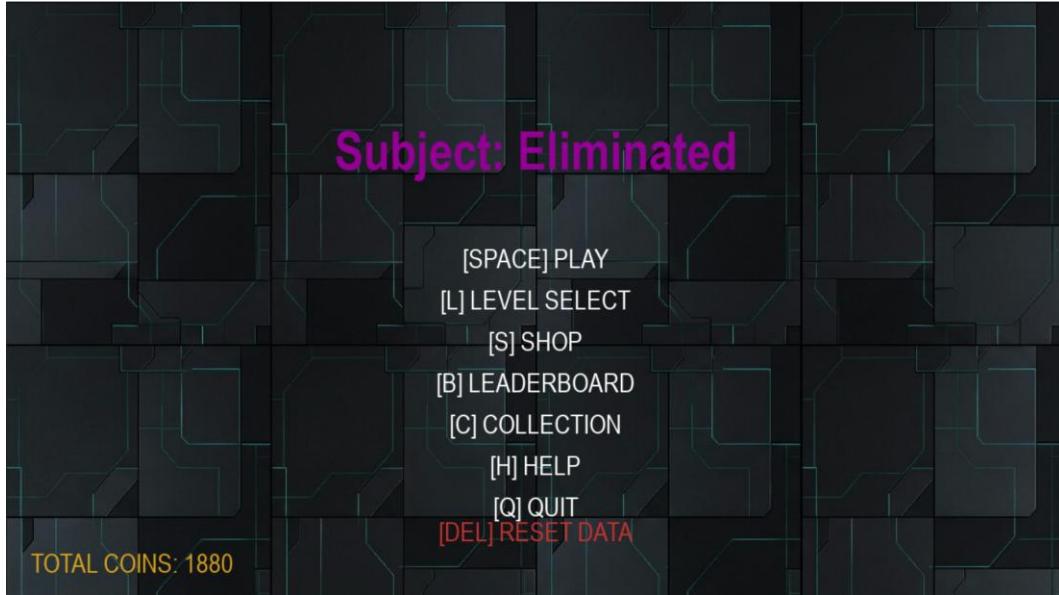
BAB III

HASIL PROYEK

3.1 Tampilan Game

Pada bagian ini ditampilkan beberapa gambar tampilan game sebagai bukti implementasi dari game Ghost - Escape The Lab. Gambar yang ditampilkan meliputi:

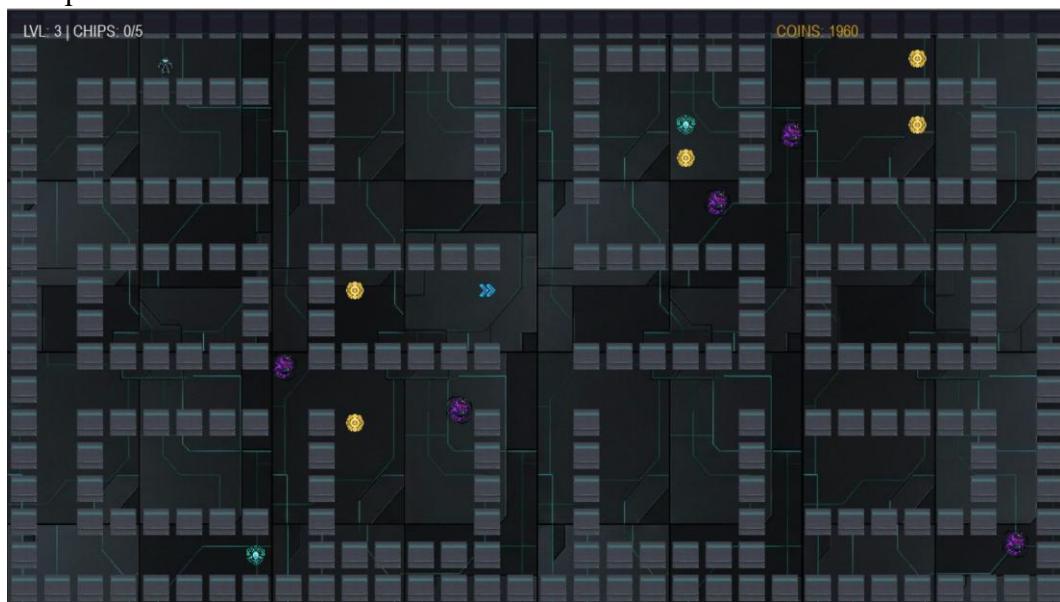
1. Tampilan menu utama



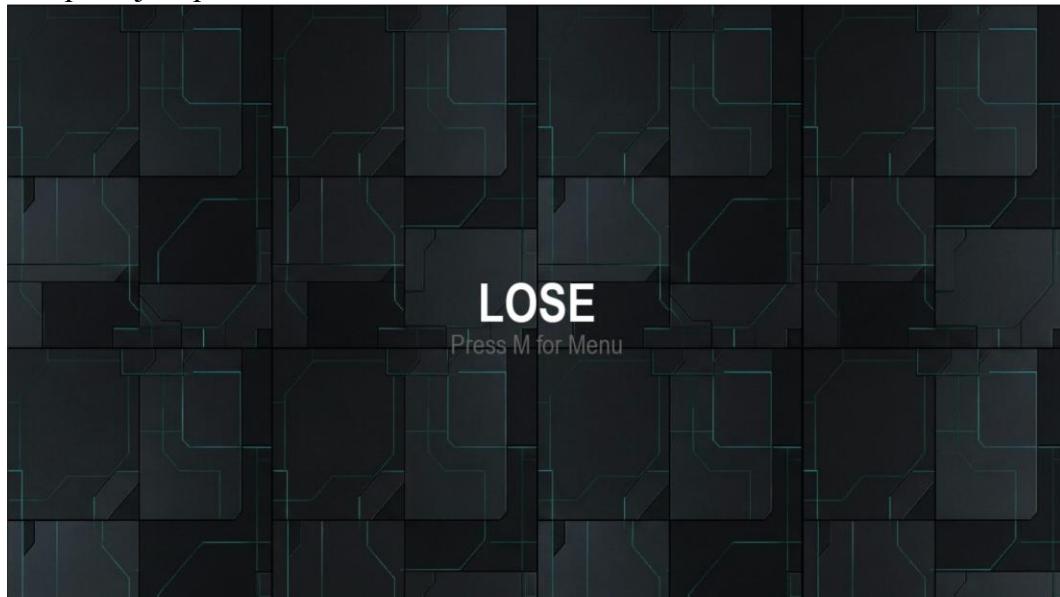
2. Tampilan level 1



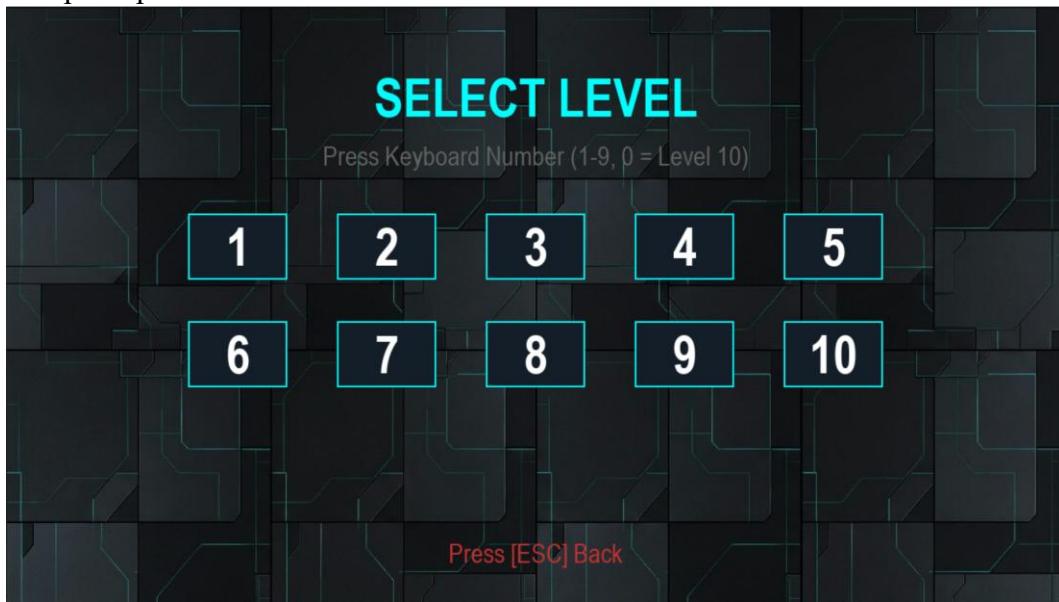
3. Tampilan level 3



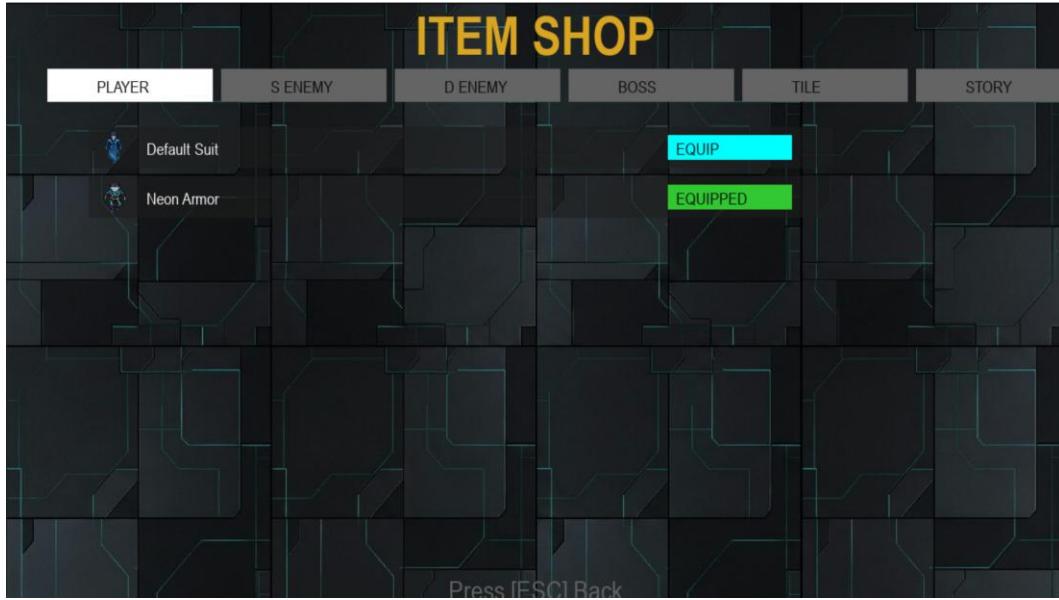
4. Tampilan jika pemain kalah



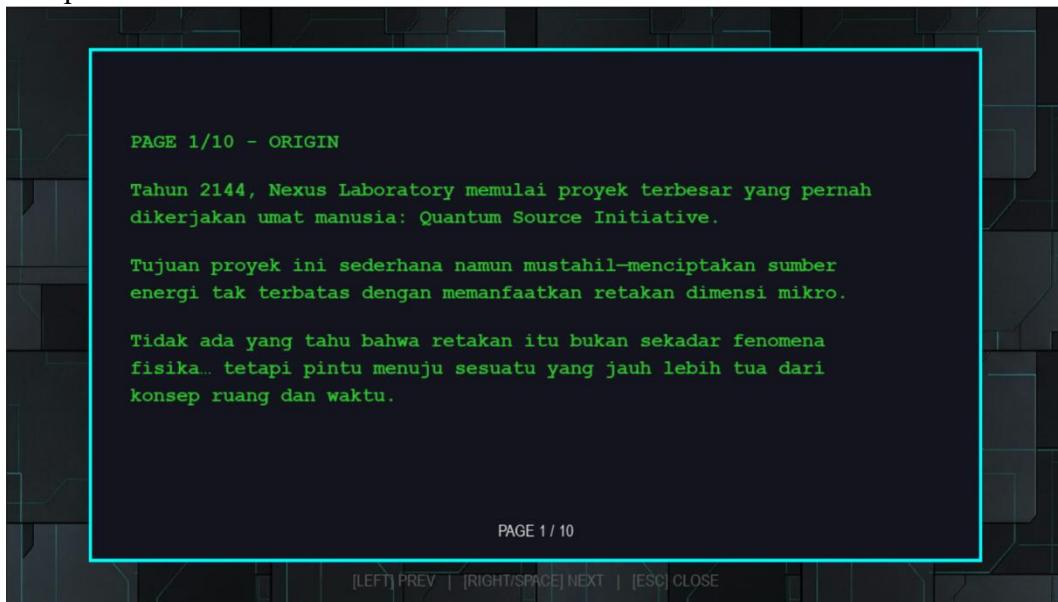
5. Tampilan pemilihan level



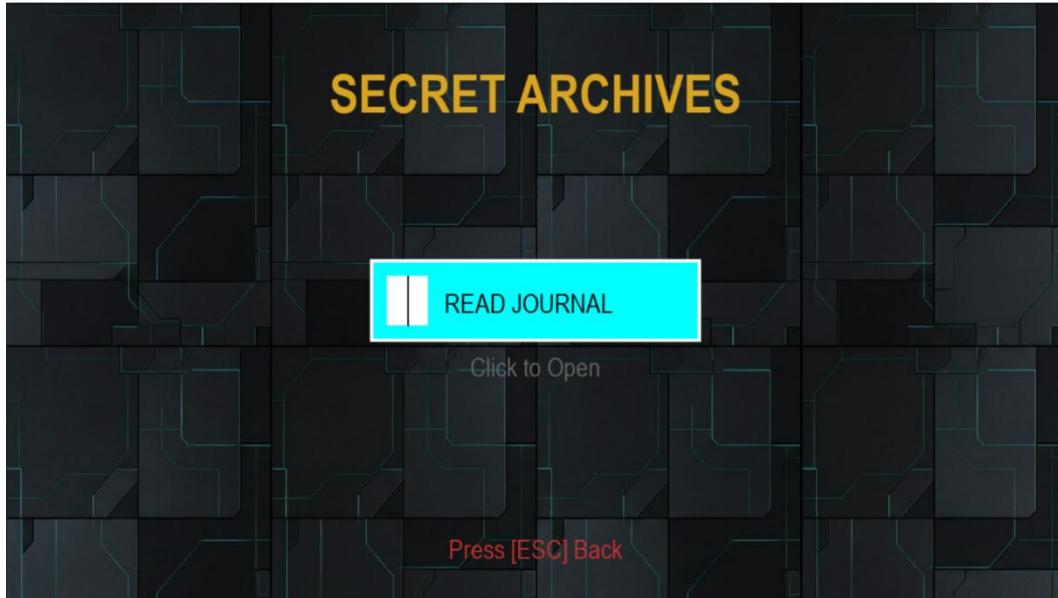
6. Tampilan toko item permainan



7. Tampilan Leaderboard



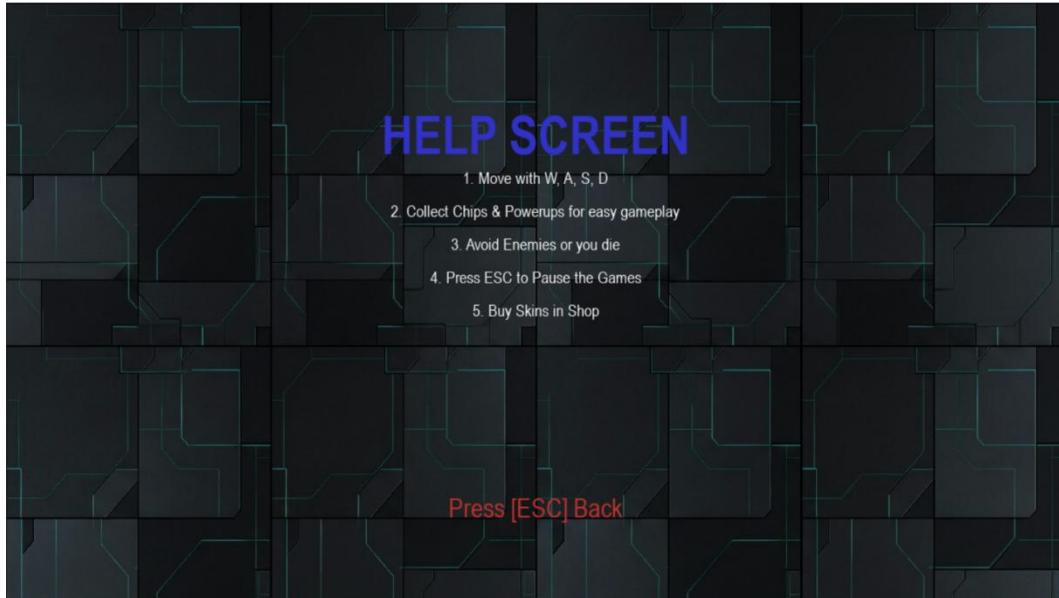
8. Tampilan menu Collection



9. Tampilan Top Score



10. Tampilan menu Help



BAB IV

PENUTUP

4.1 Kesimpulan

Proyek “Ghost - Escape The Lab” telah berhasil dikembangkan sebagai sebuah game 2D top-down berbasis Python dan library Pygame yang mengimplementasikan konsep-konsep inti Pemrograman Berorientasi Objek secara komprehensif. Seluruh fitur utama yang direncanakan, mulai dari sistem pergerakan pemain, pembangunan maze berbasis grid, AI musuh dengan mode patrol dan chase, pengumpulan Energy Chips sebagai syarat kemenangan, hingga mekanisme power-up sementara, telah terwujud dengan baik.

Penerapan prinsip OOP dapat terlihat dari penggunaan encapsulation pada pengelolaan data sensitif, inheritance pada hierarki kelas entitas (Player, Enemy, Item, dan turunannya), serta polymorphism pada metode-metode seperti update() dan apply() yang memiliki perilaku berbeda sesuai jenis objeknya. Selain itu, penambahan fitur lanjutan seperti sistem save/load, shop kustomisasi visual (skin player, enemy, boss, dan tile), leaderboard, serta narasi lore mendalam melalui “The Lost Journal” diharapkan dapat memperkaya pengalaman bermain game Ghost - Escape The Lab ini.

Hasil akhir yang berupa game dengan 10 level permainan, efek visual partikel, transisi antar menu yang halus, serta keseimbangan gameplay yang menantang diharapkan sudah cukup untuk memenuhi tujuan akademis serta menghasilkan sebuah permainan yang fungsional, estetis, dan menyenangkan untuk dimainkan.

4.2 Saran dan Pengembangan Lanjutan

Proyek “Ghost - Escape The Lab” memiliki fondasi yang kuat dan potensi untuk dikembangkan lebih jauh menjadi sebuah game yang memiliki lebih banyak fitur. Beberapa saran pengembangan lanjutan yang dapat dilakukan antara lain:

1. Penambahan Level dan Variasi Maze

Menambah jumlah level lebih dari 10 dengan desain maze yang semakin kompleks, termasuk tema visual berbeda untuk setiap chapter (misalnya laboratorium bawah tanah, area berbahaya, atau fasilitas karantina yang terkontaminasi).

2. Perluasan Sistem Narasi

Mengembangkan lebih banyak buku atau catatan rahasia yang dapat dibeli di shop, dengan cerita cabang (branching story) yang terbuka berdasarkan pencapaian pemain.

3. Fitur Multiplayer atau Co-op

Mengimplementasikan mode lokal co-op di mana dua pemain dapat bekerja sama mengumpulkan chip sambil menghindari musuh, atau mode kompetitif dengan leaderboard online.

4. Efek Visual dan Audio Lebih Lanjut

Menambahkan lebih banyak variasi partikel, animasi karakter, serta soundtrack dinamis yang berubah sesuai situasi (patrol vs chase vs boss fight).

5. Sistem Achievement dan Progression

Menambahkan sistem achievement yang memberikan reward berupa koin tambahan atau skin eksklusif, serta daily challenge dengan maze yang acak.

Dengan fondasi OOP yang solid dan struktur kode yang modular, semua pengembangan di atas dapat dilakukan dengan relatif mudah tanpa harus merombak keseluruhan arsitektur awal proyek.