

**LAPORAN TUGAS BESAR
PEMROGRAMAN BERORIENTASI OBJEK**

“JUPPA JUNGLE”



KELOMPOK 08 :

1. Fadzilah Saputri (NIM.123140149)
2. Hildyah Maretasya Araffad (NIM.123140151)
3. Martino Kelvin (NIM.123140165)
4. Raisya Syifa Saleh (NIM.123140169)
5. Atika Adelia (NIM.123140172)

**PROGRAM STUDI TEKNIK INFORMATIKA
FAKULTAS TEKNOLOGI INDUSTRI
INSTITUT TEKNOLOGI SUMATERA
2025**

DAFTAR ISI

A. DESKRIPSI PENUGASAN.....	3
B. NAMA GAME	3
C. KATEGORI GAME.....	3
D. ENTITAS	4
E. DESKRIPSI GAME	5
F. TAMPILAN GAME	6
1. Tampilan Awal Game (Overworld).....	6
2. Tampilan Akhir Game	7
3. Tampilan Saat Bermain di Map	7
G. KELAS	8
H. OBJEK	9
I. ENKAPSULASI.....	10
J. PEWARISAN	10
K. POLIMORFISME.....	11
L. ABSTRAKSI.....	12
M. PENANGANAN KESALAHAN	12
N. GRAPHICAL USER INTERFACE	13
O. DIAGRAM UNIFIED MODELING LANGUAGE	15
P. KODE.....	17
1. main.py (Class Game)	17
1.1 Contoh Objek Pada Class Game	18
2. Class Player	19
3. Class Particle	22
4. Class Node.....	22
5. Class Icon.....	23
6. Class Overworld	23
6.1 Contoh Objek Class Overworld dari Class Sky	24
7. Class Level.....	25
7.1 Contoh Objek Class Level dari Class Sky, Class Water, Class Clouds	29
7.2 Contoh Objek Pada Class Level dari Class Palm.....	29
7.3 Contoh Objek Pada Class Level dari Class Palm.....	29
7.4 Contoh Objek Pada Class Level dari Class Palm.....	29
8. Class Enemy	29
9. Class Sky	30
10. Class Water.....	31
11. Class Clouds	31
12. Class UI	32
12.1 Contoh Enkapsulasi Pada Class UI	32
13. Class Tile	33
13.1 Contoh Polimorfisme Pada Class Tile.....	33
14. Class StaticTile	33

14.1 Contoh Polimorfisme Pada Class StaticTile	33
15. Class Crate	34
15.1 Contoh Polimorfisme Pada Class Crate	34
16. Class AnimatedTile	35
16.1 Contoh Polimorfisme Pada Class AnimatedTile.....	35
17. Class TileInterface	35
17.1 Contoh Abstrak Pada Class TileInterface	35
18. Class Coin dan Class Palm	36
18.1 Contoh Enkapsulasi Pada Class UI	37
18.2 Contoh Polimorfisme Pada Class Coin dan Class Palm.....	37
19. Support.....	37
20. Settings	38
21. Game_Data	38
Q. LAMPIRAN.....	41

A. DESKRIPSI PENUGASAN

No.	NIM	Nama	Penugasan
1.	123140149	Fadzilah Saputri	Programmer, Laporan, Game Design (Design Karakter), UML, Presentasi
2.	123140151	Hildyah Maretasya Araffad	Programmer, Game Designer (Design Lingkungan/Area), PPT, Presentasi
3.	123140165	Martino Kelvin	Lead Programmer, UML, Sound Engineer, Presentasi
4.	123140169	Raisya Syifa Saleh	Programmer, Sound Engineer, Laporan, Presentasi
5.	123140172	Atika Adelia	Programmer, Game Design (Design Karakter), PPT, UML Presentasi

B. NAMA GAME

Nama “**Juppa Jungle**” dipilih karena menggambarkan inti dari game ini, yaitu ‘Jungle’ berarti hutan dan ‘Juppa’ adalah nama karakter player. Kata ‘Jungle’ menggambarkan latar permainan yang penuh dengan pepohonan, rintangan alam, dan suasana hutan yang membentuk suatu dunia petualangan di mana tempat player melompat, menghindari musuh, dan mengumpulkan koin. Sedangkan, ‘Juppa’ merupakan seorang penjelajah pemberani yang harus menavigasi berbagai rintangan untuk mencapai gerbang level berikutnya.

C. KATEGORI GAME

‘Juppa Jungle’ adalah game **action** dan **adventure** yang menggabungkan eksplorasi mendalam dengan aksi dinamis di tengah hutan yang penuh misteri. Game ini termasuk kategori **action** karena pemain harus mengendalikan Juppa dalam berbagai tantangan fisik, seperti melompat, menghindari musuh, mengumpulkan koin dan berinteraksi dengan lingkungan secara cepat untuk bertahan hidup. Elemen action semakin terasa dengan adanya sistem navigasi yang menuntut dan menguji kecerdasan dan refleks pemain untuk melewati berbagai tantangan. Game ini juga masuk dalam kategori **adventure** karena menawarkan pengalaman eksplorasi hutan yang luas untuk menemukan jalur tersembunyi dan mencapai jalan keluar.

D. ENTITAS

Entitas adalah sesuatu yang nyata atau bisa dikenali, yang memiliki data dan penting dalam sistem yang sedang dianalisis atau dikembangkan. Entitas bisa berupa orang, benda. Contoh entitas dalam game Juppa Jungle, yaitu :

No	Entitas	Tugas	Aktivitas
1.	Player	Bertahan hidup dan mengumpulkan koin	Berlari dan melompat untuk mengumpulkan koin, serta menghindari dan menyerang musuh untuk bertahan hidup
2.	Enemy	Menyerang player	Enemy akan menyerang player jika player menghampirinya
3.	Senjata	Menyerang Enemy	Senjata berupa sword akan dilayangkan player ke arah enemy
4.	Coin	Mengumpulkan koin sebanyak mungkin	Koin akan tersebar di beberapa titik yang dapat dikumpulkan oleh player sebanyak mungkin
5.	Health	Menyimpan nyawa player untuk menentukan player hidup atau mati	Nyawa akan terus berkurang jika player tidak melindungi dirinya dari enemy sehingga player harus menyimpan nyawa hingga mencapai akhir dari level tersebut
6.	Gerbang Level	Sebagai tujuan akhir setiap level	Player akan mencapai akhir gerbang di setiap level sebagai penghubung menuju level selanjutnya
7.	Platforms	Sebagai tempat player berlari dan melompat	Player dan enemy akan berlari dan melompat di atas tanah

E. DESKRIPSI GAME

Alur Permainan :

1. Pemain mengendalikan karakter Juppa dari titik awal
 - Saat level dimulai, posisi awal karakter juppa akan diatur pada titik awal platforms
 - Pemain menggunakan keyboard untuk mengatur karakter Juppa ke kanan dan ke kiri, serta melompat
2. Pengumpulan Koin
 - Koin akan tersebar di beberapa titik platform tertentu
 - Koin bisa berada di atas platform, di antara rintangan, atau di lokasi tersembunyi (melompat dan membunuh musuh untuk menemukan koin yang tersembunyi)
 - Ketika karakter dikendalikan untuk menyentuh koin maka koin akan hilang dari platform dan jumlah koin yang terkumpul bertambah
3. Menghindari Musuh dan Rintangan Alam
 - Karakter enemy akan ditempatkan di beberapa bagian platform dan bergerak secara otomatis
 - Karakter Juppa akan dikendalikan untuk menghindar dan menyerang musuh, apabila Juppa menyentuh enemy maka nyawa akan berkurang sehingga perlunya pengendalian untuk menghindari dan menyerah musuh
 - Karakter Juppa akan dikendalikan untuk melompat dari satu platform tanah ke platform tanah lainnya agar tidak terjatuh ke jurang dan menyebabkan nyawa hilang sepenuhnya hingga karakter juppa mati
4. Menuju Gerbang Level
 - Karakter memasuki level dari titik awal
 - Karakter akan melewati beberapa rintangan, seperti melawan musuh dan melewati rintangan agar bisa menuju ke level selanjutnya
 - Karakter akan mengumpulkan koin sebanyak mungkin, pada saat tertentu enemy akan muncul sebagai lawan, karakter bisa mengalahkan enemy dengan menggunakan senjata.
 - Setelah karakter melewati semua rintangan dan musuh, di akhir tujuan karakter akan mencapai titik akhir dari level, dan akan menuju ke level selanjutnya.
5. Sistem Nyawa
 - Jika karakter juppa menghindari dan menyentuh musuh, maka nyawa akan berkurang.

- Jika nyawa habis, maka permainan akan berakhir dan mengulang kembali permainan di level tersebut.

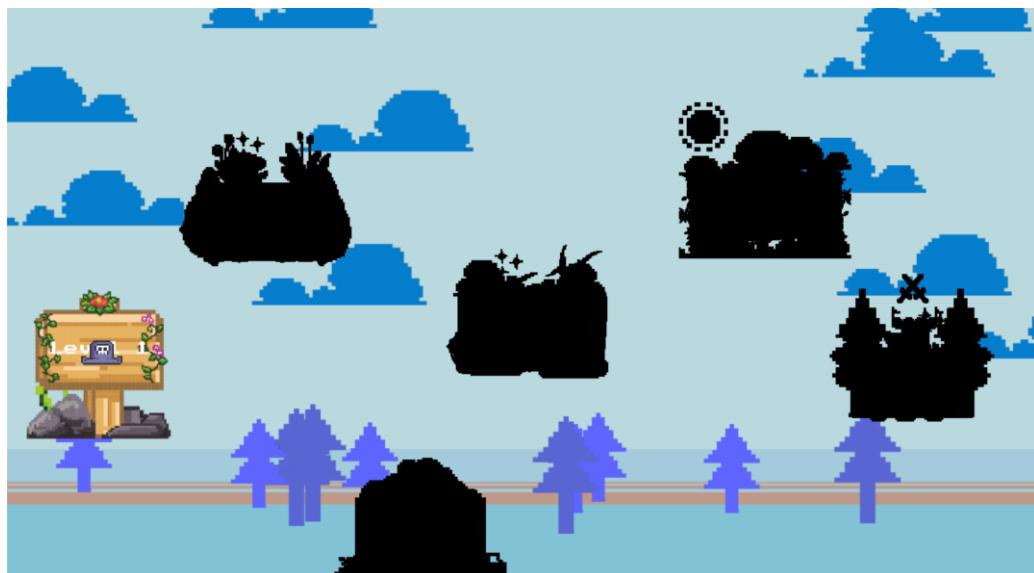
Kondisi Permainan

1. Hidup : Kondisi Hidup ditandai dengan adanya nyawa pada karakter Juppa sehingga Juppa masih bisa berlari dan melompat untuk menghindari dan menyerang enemy, serta mengumpulkan koin sebanyaknya
2. Mati : Kondisi Mati terjadi apabila karakter Juppa tidak melakukan penyerangan terhadap musuh sehingga nyawa-nya akan berkurang hingga habis dan Juppa akan mati, selain itu apabila Juppa terjatuh ke jurang maka nyawa-nya secara otomatis habis atau mati

F. TAMPILAN GAME

1. Tampilan Awal Game (Overworld)

Tampilan awal yang menampilkan seluruh level yang mana diawali dengan level 1 hingga level 6. Berikut adalah tampilan awal game :



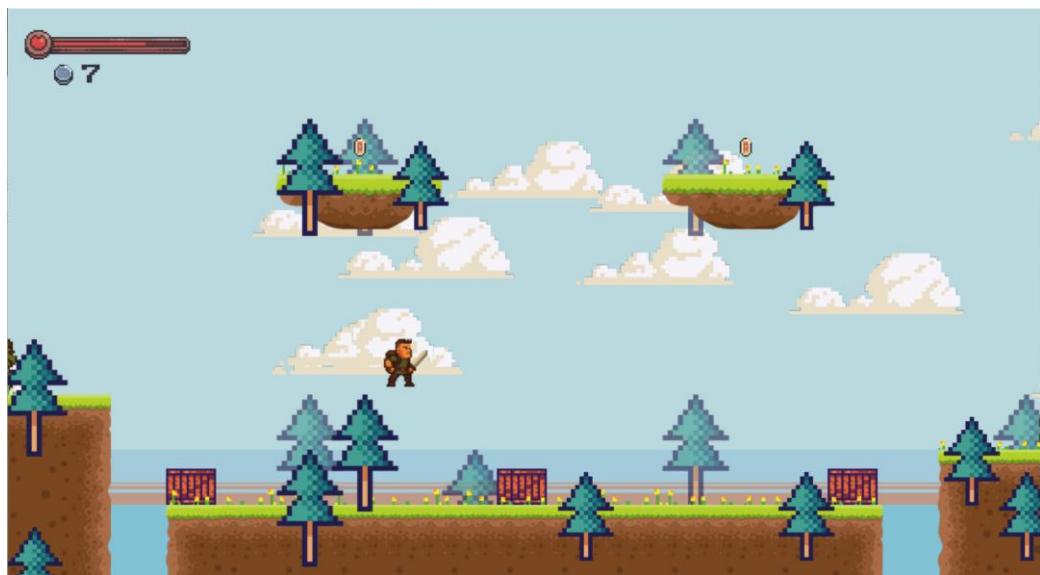
2. Tampilan Akhir Game

Tampilan akhir game adalah menampilkan seluruh level yang sudah terlewati. Berikut adalah tampilan akhir game, yaitu :



3. Tampilan Saat Bermain di Map

Berikut adalah tampilan saat karakter Juppa bermain di map level tertentu :



G. KELAS

Kelas adalah struktur dalam OOP yang digunakan untuk mendefinisikan atribut dan metode dari objek. Pada code game ini terdapat beberapa kelas. Berikut adalah contoh kelas pada code game ini, yaitu :

Objek	Penjelasan Objek
Game	Mengelola logika utama permainan
Enemy	Karakter musuh yang memiliki logika dan atribut khusus
Player	Karakter pemain yang merepresentasikan "Juppa" dengan logika dan atribut khusus
Level	Mengatur dan memproses level dalam game
UI	Mengelola elemen-elemen antarmuka grafis (HUD) dalam game
Coin	Item yang dapat dikumpulkan pemain dalam permainan
Sky	Elemen permainan berupa langit dengan cuaca cerah
Tile	Bagian dari peta platform permainan
Animated Tile	Meningkatkan efek visual dan interaktivitas
TileInterface	Mengelola sebuah abstraksi yang mengatur cara tabrakan dan mengatur jenis tile/elemen
StaticTile	Digunakan untuk tile tetap seperti tanah, platform batu, dinding, dll.
Crate	Mengelola tile/elemen berupa crate
Coin	Mengelola tile/elemen berupa Koin
Palm	Mengelola tile/elemen berupa pohon
Water	Mengelola tile/elemen berupa Air
Clouds	Mengelola tile/elemen berupa awan
Overworld	Mengelola elemen-elemen yang berada di halaman utama
Node	Mengelola animasi 'level' di halaman utama
Icon	Mengelola animasi karakter 'juppa' yang bergerak dari animasi satu level ke level lain
ParticleEffect	Mengelola efek yang muncul pada saat karakter juppa

Objek	Penjelasan Objek
	berlari, melompat, dan terjatuh

H. OBJEK

Objek adalah perwujudan nyata dari suatu kelas, yang berisi data berupa atribut dan metode sesuai definisi kelas tersebut. Pada code ini terdapat beberapa objek yang berasal dari beberapa kelas. Berikut adalah contoh penerapan objek pada code game ini, yaitu :

Objek	Penjelasan
player = Player(position)	Mengelola karakter pemain
tile = Tile(size, x, y)	Mengelola ubin (lantai/platform) dalam level
enemy = Enemy(Size, x, y)	Mengelola musuh yang bergerak otomatis dan menyerang player
coin = Coin(size, x, y, path, value)	Mengelola koin yang dapat dikoleksi pemain dengan nilai tertentu
crate = Crate(Size, x, y)	Mengelola kotak digunakan sebagai rintangan atau interaktif dalam game
palm = Palm(size, x, y, path, offset)	Mengelola dekorasi pohon pada latar belakang level
sky = Sky(horizon, style)	Mengelola pengaturan latar langit dan elemen dekoratif dalam permainan
water = Water(top, level_width)	Mengelola animasi air di level
Clouds = Clouds(horizon, level_width, cloud_number)	Mengelola kumpulan awan yang ada di latar belakang game
Node = Node(pos, status, icon_speed, path)	Mengelola titik di peta overworld yang menunjukkan level dengan animasi berbeda sesuai statusnya: terbuka (bergerak) atau terkunci (gelap dan diam)
icon = Icon(pos)	Mengelola ikon permain yang bergerak di peta overworld
overworld = Overworld(start_level, max_level, surface, create_level)	Mengelola peta overworld yang mengatur navigasi level dan ikon
particle_effect = ParticleEffect(pos, type)	Mengelola efek partikel animasi untuk visual seperti debu atau ledakan saat karakter loncat dan terjatuh
ui = UI(surface)	Mengelola tampilan pengguna yang mengatur health bar dan koin di layar
level = Level(current_level, surface, create_overworld, change_coins, change_health)	Mengelola level game yang memuat seluruh elemen dan logika level

Objek	Penjelasan
game = Game()	Mengelola keseluruhan game yang memuat jalannya game, status, dan perpindahan antar mode

I. ENKAPSULASI

Pada code game ini terdapat contoh enkapsulasi pada Class Coin dan Class UI yang mengimplementasikan penerapan setter dan getter. Berikut adalah contoh enkapsulasi dalam code game ini, yaitu :

Kelas	Atribut
Coin	_value
UI	_current_health _full_health

J. PEWARISAN

Pewarisan memungkinkan objek-objek dalam permainan untuk memiliki kemampuan dan sifat yang terstruktur. Misalnya, Class Player, Tile, Node, Icon, dan ParticleEffect mewarisi kelas pygame.sprite.Sprite, Class Tile memiliki dua turunan utama, yaitu StaticTile dan AnimatedTile. Berikut adalah contoh pewarisan yang diturunkan dari beberapa kelas induk, yaitu :

Kelas Induk	Kelas Turunan
pygame.sprite.Sprite	Player Tile Node Icon ParticleEffect Player Tile
AnimatedTile	Enemy
Tile	Static Tile AnimatedTile
StaticTile	Crate
AnimatedTile	Coin Palm

K. POLIMORFISME

Konsep polimorfisme dalam kode dapat dilihat dengan jelas pada metode `get_tile_type()`, `get_collision_type()`, dan `update()`. Ketiga metode ini mencerminkan sebuah metode yang memiliki nama sama dapat berperilaku berbeda tergantung pada objek yang mengimplementasikannya. Misalnya, Class Tile memiliki metode `update()` untuk menggeser posisi tile berdasarkan shift, maka kelas turunan seperti AnimatedTile, Coin, dan Palm bisa melakukan override terhadap metode ini untuk menambahkan logika animasi. Begitu juga dengan metode `get_tile_type()` dan `get_collision_type()`, jika metode ini diimplementasikan dalam kelas Tile, maka kelas-kelas seperti Crate, Coin, atau Palm bisa menyediakan versinya sendiri. Berikut adalah contoh polimorfisme pada code game ini, yaitu :

Kelas	Metode
Tile	<code>update()</code>
Tile	
StaticTile	
Crate	
AnimatedTile	<code>get_tile_type()</code>
Coin	
Palm	
Tile	
StaticTile	
Crate	
AnimatedTile	<code>get_collision_type()</code>
Coin	
Palm	

Pada code game ini, digunakan jenis polimorfisme yaitu polimorfisme dengan Inheritance (Method Overriding) di mana kelas-kelas turunan seperti StaticTile, Crate, AnimatedTile, Coin, dan Palm mengimplementasikan ulang metode `get_tile_type()` dan `get_collision_type()` yang didefinisikan pada kelas induk Tile. Polimorfisme jenis ini memungkinkan setiap objek dari kelas berbeda untuk merespons pemanggilan metode dengan cara yang sesuai karakteristiknya masing-masing, meskipun metode tersebut memiliki nama yang sama.

Dengan demikian, walaupun pemanggilan metode dilakukan secara seragam melalui referensi kelas induk, hasil yang diperoleh bersifat dinamis dan spesifik sesuai tipe objek yang digunakan. Pendekatan ini sesuai dengan definisi polimorfisme dalam pemrograman berorientasi objek yang menekankan kemampuan sebuah fungsi untuk mengambil banyak bentuk (many forms) melalui mekanisme pewarisan dan overriding fungsi pada subclass, sehingga meningkatkan fleksibilitas dan modularitas kode.

L. ABSTRAKSI

Pada code game ini, terdapat satu kelas yang mengandung abstraksi yaitu class `TileInterface`, yang mana memiliki dua metode abstrak, yaitu `get_tile_type` berguna untuk mendapatkan tipe dari sebuah "tile" sebagai elemen dalam permainan yang dapat digerakkan atau dihancurkan. Lalu, `get_collision_type` berguna untuk mendapatkan jenis dari "collision" atau interaksi antara objek yang berbeda, seperti apakah objek tersebut dapat berinteraksi atau tidak dengan objek lain di dalam permainan. Berikut adalah contoh abstraksi pada code game, yaitu :

Kelas Abstrak	Metode Abstrak
<code>TileInterface</code>	<code>get_tile_type</code> <code>get_collision_type</code>

M. PENANGANAN KESALAHAN

Penanganan kesalahan pada game dilakukan untuk meningkatkan kenyamanan dan pengalaman pengguna saat bermain. Salah satu bug yang ditemukan terdapat pada

fitur loncat (jump), di mana karakter justru melompat ke bawah alih-alih ke atas saat berada dekat dengan objek seperti kotak atau pohon. Bug ini mengganggu mekanisme dasar permainan dan dapat menyulitkan pemain dalam melewati rintangan atau mencapai area tertentu. Untuk mengatasi masalah ini, logika deteksi tabrakan (collision detection) dan arah gaya loncatan dapat diperbaiki agar karakter dapat melompat ke arah yang benar, yaitu ke atas, meskipun berada dalam posisi berdekatan dengan objek di sekitarnya. Dengan perbaikan ini, kontrol karakter menjadi lebih intuitif dan sesuai dengan harapan pengguna, sehingga gameplay terasa lebih responsif dan menyenangkan.

N. GRAPHICAL USER INTERFACE

GUI pada game Juppa Jungle dibuat menggunakan Pygame sebagai pustaka utama dalam pengembangan antarmuka grafisnya. Pygame adalah pustaka (*library*) dalam bahasa pemrograman Python yang digunakan untuk membuat game dan aplikasi multimedia berbasis grafis. Pygame menyediakan berbagai fungsi untuk menangani gambar, suara, animasi, dan interaksi pengguna. Beberapa alasan memilih Pygame untuk GUI game Juppa Jungle yaitu sebagai berikut :

1. Kemudahan Penggunaan

Pygame menawarkan antarmuka yang mudah digunakan dan dipahami, terutama bagi pengembang yang sudah familiar dengan Python. Sifatnya yang sederhana namun fleksibel menjadikannya pilihan ideal untuk proyek game berskala kecil hingga menengah.

2. Fleksibilitas

Pygame menyediakan berbagai modul yang dapat menangani aspek penting dalam pengembangan game. Modul-modul tersebut mencakup grafik, suara, dan input pengguna serta fungsi yang membantu dalam pemrosesan game. Dengan fitur ini, pengembang bisa menciptakan permainan dengan berbagai elemen tanpa harus bergantung pada pustaka eksternal tambahan.

3. Komunikasi yang Aktif dan Dukungan Luas

Salah satu keunggulan Pygame adalah dukungan dari komunitas yang aktif. Banyak dokumentasi, tutorial, dan contoh kode yang tersedia secara online untuk membantu pengembang pemula maupun yang sudah berpengalaman. Ini memudahkan pengembang untuk mempelajari dan mengatasi masalah yang mungkin timbul selama pengembangan game.

4. Kinerja yang Optimal untuk Game 2D

Meskipun Pygame tidak secepat pustaka grafis yang ditulis dalam bahasa pemrograman tingkat rendah seperti C++ atau Rust, Pygame tetap menawarkan kinerja yang cukup baik untuk game 2D dengan kompleksitas menengah.

Kemampuannya dalam mengelola sprite dan animasi dengan baik membuatnya sangat sesuai untuk pengembangan game seperti Juppa Jungle.

5. Dukungan Multi-Platform

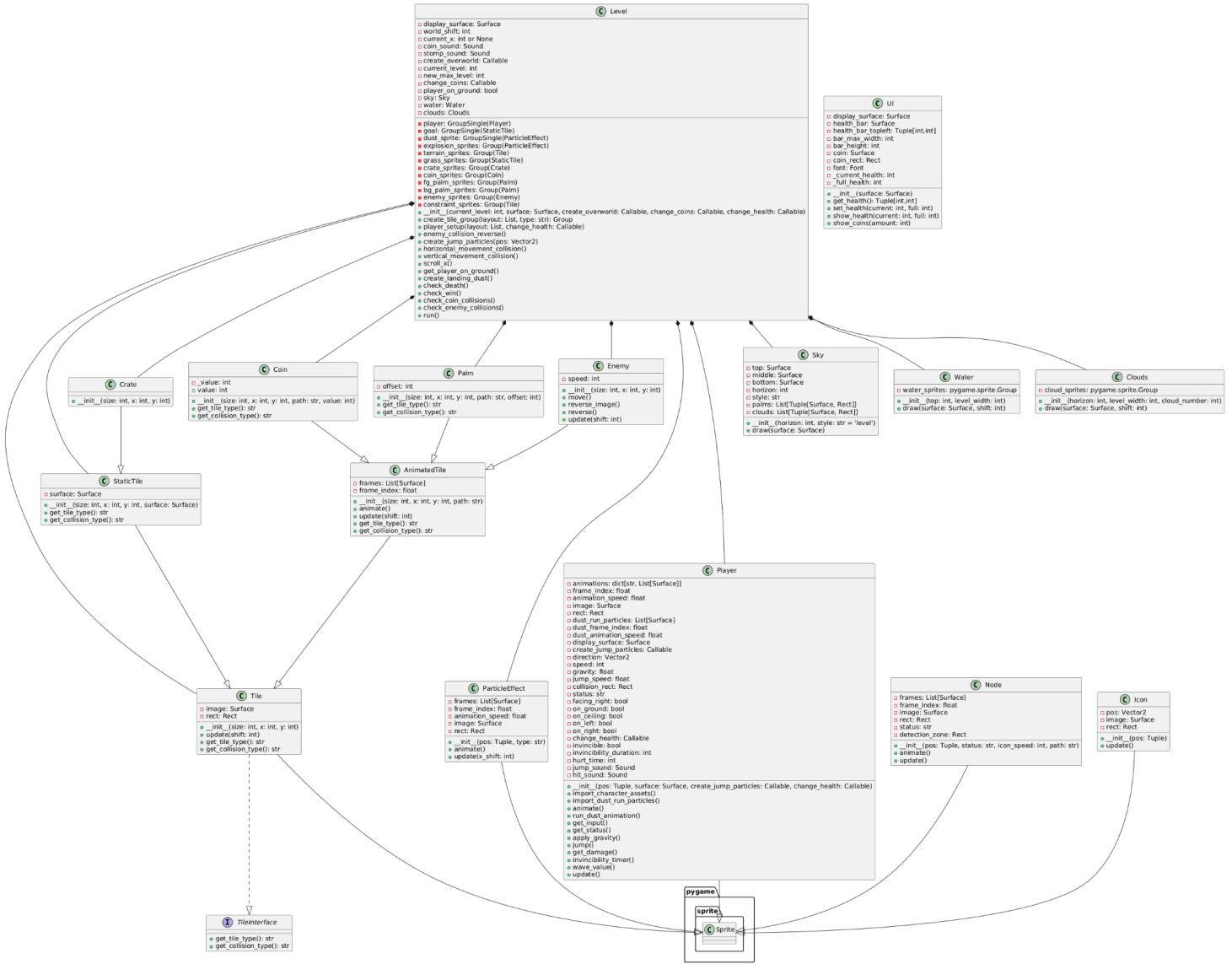
Pygame bersifat lintas platform, sehingga memungkinkan game yang dikembangkan untuk berjalan di berbagai sistem operasi utama seperti Windows, macOS, dan Linux. Hal ini memberikan keuntungan bagi pengembang dalam menciptakan game yang dapat diakses oleh lebih banyak pengguna tanpa perlu modifikasi yang kompleks.

6. Prototyping yang Cepat dan Efisien

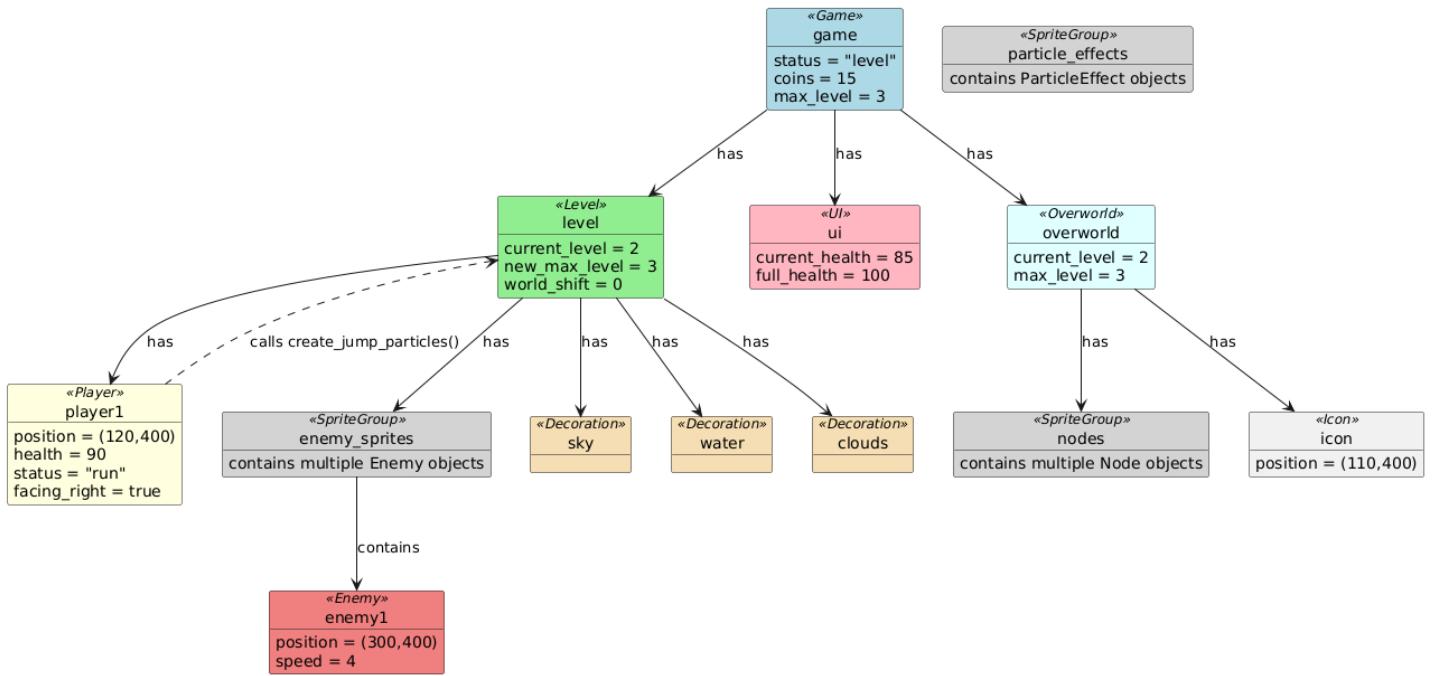
Dengan Pygame, pengembang dapat dengan cepat membuat dan menguji prototipe game sebelum merancang versi finalnya. Hal ini memungkinkan fleksibilitas dalam menyesuaikan desain, mencoba fitur baru, dan mengoptimalkan pengalaman pengguna tanpa harus melalui proses yang rumit.

Dengan memilih Pygame untuk GUI, kami dapat dengan cepat membuat dan menguji prototipe game, serta menyesuaikan desain game dengan kebutuhan dan preferensi kami.

O. DIAGRAM UNIFIED MODELING LANGUAGE



➤ DIAGRAM OBJEK



P. KODE

1. main.py (Class Game)

```
class Game:
    def __init__(self):
        # game attributes
        self.max_level = 2
        self.max_health = 100
        self.cur_health = 100
        self.coins = 0

        # audio
        self.level_bg_music = pygame.mixer.Sound('../audio/level_music.wav')
        self.overworld_bg_music = pygame.mixer.Sound('../audio/overworld_music.wav')

        # overworld creation
        self.overworld = Overworld(0, self.max_level, screen, self.create_level)
        self.status = 'overworld'
        self.overworld_bg_music.play(loops = -1)

        # user interface
        self.ui = UI(screen)

    def create_level(self, current_level):
        self.level = Level(current_level, screen, self.create_overworld,
                           self.change_coins, self.change_health)
        self.status = 'level'
        self.overworld_bg_music.stop()
        self.level_bg_music.play(loops = -1)

    def create_overworld(self, current_level, new_max_level):
        if new_max_level > self.max_level:
            self.max_level = new_max_level
        self.overworld = Overworld(current_level, self.max_level, screen, self.create_level)
        self.status = 'overworld'
        self.overworld_bg_music.play(loops = -1)
        self.level_bg_music.stop()

    def change_coins(self, amount):
        self.coins += amount

    def change_health(self, amount):
        self.cur_health += amount

    def check_game_over(self):
        if self.cur_health <= 0:
            self.cur_health = 100
            self.coins = 0
            self.max_level = 0
            self.overworld = Overworld(0, self.max_level, screen, self.create_level)
            self.status = 'overworld'
            self.level_bg_music.stop()
            self.overworld_bg_music.play(loops = -1)

    def run(self):
        if self.status == 'overworld':
            self.overworld.run()
        else:
            self.level.run()
            self.ui.show_health(self.cur_health, self.max_health)
            self.ui.show_coins(self.coins)
            self.check_game_over()
```

```

pygame.init()
screen = pygame.display.set_mode((screen_width,screen_height))
clock = pygame.time.Clock()
game = Game()

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

    screen.fill('grey')
    game.run()

    pygame.display.update()
    clock.tick(60)

```

1.1 Contoh Objek Pada Class Game

➤ Dari Class UI

```

# user interface
self.ui = UI(screen)

```

➤ Dari Class Level

```

self.level = Level(current_level,screen,self.create_overworld,
                    self.change_coins,self.change_health)

```

➤ Dari Class Overworld

```

self.overworld = Overworld(current_level,self.max_level,screen,self.create_level)

self.overworld = Overworld(0,self.max_level,screen,self.create_level)

```

2. Class Player

```
class Player(pygame.sprite.Sprite):
    def __init__(self, pos, surface, create_jump_particles, change_health):
        super().__init__()
        self.import_character_assets()
        self.frame_index = 0
        self.animation_speed = 0.15
        self.image = self.animations['idle'][self.frame_index]
        self.rect = self.image.get_rect(topleft = pos)

        # dust particles
        self.import_dust_run_particles()
        self.dust_frame_index = 0
        self.dust_animation_speed = 0.15
        self.display_surface = surface
        self.create_jump_particles = create_jump_particles

        # player movement
        self.direction = pygame.math.Vector2(0,0)
        self.speed = 8
        self.gravity = 0.8
        self.jump_speed = -16
        self.collision_rect = pygame.Rect(self.rect.topleft,(50,self.rect.height))

        # player status
        self.status = 'idle'
        self.facing_right = True
        self.on_ground = False
        self.on_ceiling = False
        self.on_left = False
        self.on_right = False

        # health management
        self.change_health = change_health
        self.invincible = False
        self.invincibility_duration = 500
        self.hurt_time = 0
```

```

# audio
    self.jump_sound = pygame.mixer.Sound('../audio/effects/jump.wav')
    self.jump_sound.set_volume(0.5)
    self.hit_sound = pygame.mixer.Sound('../audio/effects/hit.wav')

def import_character_assets(self):
    character_path = '../graphics/character/'
    self.animations = {'idle':[], 'run':[], 'jump':[], 'fall':[]}

    for animation in self.animations.keys():
        full_path = character_path + animation
        self.animations[animation] = import_folder(full_path)

def import_dust_run_particles(self):
    self.dust_run_particles = import_folder('../graphics/character/dust_particles/run')

def animate(self):
    animation = self.animations[self.status]

    # loop over frame index
    self.frame_index += self.animation_speed
    if self.frame_index >= len(animation):
        self.frame_index = 0

    image = animation[int(self.frame_index)]
    if self.facing_right:
        self.image = image
        self.rect.bottomleft = self.collision_rect.bottomleft
    else:
        flipped_image = pygame.transform.flip(image, True, False)
        self.image = flipped_image
        self.rect.bottomright = self.collision_rect.bottomright

    if self.invincible:
        alpha = self.wave_value()
        self.image.set_alpha(alpha)
    else:
        self.image.set_alpha(255)

    self.rect = self.image.get_rect(midbottom = self.rect.midbottom)

def run_dust_animation(self):
    if self.status == 'run' and self.on_ground:
        self.dust_frame_index += self.dust_animation_speed
        if self.dust_frame_index >= len(self.dust_run_particles):
            self.dust_frame_index = 0

    dust_particle = self.dust_run_particles[int

```

```

def run_dust_animation(self):
    if self.status == 'run' and self.on_ground:
        self.dust_frame_index += self.dust_animation_speed
        if self.dust_frame_index >= len(self.dust_run_particles):
            self.dust_frame_index = 0

    dust_particle = self.dust_run_particles[int(self.dust_frame_index)]

    if self.facing_right:
        pos = self.rect.bottomleft - pygame.math.Vector2(6,10)
        self.display_surface.blit(dust_particle,pos)
    else:
        pos = self.rect.bottomright - pygame.math.Vector2(6,10)
        flipped_dust_particle = pygame.transform.flip(dust_particle,True,False)
        self.display_surface.blit(flipped_dust_particle,pos)

def get_input(self):
    keys = pygame.key.get_pressed()

    if keys[pygame.K_RIGHT]:
        self.direction.x = 1
        self.facing_right = True
    elif keys[pygame.K_LEFT]:
        self.direction.x = -1
        self.facing_right = False
    else:
        self.direction.x = 0

    if keys[pygame.K_SPACE] and self.on_ground:
        self.jump()
        self.create_jump_particles(self.rect.midbottom)

def get_status(self):
    if self.direction.y < 0:
        self.status = 'jump'
    elif self.direction.y > 1:
        self.status = 'fall'
    else:
        if self.direction.x != 0:
            self.status = 'run'
        else:
            self.status = 'idle'

def apply_gravity(self):
    self.direction.y += self.gravity
    self.collision_rect.y += self.direction.y

def jump(self):
    self.direction.y

```

3. Class Particle

```
class ParticleEffect(pygame.sprite.Sprite):
    def __init__(self, pos, type):
        super().__init__()
        self.frame_index = 0
        self.animation_speed = 0.5
        if type == 'jump':
            self.frames = import_folder('../graphics/character/dust_particles/jump')
        if type == 'land':
            self.frames = import_folder('../graphics/character/dust_particles/land')
        if type == 'explosion':
            self.frames = import_folder('../graphics/enemy/explosion')
        self.image = self.frames[self.frame_index]
        self.rect = self.image.get_rect(center = pos)

    def animate(self):
        self.frame_index += self.animation_speed
        if self.frame_index >= len(self.frames):
            self.kill()
        else:
            self.image = self.frames[int(self.frame_index)]

    def update(self, x_shift):
        self.animate()
        self.rect.x += x_shift
```

4. Class Node

```
class Node(pygame.sprite.Sprite):
    def __init__(self, pos, status, icon_speed, path):
        super().__init__()
        self.frames = import_folder(path)
        self.frame_index = 0
        self.image = self.frames[self.frame_index]
        if status == 'available':
            self.status = 'available'
        else:
            self.status = 'locked'
        self.rect = self.image.get_rect(center = pos)

        self.detection_zone = pygame.Rect(self.rect.centerx-(icon_speed/2),
                                           self.rect.centery-(icon_speed/2), icon_speed, icon_speed)

    def animate(self):
        self.frame_index += 0.15
        if self.frame_index >= len(self.frames):
            self.frame_index = 0
        self.image = self.frames[int(self.frame_index)]

    def update(self):
        if self.status == 'available':
            self.animate()
        else:
            tint_surf = self.image.copy()
            tint_surf.fill('black', None, pygame.BLEND_RGBA_MULT)
            self.image.blit(tint_surf, (0,0))
```

5. Class Icon

```
class Icon(pygame.sprite.Sprite):
    def __init__(self, pos):
        super().__init__()
        self.pos = pos
        self.image = pygame.image.load('../graphics/overworld/hat.png').convert_alpha()
        self.rect = self.image.get_rect(center = pos)

    def update(self):
        self.rect.center = self.pos
```

6. Class Overworld

```
class Overworld:
    def __init__(self, start_level, max_level, surface, create_level):

        # setup
        self.display_surface = surface
        self.max_level = max_level
        self.current_level = start_level
        self.create_level = create_level

        # movement logic
        self.moving = False
        self.move_direction = pygame.math.Vector2(0,0)
        self.speed = 8

        # sprites
        self.setup_nodes()
        self.setup_icon()
        self.sky = Sky(8, 'overworld')

        # time
        self.start_time = pygame.time.get_ticks()
        self.allow_input = False
        self.timer_length = 300

    def setup_nodes(self):
        self.nodes = pygame.sprite.Group()

        for index, node_data in enumerate(levels.values()):
            if index <= self.max_level:
                node_sprite = Node(node_data['node_pos'], 'available', self.speed, node_data['node_graphics'])
            else:
                node_sprite = Node(node_data['node_pos'], 'locked', self.speed, node_data['node_graphics'])
            self.nodes.add(node_sprite)

    def draw_paths(self):
        if self.max_level > 0:
            points = [node['node_pos'] for index, node in enumerate(levels.values()) if index <= self.max_level]
            pygame.draw.lines(self.display_surface, '#a04f45', False, points, 6)

    def setup_icon(self):
        self.icon = pygame.sprite.GroupSingle()
        icon_sprite = Icon(self.nodes.sprites()[self.current_level].rect.center)
        self.icon.add(icon_sprite)
```

```

def input(self):
    keys = pygame.key.get_pressed()

    if not self.moving and self.allow_input:
        if keys[pygame.K_RIGHT] and self.current_level < self.max_level:
            self.move_direction = self.get_movement_data('next')
            self.current_level += 1
            self.moving = True
        elif keys[pygame.K_LEFT] and self.current_level > 0:
            self.move_direction = self.get_movement_data('previous')
            self.current_level -= 1
            self.moving = True
        elif keys[pygame.K_SPACE]:
            self.create_level(self.current_level)

    def get_movement_data(self,target):
        start = pygame.math.Vector2(self.nodes.sprites()[self.current_level].rect.center)

        if target == 'next':
            end = pygame.math.Vector2(self.nodes.sprites()[self.current_level + 1].rect.center)
        else:
            end = pygame.math.Vector2(self.nodes.sprites()[self.current_level - 1].rect.center)

        return (end - start).normalize()

    def update_icon_pos(self):
        if self.moving and self.move_direction:
            self.icon.sprite.pos += self.move_direction * self.speed
            target_node = self.nodes.sprites()[self.current_level]
            if target_node.detection_zone.collidepoint(self.icon.sprite.pos):
                self.moving = False
                self.move_direction = pygame.math.Vector2(0,0)

    def input_timer(self):
        if not self.allow_input:
            current_time = pygame.time.get_ticks()
            if current_time - self.start_time >= self.timer_length:
                self.allow_input = True

    def run(self):
        self.input_timer()
        self.input()
        self.update_icon_pos()
        self.icon.update()
        self.nodes.update()

        self.sky.draw(self.display_surface)
        self.draw_paths()
        self.nodes.draw(self.display_surface)
        self.icon.draw(self.display_surface)

```

6.1 Contoh Objek Class Overworld dari Class Sky

```
self.sky = Sky(8,'overworld')
```

7. Class Level

```
class Level:
    def __init__(self,current_level,surface,create_overworld,change_coins,change_health):
        # general setup
        self.display_surface = surface
        self.world_shift = 0
        self.current_x = None

        # audio
        self.coin_sound = pygame.mixer.Sound('../audio/effects/coin.wav')
        self.stomp_sound = pygame.mixer.Sound('../audio/effects/stomp.wav')

        # overworld connection
        self.create_overworld = create_overworld
        self.current_level = current_level
        level_data = levels[self.current_level]
        self.new_max_level = level_data['unlock']

        # player
        player_layout = import_csv_layout(level_data['player'])
        self.player = pygame.sprite.GroupSingle()
        self.goal = pygame.sprite.GroupSingle()
        self.player_setup(player_layout,change_health)

        # user interface
        self.change_coins = change_coins

        # dust
        self.dust_sprite = pygame.sprite.GroupSingle()
        self.player_on_ground = False

        # explosion particles
        self.explosion_sprites = pygame.sprite.Group()

        # terrain setup
        terrain_layout = import_csv_layout(level_data['terrain'])
        self.terrain_sprites = self.create_tile_group(terrain_layout,'terrain')

        # grass setup
        grass_layout = import_csv_layout(level_data['grass'])
        self.grass_sprites = self.create_tile_group(grass_layout,'grass')

        # crates
        crate_layout = import_csv_layout(level_data['crates'])
        self.crate_sprites = self.create_tile_group(crate_layout,'crates')

        # coins
        coin_layout = import_csv_layout(level_data['coins'])
        self.coin_sprites = self.create_tile_group(coin_layout,'coins')

        # foreground palms
        fg_palm_layout = import_csv_layout(level_data['fg palms'])
        self.fg_palm_sprites = self.create_tile_group(fg_palm_layout,'fg palms')

        # background palms
        bg_palm_layout = import_csv_layout(level_data['bg palms'])
        self.bg_palm_sprites = self.create_tile_group(bg_palm_layout,'bg palms')
```

```

# background palms
bg_palm_layout = import_csv_layout(level_data['bg palms'])
self.bg_palm_sprites = self.create_tile_group(bg_palm_layout,'bg palms')

# enemy
enemy_layout = import_csv_layout(level_data['enemies'])
self.enemy_sprites = self.create_tile_group(enemy_layout,'enemies')

# constraint
constraint_layout = import_csv_layout(level_data['constraints'])
self.constraint_sprites = self.create_tile_group(constraint_layout,'constraint')

# decoration
self.sky = Sky(8)
level_width = len(terrain_layout[0]) * tile_size
self.water = Water(screen_height - 20,level_width)
self.clouds = Clouds(400,level_width,30)

def create_tile_group(self,layout,type):
    sprite_group = pygame.sprite.Group()

    for row_index, row in enumerate(layout):
        for col_index,val in enumerate(row):
            if val != '-1':
                x = col_index * tile_size
                y = row_index * tile_size

                if type == 'terrain':
                    terrain_tile_list = import_cut_graphics('../graphics/terrain/terrain_tiles.png')
                    tile_surface = terrain_tile_list[int(val)]
                    sprite = StaticTile(tile_size,x,y,tile_surface)

                if type == 'grass':
                    grass_tile_list = import_cut_graphics('../graphics/decoration/grass/grass.png')
                    tile_surface = grass_tile_list[int(val)]
                    sprite = StaticTile(tile_size,x,y,tile_surface)

                if type == 'crates':
                    sprite = Crate(tile_size,x,y)

                if type == 'coins':
                    if val == '0': sprite = Coin(tile_size,x,y,'../graphics/coins/gold',5)
                    if val == '1': sprite = Coin(tile_size,x,y,'../graphics/coins/silver',1)

                if type == 'fg palms':
                    if val == '0': sprite = Palm(tile_size,x,y,'../graphics/terrain/palm_small',38)
                    if val == '1': sprite = Palm(tile_size,x,y,'../graphics/terrain/palm_large',64)

                if type == 'bg palms':
                    sprite = Palm(tile_size,x,y,'../graphics/terrain/palm_bg',64)

                if type == 'enemies':
                    sprite = Enemy(tile_size,x,y)

                if type == 'constraint':
                    sprite = Tile(tile_size,x,y)

            sprite_group.add(sprite)

    return sprite_group

```

```

def player_setup(self,layout,change_health):
    for row_index, row in enumerate(layout):
        for col_index, val in enumerate(row):
            x = col_index * tile_size
            y = row_index * tile_size
            if val == '0':
                sprite = Player((x,y),self.display_surface,self.create_jump_particles,change_health)
                self.player.add(sprite)
            if val == '1':
                hat_surface = pygame.image.load('../graphics/character/hat.png').convert_alpha()
                sprite = StaticTile(tile_size,x,y,hat_surface)
                self.goal.add(sprite)

def enemy_collision_reverse(self):
    for enemy in self.enemy_sprites.sprites():
        if pygame.sprite.spritecollide(enemy,self.constraint_sprites,False):
            enemy.reverse()

def create_jump_particles(self,pos):
    if self.player.sprite.facing_right:
        pos -= pygame.math.Vector2(10,5)
    else:
        pos += pygame.math.Vector2(10,-5)
    jump_particle_sprite = ParticleEffect(pos,'jump')
    self.dust_sprite.add(jump_particle_sprite)

def horizontal_movement_collision(self):
    player = self.player.sprite
    player.collision_rect.x += player.direction.x * player.speed
    collidable_sprites = self.terrain_sprites.sprites() + self.create_sprites.sprites() + self_fg_palm_sprites.sprites()
    for sprite in collidable_sprites:
        if sprite.rect.colliderect(player.collision_rect):
            if player.direction.x < 0:
                player.collision_rect.left = sprite.rect.right
                player.on_left = True
                self.current_x = player.rect.left
            elif player.direction.x > 0:
                player.collision_rect.right = sprite.rect.left
                player.on_right = True
                self.current_x = player.rect.right

def vertical_movement_collision(self):
    player = self.player.sprite
    player.apply_gravity()
    collidable_sprites = self.terrain_sprites.sprites() + self.create_sprites.sprites() + self_fg_palm_sprites.sprites()

    for sprite in collidable_sprites:
        if sprite.rect.colliderect(player.collision_rect):
            if player.direction.y > 0:
                player.collision_rect.bottom = sprite.rect.top
                player.direction.y = 0
                player.on_ground = True
            elif player.direction.y < 0:
                player.collision_rect.top = sprite.rect.bottom
                player.direction.y = 0
                player.on_ceiling = True

    if player.on_ground and player.direction.y < 0 or player.direction.y > 1:
        player.on_ground = False

```

```

def scroll_x(self):
    player = self.player.sprite
    player_x = player.rect.centerx
    direction_x = player.direction.x

    if player_x < screen_width / 4 and direction_x < 0:
        self.world_shift = 8
        player.speed = 0
    elif player_x > screen_width - (screen_width / 4) and direction_x > 0:
        self.world_shift = -8
        player.speed = 0
    else:
        self.world_shift = 0
        player.speed = 8

def get_player_on_ground(self):
    if self.player.sprite.on_ground:
        self.player_on_ground = True
    else:
        self.player_on_ground = False

def create_landing_dust(self):
    if not self.player_on_ground and self.player.sprite.on_ground and not self.dust_sprite.sprites():
        if self.player.sprite.facing_right:
            offset = pygame.math.Vector2(10,15)
        else:
            offset = pygame.math.Vector2(-10,15)
        fall_dust_particle = ParticleEffect(self.player.sprite.rect.midbottom - offset, 'land')
        self.dust_sprite.add(fall_dust_particle)

def check_death(self):
    if self.player.sprite.rect.top > screen_height:
        self.create_overworld(self.current_level,0)

def check_win(self):
    if pygame.sprite.spritecollide(self.player.sprite,self.goal,False):
        self.create_overworld(self.current_level,self.new_max_level)

def check_coin_collisions(self):
    collided_coins = pygame.sprite.spritecollide(self.player.sprite,self.coin_sprites,True)
    if collided_coins:
        self.coin_sound.play()
        for coin in collided_coins:
            self.change_coins(coin.value)

def check_enemy_collisions(self):
    enemy_collisions = pygame.sprite.spritecollide(self.player.sprite,self.enemy_sprites,False)

    if enemy_collisions:
        for enemy in enemy_collisions:
            enemy_center = enemy.rect.centery
            enemy_top = enemy.rect.top
            player_bottom = self.player.sprite.rect.bottom
            if enemy_top < player_bottom < enemy_center and self.player.sprite.direction.y >= 0:
                self.stomp_sound.play()
                self.player.sprite.direction.y = -15
                explosion_sprite = ParticleEffect(enemy.rect.center,'explosion')
                self.explosion_sprites.add(explosion_sprite)
                enemy.kill()
            else:

```

7.1 Contoh Objek Class Level dari Class Sky, Class Water, Class Clouds

```
self.sky = Sky(8)
level_width = len(terrain_layout[0]) * tile_size
self.water = Water(screen_height - 20, level_width)
self.clouds = Clouds(400, level_width, 30)
```

7.2 Contoh Objek Pada Class Level dari Class Palm

```
sprite = Palm(tile_size,x,y,'../graphics/terrain/palm_bg',64)
```

7.3 Contoh Objek Pada Class Level dari Class Palm

```
sprite = Enemy(tile_size,x,y)
```

7.4 Contoh Objek Pada Class Level dari Class Palm

```
sprite = Tile(tile_size,x,y)
```

8. Class Enemy

```
class Enemy(AnimatedTile):
    def __init__(self,size,x,y):
        super().__init__(size,x,y,'../graphics/enemy/run')
        self.rect.y += size - self.image.get_size()[1]
        self.speed = randint(3,5)

    def move(self):
        self.rect.x += self.speed

    def reverse_image(self):
        if self.speed > 0:
            self.image = pygame.transform.flip(self.image,True,False)

    def reverse(self):
        self.speed *= -1

    def update(self,shift):
        self.rect.x += shift
        self.animate()
        self.move()
        self.reverse_image()
```

9. Class Sky

```
class Sky:
    def __init__(self,horizon,style = 'level'):
        self.top = pygame.image.load('../graphics/decoration/sky/sky_top.png').convert()
        self.bottom = pygame.image.load('../graphics/decoration/sky/sky_bottom.png').convert()
        self.middle = pygame.image.load('../graphics/decoration/sky/sky_middle.png').convert()
        self.horizon = horizon

        # stretch
        self.top = pygame.transform.scale(self.top,(screen_width,tile_size))
        self.bottom = pygame.transform.scale(self.bottom,(screen_width,tile_size))
        self.middle = pygame.transform.scale(self.middle,(screen_width,tile_size))

    self.style = style
    if self.style == 'overworld':
        palm_surfaces = import_folder('../graphics/overworld/palms')
        self.palms = []

        for surface in [choice(palm_surfaces) for image in range(10)]:
            x = randint(0,screen_width)
            y = (self.horizon * tile_size) + randint(50,100)
            rect = surface.get_rect(midbottom = (x,y))
            self.palms.append((surface,rect))

        cloud_surfaces = import_folder('../graphics/overworld/clouds')
        self.clouds = []

        for surface in [choice(cloud_surfaces) for image in range(10)]:
            x = randint(0,screen_width)
            y = randint(0,(self.horizon * tile_size) - 100)
            rect = surface.get_rect(midbottom = (x,y))
            self.clouds.append((surface,rect))

    def draw(surface):
        for row in range(vertical_tile_number):
            y = row * tile_size
            if row < self.horizon:
                surface.blit(self.top,(0,y))
            elif row == self.horizon:
                surface.blit(self.middle,(0,y))
            else:
                surface.blit(self.bottom,(0,y))

        if self.style == 'overworld':
            for palm in self.palms:
                surface.blit(palm[0],palm[1])
            for cloud in self.clouds:
                surface.blit(cloud[0],cloud[1])
```

10. Class Water

```
class Water:  
    def __init__(self,top,level_width):  
        water_start = -screen_width  
        water_tile_width = 192  
        tile_x_amount = int((level_width + screen_width * 2) / water_tile_width)  
        self.water_sprites = pygame.sprite.Group()  
  
        for tile in range(tile_x_amount):  
            x = tile * water_tile_width + water_start  
            y = top  
            sprite = AnimatedTile(192,x,y,'../graphics/decoration/water')  
            self.water_sprites.add(sprite)  
  
    def draw(self,surface,shift):  
        self.water_sprites.update(shift)  
        self.water_sprites.draw(surface)
```

11. Class Clouds

```
class Clouds:  
    def __init__(self,horizon,level_width,cloud_number):  
        cloud_surf_list = import_folder('../graphics/decoration/clouds')  
        min_x = -screen_width  
        max_x = level_width + screen_width  
        min_y = 0  
        max_y = horizon  
        self.cloud_sprites = pygame.sprite.Group()  
  
        for cloud in range(cloud_number):  
            cloud = choice(cloud_surf_list)  
            x = randint(min_x,max_x)  
            y = randint(min_y,max_y)  
            sprite = StaticTile(0,x,y,cloud)  
            self.cloud_sprites.add(sprite)  
  
    def draw(self,surface,shift):  
        self.cloud_sprites.update(shift)  
        self.cloud_sprites.draw(surface)
```

12. Class UI

```
class UI:
    def __init__(self,surface):
        # setup
        self.display_surface = surface
        self._current_health = 0 # private
        self._full_health = 0 # private

        # health
        self.health_bar = pygame.image.load('../graphics/ui/health_bar.png').convert_alpha()
        self.health_bar_topleft = (54,39)
        self.bar_max_width = 152
        self.bar_height = 4

        # coins
        self.coin = pygame.image.load('../graphics/ui/coin.png').convert_alpha()
        self.coin_rect = self.coin.get_rect(topleft = (50,61))
        self.font = pygame.font.Font('../graphics/ui/ARCADEPI.ttf',30)

    def get_health(self):
        return self._current_health, self._full_health

    def set_health(self, current, full):
        self._current_health = current
        self._full_health = full

    def show_health(self,current,full):
        self.set_health(current, full)
        self.display_surface.blit(self.health_bar,(20,10))
        current_health_ratio = self._current_health / self._full_health
        current_bar_width = self.bar_max_width * current_health_ratio
        health_bar_rect = pygame.Rect(self.health_bar_topleft,(current_bar_width,self.bar_height))
        pygame.draw.rect(self.display_surface,'#dc4949',health_bar_rect)

    def show_coins(self,amount):
        self.display_surface.blit(self.coin,self.coin_rect)
        coin_amount_surf = self.font.render(str(amount),False,'#33323d')
        coin_amount_rect = coin_amount_surf.get_rect(midleft = (self.coin_rect.right + 4,self.coin_rect.centery))
        self.display_surface.blit(coin_amount_surf,coin_amount_rect)
```

12.1 Contoh Enkapsulasi Pada Class UI

```
def get_health(self):
    return self._current_health, self._full_health
```

13. Class Tile

```
class Tile(pygame.sprite.Sprite, TileInterface):
    def __init__(self, size, x, y):
        super().__init__()
        self.image = pygame.Surface((size, size))
        self.rect = self.image.get_rect(topleft = (x, y))

    def update(self, shift):
        self.rect.x += shift

    def get_tile_type(self):
        return "base_tile"

    def get_collision_type(self):
        return "solid"
```

13.1 Contoh Polimorfisme Pada Class Tile

```
def get_collision_type(self):
    return "solid"

def update(self, shift):
    self.rect.x += shift

def get_tile_type(self):
    return "base_tile"
```

14. Class StaticTile

```
class StaticTile(Tile):
    def __init__(self, size, x, y, surface):
        super().__init__(size, x, y)
        self.image = surface

    def get_tile_type(self):
        return "static"

    def get_collision_type(self):
        return "solid"
```

14.1 Contoh Polimorfisme Pada Class StaticTile

```
def get_collision_type(self):
    return "solid"
```

```
def get_tile_type(self):
    return "static"
```

15. Class Crate

```
class Crate(StaticTile):
    def __init__(self, size, x, y):
        super().__init__(size, x, y, pygame.image.load('../graphics/terrain/crate.png').convert_alpha())
        offset_y = y + size
        self.rect = self.image.get_rect(bottomleft = (x, offset_y))

    def get_tile_type(self):
        return "crate"

    def get_collision_type(self):
        return "solid"
```

15.1 Contoh Polimorfisme Pada Class Crate

```
def get_collision_type(self):
    return "solid"

def get_tile_type(self):
    return "crate"
```

16. Class AnimatedTile

```
class AnimatedTile(Tile):
    def __init__(self, size, x, y, path):
        super().__init__(size, x, y)
        self.frames = import_folder(path)
        self.frame_index = 0
        self.image = self.frames[self.frame_index]

    def animate(self):
        self.frame_index += 0.15
        if self.frame_index >= len(self.frames):
            self.frame_index = 0
        self.image = self.frames[int(self.frame_index)]

    def update(self, shift):
        self.animate()
        self.rect.x += shift

    def get_tile_type(self):
        return "animated"

    def get_collision_type(self):
        return "none"
```

16.1 Contoh Polimorfisme Pada Class AnimatedTile

```
def get_collision_type(self):
    return "none"

def get_tile_type(self):
    return "animated"
```

17. Class TileInterface

17.1 Contoh Abstrak Pada Class TileInterface

```
class TileInterface(ABC):
    @abstractmethod
    def get_tile_type(self):
        pass

    @abstractmethod
    def get_collision_type(self):
        pass
```

18. Class Coin dan Class Palm

```
class Coin(AnimatedTile):
    def __init__(self, size, x, y, path, value):
        super().__init__(size, x, y, path)
        center_x = x + int(size / 2)
        center_y = y + int(size / 2)
        self.rect = self.image.get_rect(center = (center_x, center_y))
        self._value = value # mengubah value menjadi private dengan _

    # getter
    def get_value(self):
        return self._value

    # setter
    def set_value(self, new_value):
        self._value = new_value

    @property
    def value(self):
        return self._value

    @value.setter
    def value(self, new_value):
        self._value = new_value

    def get_tile_type(self):
        return "coin"

    def get_collision_type(self):
        return "collectible"

class Palm(AnimatedTile):
    def __init__(self, size, x, y, path, offset):
        super().__init__(size, x, y, path)
        offset_y = y - offset
        self.rect.topleft = (x, offset_y)

    def get_tile_type(self):
        return "palm"

    def get_collision_type(self):
        return "decoration"
```

18.1 Contoh Enkapsulasi Pada Class UI

```
# getter
def get_value(self):
    return self._value

# setter
def set_value(self, new_value):
    self._value = new_value

@property
def value(self):
    return self._value

@value.setter
def value(self, new_value):
    self._value = new_value
```

18.2 Contoh Polimorfisme Pada Class Coin dan Class Palm

Class Coin :

```
def get_collision_type(self):
    return "collectible"

def get_tile_type(self):
    return "coin"
```

Class Palm :

```
def get_collision_type(self):
    return "decoration"

def get_tile_type(self):
    return "palm"
```

19. Support

```
def import_folder(path):
    surface_list = []

    for _, __, image_files in walk(path):
        for image in image_files:
            full_path = path + '/' + image
            image_surf = pygame.image.load(full_path).convert_alpha()
            surface_list.append(image_surf)

    return surface_list
```

```

def import_csv_layout(path):
    terrain_map = []
    with open(path) as map:
        level = reader(map, delimiter = ',')
        for row in level:
            terrain_map.append(list(row))
    return terrain_map

def import_cut_graphics(path):
    surface = pygame.image.load(path).convert_alpha()
    tile_num_x = int(surface.get_size()[0] / tile_size)
    tile_num_y = int(surface.get_size()[1] / tile_size)

    cut_tiles = []
    for row in range(tile_num_y):
        for col in range(tile_num_x):
            x = col * tile_size
            y = row * tile_size
            new_surf = pygame.Surface((tile_size,tile_size), flags = pygame.SRCALPHA)
            new_surf.blit(surface,(0,0),pygame.Rect(x,y,tile_size,tile_size))
            cut_tiles.append(new_surf)

    return cut_tiles

```

20. Settings

```

vertical_tile_number = 11
tile_size = 64

screen_height = vertical_tile_number * tile_size
screen_width = 1200

```

21. Game_Data

```

level_0 = {
    'terrain': '../levels/0/level_0_terrain.csv',
    'coins':'../levels/0/level_0_coins.csv',
    'fg palms':'../levels/0/level_0_fg_palms.csv',
    'bg palms':'../levels/0/level_0_bg_palms.csv',
    'crates': '../levels/0/level_0_crates.csv',
    'enemies':'../levels/0/level_0_enemies.csv',
    'constraints':'../levels/0/level_0_constraints.csv',
    'player': '../levels/0/level_0_player.csv',
    'grass': '../levels/0/level_0_grass.csv',
    'node_pos': (110,400),
    'unlock': 1,
    'node_graphics': '../graphics/overworld/0'}

```

```

level_1 = {
    'terrain': '../levels/1/level_1_terrain.csv',
    'coins':'../levels/1/level_1_coins.csv',
    'fg palms':'../levels/1/level_1_fg_palms.csv',
    'bg palms':'../levels/1/level_1_bg_palms.csv',
    'crates': '../levels/1/level_1_crates.csv',
    'enemies':'../levels/1/level_1_enemies.csv',
    'constraints':'../levels/1/level_1_constraints.csv',
    'player': '../levels/1/level_1_player.csv',
    'grass': '../levels/1/level_1_grass.csv',
    'node_pos': (300,220),
    'node_graphics': '../graphics/overworld/1',
    'unlock': 2}
level_2 = {
    'terrain': '../levels/2/level_2_terrain.csv',
    'coins':'../levels/2/level_2_coins.csv',
    'fg palms':'../levels/2/level_2_fg_palms.csv',
    'bg palms':'../levels/2/level_2_bg_palms.csv',
    'crates': '../levels/2/level_2_crates.csv',
    'enemies':'../levels/2/level_2_enemies.csv',
    'constraints':'../levels/2/level_2_constraints.csv',
    'player': '../levels/2/level_2_player.csv',
    'grass': '../levels/2/level_2_grass.csv',
    'node_pos': (480,610),
    'node_graphics': '../graphics/overworld/2',
    'unlock': 3}
level_3 = {
    'terrain': '../levels/2/level_2_terrain.csv',
    'coins':'../levels/2/level_2_coins.csv',
    'fg palms':'../levels/2/level_2_fg_palms.csv',
    'bg palms':'../levels/2/level_2_bg_palms.csv',
    'crates': '../levels/2/level_2_crates.csv',
    'enemies':'../levels/2/level_2_enemies.csv',
    'constraints':'../levels/2/level_2_constraints.csv',
    'player': '../levels/2/level_2_player.csv',
    'grass': '../levels/2/level_2_grass.csv',
    'node_pos': (610,350),
    'node_graphics': '../graphics/overworld/3',
    'unlock': 4}

```

```

level_4 = {
    'terrain': '../levels/2/level_2_terrain.csv',
    'coins':'../levels/2/level_2_coins.csv',
    'fg palms':'../levels/2/level_2_fg_palms.csv',
    'bg palms':'../levels/2/level_2_bg_palms.csv',
    'crates': '../levels/2/level_2_crates.csv',
    'enemies':'../levels/2/level_2_enemies.csv',
    'constraints':'../levels/2/level_2_constraints.csv',
    'player': '../levels/2/level_2_player.csv',
    'grass': '../levels/2/level_2_grass.csv',
    'node_pos': (880,210),
    'node_graphics': '../graphics/overworld/4',
    'unlock': 5}

level_5 = {
    'terrain': '../levels/2/level_2_terrain.csv',
    'coins':'../levels/2/level_2_coins.csv',
    'fg palms':'../levels/2/level_2_fg_palms.csv',
    'bg palms':'../levels/2/level_2_bg_palms.csv',
    'crates': '../levels/2/level_2_crates.csv',
    'enemies':'../levels/2/level_2_enemies.csv',
    'constraints':'../levels/2/level_2_constraints.csv',
    'player': '../levels/2/level_2_player.csv',
    'grass': '../levels/2/level_2_grass.csv',
    'node_pos': (1050,400),
    'node_graphics':'../graphics/overworld/5',
    'unlock': 5}

levels = {
    0: level_0,
    1: level_1,
    2: level_2,
    3: level_3,
    4: level_4,
    5: level_5}

```

Q. LAMPIRAN

Link GitHub : <https://tinyurl.com/GitHub-TubesPBO-08-RA>

Link Demo Game Youtube : https://youtu.be/_jfKiyt_g7A?si=vbjm18fG3rR6_Lal



Asset Karakter Juppa Berlari



Asset Karakter Juppa Meloncat



Asset Karakter Juppa Terjatuh



Asset Karakter Enemy Berlari



Asset Pohon Besar



Asset Pohon Kecil



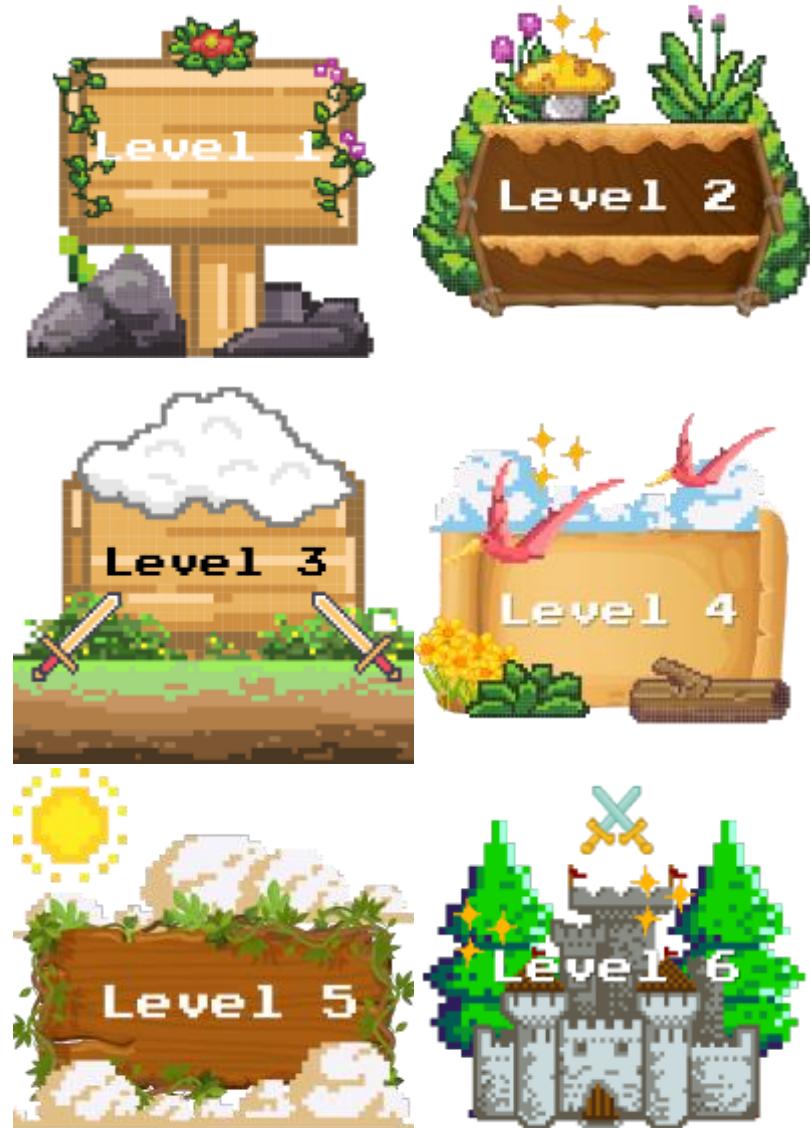
Asset Background Pohon



Asset Coin Emas



Asset Coin Silver



Asset Level 1, Level2, Level 3, Level 4, Level 5, Level 6



Asset Awan Halaman Utama



Asset Awan Halaman Map



Asset Rumput



Asset Sky (Langit)



Asset Water (Air)



Asset Crate





Asset Map Tiles