

PROJECT
PEMROGRAMAN KOMPUTER



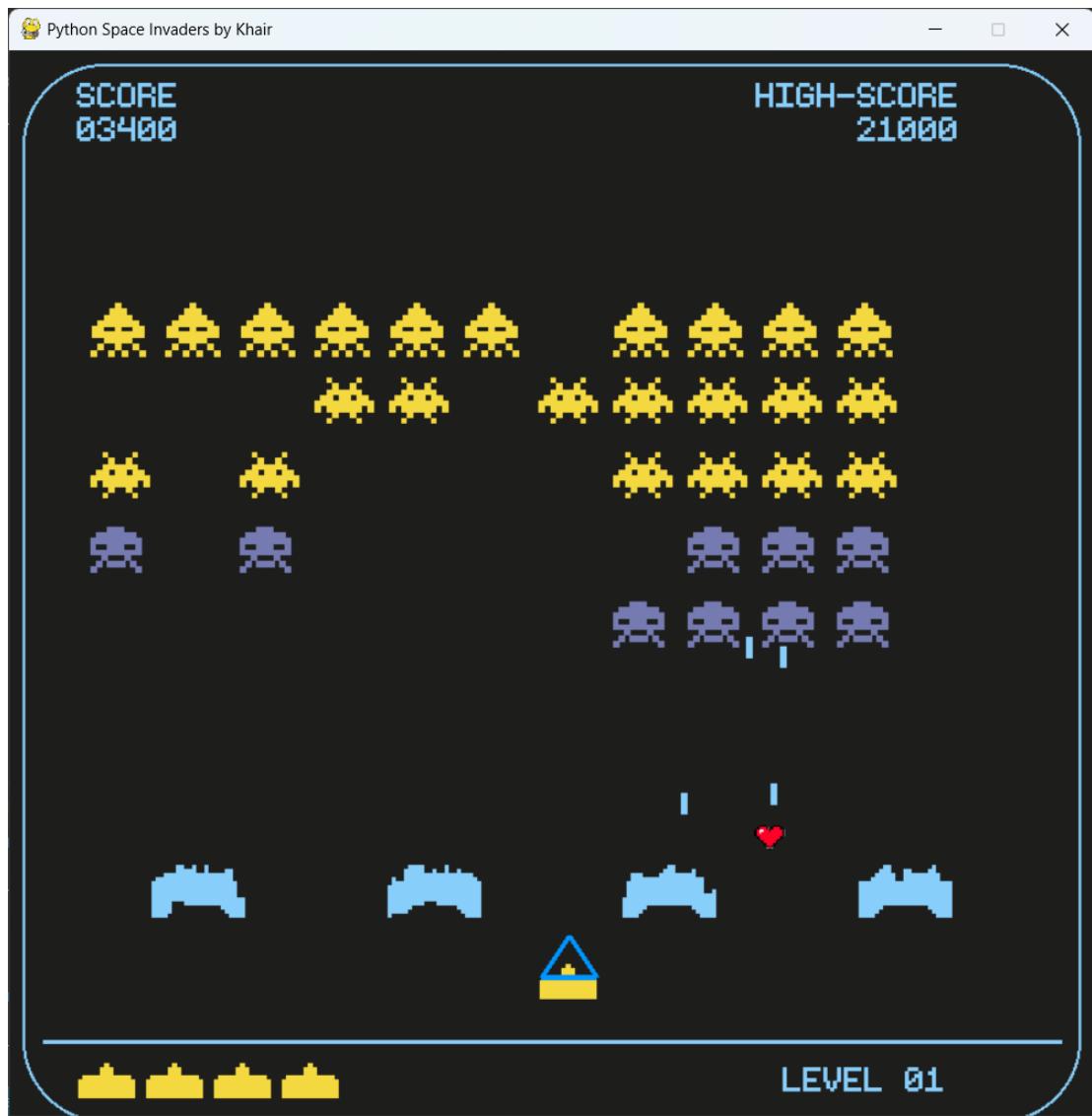
Nama : Muhammad Khair Syawaludin

NIM : G1D022037

PROGRAM STUDI MATEMATIKA
FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM
UNIVERSITAS MATARAM
2024

Python Space Invader

A. DESKRIPSI



Permainan ini adalah permainan video klasik yaitu "Space Invaders" yang dibuat menggunakan Python, dimana terlihat formasi musuh piksel berwarna kuning dan ungu di bagian atas sebagai alien, sebuah pesawat pemain berbentuk segitiga biru (shield/tameng) dengan dasar kuning di bagian bawah tengah, serta tiga penghalang berwarna biru (obstacle) di antara keduanya; laser tembakan terlihat di atas pesawat pemain, sementara informasi skor ("SCORE 03400"), skor tertinggi ("HIGH-SCORE 21000"), dan level ("LEVEL 01") ditampilkan di layar,

terdapat juga gambar spaceship berjumlah 4 yang mengindikasikan sisa nyawa pemain. Terakhir terdapat ikon hati dimana berguna untuk menambahkan nyawa jika pemain mengenai ikon tersebut.

B. SYNTAX DAN PENJELASAN

- File *settings.py*

```
SCREEN_WIDTH = 750
SCREEN_HEIGHT = 700
OFFSET = 50

GREY = (29, 29, 27)
COLOR = (135, 206, 250)

FONT_PATH = "Font/monogram.ttf"
FONT_SIZE = 40

MUSIC_PATH = "Sounds/music2.mp3"
LASER_SOUND_PATH = "Sounds/laser.ogg"
EXPLOSION_SOUND_PATH = "Sounds/explosion.ogg"
POWERUP_SOUND_PATH = "Sounds/powerup.wav"

SPACESHIP_IMAGE_PATH = "Graphics/spaceship.png"
ALIEN1_IMAGE_PATH = "Graphics/alien_1.png"
ALIEN2_IMAGE_PATH = "Graphics/alien_2.png"
ALIEN3_IMAGE_PATH = "Graphics/alien_3.png"
MYSTERY_SHIP_IMAGE_PATH = "Graphics/mystery.png"
EXTRA_LIFE_IMAGE_PATH = "Graphics/extra_life.png"
SHIELD_POWERUP_IMAGE_PATH = "Graphics/shield_powerup.png"
LASER_COLOR = (135, 206, 250)
LASER_WIDTH = 5
LASER_HEIGHT = 16
SPACESHIP_LASER_SPEED = 5
ALIEN_LASER_SPEED = -6

BLOCK_COLOR = (135, 206, 250)
BLOCK_SIZE = 3

ALIEN_START_X = 75
ALIEN_START_Y = 110
ALIEN_HORIZONTAL_SPACING = 55
```

```

ALIEN_VERTICAL_SPACING = 55
ALIEN_MOVE_DOWN_DISTANCE = 2

MYSTERY_SHIP_SPEED = 3

SHIELD_COLOR = (0, 150, 255)
SHIELD_THICKNESS = 4
SHIELD_DURATION = 5000 # milliseconds

SHIELD_V_OFFSET_TOP = 20
SHIELD_V_WIDTH = 40
SHIELD_V_HEIGHT = 30

HIGHSCORE_FILE = "highscore.txt"

ALIEN_SHOOT_DELAY = 300 # milliseconds
MYSTERY_SHIP_SPAWN_MIN = 8000 # milliseconds
MYSTERY_SHIP_SPAWN_MAX = 15000 # milliseconds
EXTRA_LIFE_SPAWN_MIN = 10000 # milliseconds
EXTRA_LIFE_SPAWN_MAX = 20000 # milliseconds
SHIELD_SPAWN_MIN = 5000 # milliseconds
SHIELD_SPAWN_MAX = 35000 # milliseconds

```

Sintaks diatas mendefinisikan konstanta yang akan digunakan dalam permainan.

Memisahkan konstanta juga membuat kode lebih mudah dibaca dan dimodifikasi.

Konstanta ini mencakup:

- Pengaturan Tampilan: Ukuran layar, warna, font, dan jalur aset media.
- Suara: Lokasi file suara untuk musik, laser, ledakan, dan power-up.
- Aset Grafis: Lokasi file gambar untuk pesawat luar angkasa, alien, kapal misteri, power-up, dll.
- Pengaturan Gameplay: Kecepatan, jarak, waktu, dan parameter lain yang mengontrol bagaimana permainan berjalan.
- File *alien.py*

```
import pygame, random
```

```
from settings import ALIEN1_IMAGE_PATH, ALIEN2_IMAGE_PATH,
ALIEN3_IMAGE_PATH, MYSTERY_SHIP_IMAGE_PATH, SCREEN_WIDTH, OFFSET,
MYSTERY_SHIP_SPEED

class Alien(pygame.sprite.Sprite):
    def __init__(self, type, x, y):
        super().__init__()
        self.type = type
        if type == 1:
            path = ALIEN1_IMAGE_PATH
        elif type == 2:
            path = ALIEN2_IMAGE_PATH
        elif type == 3:
            path = ALIEN3_IMAGE_PATH
        self.image = pygame.image.load(path)
        self.rect = self.image.get_rect(topleft = (x, y))

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

class MysteryShip(pygame.sprite.Sprite):
    def __init__(self, screen_width, offset):
        super().__init__()
        self.screen_width = screen_width
        self.offset = offset
        self.image = pygame.image.load(MYSTERY_SHIP_IMAGE_PATH)

        x = random.choice([self.offset/2, self.screen_width +
self.offset - self.image.get_width()])
        if x == self.offset/2:
            self.speed = MYSTERY_SHIP_SPEED
        else:
            self.speed = -MYSTERY_SHIP_SPEED

        self.rect = self.image.get_rect(topleft = (x, 90))

    def update(self):
        self.rect.x += self.speed
        if self.rect.right > self.screen_width + self.offset/2:
            self.kill()
        elif self.rect.left < self.offset/2:
            self.kill()
```

import pygame, random: Baris ini mengimpor library Pygame, yang menyediakan fungsionalitas inti untuk pengembangan permainan 2D, seperti manajemen grafis dan event, dan juga mengimpor modul random untuk menghasilkan angka acak yang akan digunakan dalam permainan.

```
from settings import ALIEN1_IMAGE_PATH,  
ALIEN2_IMAGE_PATH, ALIEN3_IMAGE_PATH,
```

```
MYSTERY_SHIP_IMAGE_PATH, SCREEN_WIDTH, OFFSET,
```

MYSTERY_SHIP_SPEED: Baris ini mengimpor variabel-variabel konfigurasi spesifik dari modul "settings", termasuk path gambar untuk berbagai jenis alien dan kapal misteri, lebar layar, offset, dan kecepatan kapal misteri. Ini memungkinkan konfigurasi yang mudah dan terpusat untuk permainan.

Kelas Alien: Kelas ini mendefinisikan objek alien dalam permainan. Konstruktornya (`__init__`) menerima tipe alien (1, 2, atau 3) dan posisi awal (x, y), kemudian memuat gambar alien yang sesuai dan membuat kotak pembatas di sekelilingnya. Fungsi update bertanggung jawab untuk menggerakkan alien secara horizontal berdasarkan arah yang diberikan.

Kelas MysteryShip: Kelas ini merepresentasikan kapal misteri yang muncul secara acak dalam permainan. Konstruktornya (`__init__`) mengambil lebar layar dan offset untuk menentukan posisi awal kapal di kiri atau kanan layar, serta menentukan arah geraknya. Fungsi update membuat kapal bergerak melintasi layar dan menghapusnya ketika kapal tersebut keluar dari layar.

Fungsi `__init__` pada Kelas Alien: Fungsi ini adalah konstruktor untuk objek alien. Ia menerima tipe alien dan koordinat awal, memilih gambar yang sesuai berdasarkan tipe alien, dan menginisialisasi sprite dengan gambar tersebut pada posisi yang ditentukan. Ini juga membuat kotak pembatas yang akan digunakan untuk deteksi tabrakan.

Fungsi update pada Kelas Alien: Fungsi ini bertugas untuk memperbarui posisi horizontal objek alien. Dengan menerima parameter arah, alien

digerakkan ke kiri atau kanan di layar. Ini adalah fungsi yang akan dipanggil setiap frame untuk membuat alien terlihat bergerak.

Fungsi `__init__` pada Kelas `MysteryShip`: Fungsi ini adalah konstruktor untuk objek kapal misteri. Fungsi ini menginisialisasi posisi awal kapal di kiri atau kanan layar, secara acak. Fungsi ini juga memuat gambar kapal misteri, mengatur posisi dan kecepatan awal, serta membuat kotak pembatas untuk deteksi tabrakan.

Fungsi `update` pada Kelas `MysteryShip`: Fungsi ini bertugas untuk memperbarui posisi kapal misteri, membuatnya bergerak secara horizontal melintasi layar. Jika kapal telah sepenuhnya keluar dari layar (baik di sisi kiri maupun kanan), maka kapal tersebut akan dihapus.

- File `game.py`

```
import pygame, random
from spaceship import Spaceship
from obstacle import Obstacle, grid
from alien import Alien, MysteryShip
from laser import Laser
from powerup import ExtraLife, Shield
from settings import SCREEN_WIDTH, SCREEN_HEIGHT, OFFSET,
ALIEN_MOVE_DOWN_DISTANCE, EXPLOSION_SOUND_PATH, MUSIC_PATH,
HIGHSCORE_FILE, POWERUP_SOUND_PATH, ALIEN_START_X, ALIEN_START_Y,
ALIEN_HORIZONTAL_SPACING, ALIEN_VERTICAL_SPACING

class Game:
    def __init__(self, screen_width, screen_height, offset):
        self.screen_width = screen_width
        self.screen_height = screen_height
        self.offset = offset
        self.spaceship_group = pygame.sprite.GroupSingle()
        self.spaceship_group.add(Spaceship(self.screen_width,
self.screen_height, self.offset))
        self.obstacles = self.create_obstacles()
        self.aliens_group = pygame.sprite.Group()
        self.alien_start_x = ALIEN_START_X
        self.alien_start_y = ALIEN_START_Y
        self.alien_horizontal_spacing = ALIEN_HORIZONTAL_SPACING
```

```
    self.alien_vertical_spacing = ALIEN_VERTICAL_SPACING
    self.create_aliens()
    self.aliens_direction = 1
    self.alien_lasers_group = pygame.sprite.Group()
    self.mystery_ship_group = pygame.sprite.GroupSingle()
    self.extra_life_group = pygame.sprite.Group()
    self.shield_group = pygame.sprite.Group()
    self.lives = 3
    self.run = True
    self.score = 0
    self.highscore = 0
    self.explosion_sound = pygame.mixer.Sound(EXPLOSION_SOUND_PATH)
    self.powerup_sound = pygame.mixer.Sound(POWERUP_SOUND_PATH)
    self.load_highscore()
    pygame.mixer.music.load(MUSIC_PATH)
    pygame.mixer.music.play(-1)
    self.shield_active = False
    self.shield_timer = 0
    from settings import SHIELD_DURATION
    self.shield_duration = SHIELD_DURATION
    self.level = 1

    def create_obstacles(self):
        from obstacle import grid
        obstacle_width = len(grid[0]) * 3
        gap = (self.screen_width + self.offset - (4 *
obstacle_width))/5
        obstacles = []
        for i in range(4):
            offset_x = (i + 1) * gap + i * obstacle_width
            obstacle = Obstacle(offset_x, self.screen_height - 100)
            obstacles.append(obstacle)
        return obstacles

    def create_aliens(self):
        for row in range(5):
            for column in range(11):
                x = self.alien_start_x + column *
self.alien_horizontal_spacing
                y = self.alien_start_y + row *
self.alien_vertical_spacing

                if row == 0:
```

```

        alien_type = 3
    elif row in (1,2):
        alien_type = 2
    else:
        alien_type = 1

    alien = Alien(alien_type, x + self.offset/2, y)
    self.aliens_group.add(alien)

def move_aliens(self):
    self.aliens_group.update(self.aliens_direction)

    alien_sprites = self.aliens_group.sprites()
    for alien in alien_sprites:
        if alien.rect.right >= self.screen_width + self.offset/2:
            self.aliens_direction = -1
            self.alien_move_down(ALIEN_MOVE_DOWN_DISTANCE)
        elif alien.rect.left <= self.offset/2:
            self.aliens_direction = 1
            self.alien_move_down(ALIEN_MOVE_DOWN_DISTANCE)

def alien_move_down(self, distance):
    if self.aliens_group:
        for alien in self.aliens_group.sprites():
            alien.rect.y += distance

        for alien in self.aliens_group:
            if alien.rect.bottom >=
self.spaceship_group.sprite.rect.top:
                self.game_over()
                break

def alien_shoot_laser(self):
    if self.aliens_group.sprites():
        random_alien = random.choice(self.aliens_group.sprites())
        from settings import ALIEN LASER SPEED
        laser_sprite = Laser(random_alien.rect.center,
ALIEN LASER SPEED, self.screen_height)
        self.alien_lasers_group.add(laser_sprite)

def create_mystery_ship(self):
    self.mystery_ship_group.add(MysteryShip(self.screen_width,
self.offset))

```



```
        self.explosion_sound.play()
        laser_sprite.kill()

        for obstacle in self.obstacles:
            if pygame.sprite.spritecollide(laser_sprite,
obstacle.blocks_group, True):
                laser_sprite.kill()

    if self.aliens_group:
        for alien in self.aliens_group:
            for obstacle in self.obstacles:
                pygame.sprite.spritecollide(alien,
obstacle.blocks_group, True)

                if pygame.sprite.spritecollide(alien,
self.spaceship_group, False):
                    self.game_over()

    if self.extra_life_group:
        for extra_life in
pygame.sprite.spritecollide(self.spaceship_group.sprite,
self.extra_life_group, True):
            self.lives += 1
            self.powerup_sound.play()

    if self.shield_group:
        for shield in
pygame.sprite.spritecollide(self.spaceship_group.sprite,
self.shield_group, True):
            self.shield_active = True
            self.shield_timer = pygame.time.get_ticks()
            self.powerup_sound.play()

    if not self.aliens_group.sprites() and self.run:
        self.level_complete()

def level_complete(self):
    self.level += 1
    self.aliens_group.empty()
    self.alien_lasers_group.empty()
    self.mystery_ship_group.empty()
    self.extra_life_group.empty()
    self.shield_group.empty()
```

```
    self.create_aliens()

    def game_over(self):
        self.run = False

    def reset(self):
        self.run = True
        self.lives = 3
        self.score = 0
        self.level = 1
        self.spaceship_group.sprite.reset()
        self.aliens_group.empty()
        self.alien_lasers_group.empty()
        self.mystery_ship_group.empty()
        self.extra_life_group.empty()
        self.shield_group.empty()
        self.obstacles = self.create_obstacles()
        self.create_aliens()
        self.shield_active = False

    def check_for_highscore(self):
        from settings import HIGHSCORE_FILE
        if self.score > self.highscore:
            self.highscore = self.score
            with open(HIGHSCORE_FILE, "w") as file:
                file.write(str(self.highscore))

    def load_highscore(self):
        from settings import HIGHSCORE_FILE
        try:
            with open(HIGHSCORE_FILE, "r") as file:
                self.highscore = int(file.read())
        except FileNotFoundError:
            self.highscore = 0

    def update_shield(self):
        if self.shield_active:
            if pygame.time.get_ticks() - self.shield_timer >=
self.shield_duration:
                self.shield_active = False
```

import pygame, random: Baris ini mengimpor library Pygame, yang menyediakan fungsionalitas inti untuk pengembangan permainan 2D, seperti manajemen grafis, input, dan suara, serta mengimpor modul random yang akan digunakan untuk menghasilkan angka acak dalam permainan.

from spaceship import Spaceship: Baris ini mengimpor kelas Spaceship dari modul spaceship, yang kemungkinan besar merepresentasikan objek pesawat pemain, termasuk logika pergerakan, tembakan, dan interaksinya dengan elemen permainan lainnya.

from obstacle import Obstacle, grid: Baris ini mengimpor kelas Obstacle dan variabel grid dari modul obstacle. Kelas Obstacle kemungkinan merepresentasikan rintangan dalam permainan yang terdiri dari beberapa blok, dan grid adalah struktur data yang menentukan layout rintangan.

from alien import Alien, MysteryShip: Baris ini mengimpor kelas Alien dan MysteryShip dari modul alien, yang kemungkinan besar merepresentasikan musuh-musuh dalam permainan, masing-masing dengan perilaku dan karakteristik yang berbeda.

from laser import Laser: Baris ini mengimpor kelas Laser dari modul laser, yang kemungkinan besar merepresentasikan proyektil yang ditembakkan oleh pesawat pemain dan alien, dengan logika pergerakan dan deteksi tabrakan.

from powerup import ExtraLife, Shield: Baris ini mengimpor kelas ExtraLife dan Shield dari modul powerup, yang kemungkinan merepresentasikan item yang dapat dikumpulkan pemain untuk mendapatkan keuntungan, seperti menambah nyawa atau memberikan shield.

**from settings import SCREEN_WIDTH, SCREEN_HEIGHT,
OFFSET, ALIEN_MOVE_DOWN_DISTANCE,
EXPLOSION_SOUND_PATH, MUSIC_PATH,**

HIGHSCORE_FILE, POWERUP_SOUND_PATH,
ALIEN_START_X, ALIEN_START_Y,
ALIEN_HORIZONTAL_SPACING,
ALIEN_VERTICAL_SPACING: Baris ini mengimpor berbagai konstanta konfigurasi dari modul settings, termasuk dimensi layar, offset, jarak pergerakan alien, path suara ledakan, path musik latar, path file highscore, path suara powerup, dan pengaturan posisi awal alien. Ini memungkinkan konfigurasi eksternal dan konsisten untuk permainan.

Kelas Game: Kelas utama ini mengelola seluruh logika permainan. Konstruktornya (`__init__`) menginisialisasi semua elemen permainan seperti grup sprite untuk pesawat luar angkasa, alien, laser, power-up, dan rintangan, serta mengatur variabel-variabel penting seperti nyawa, skor, dan status permainan. Ia juga memuat skor tertinggi, musik latar belakang, dan durasi perisai.

Fungsi create_obstacles: Fungsi ini bertanggung jawab untuk membuat rintangan (obstacle) yang melindungi pemain dari tembakan alien. Fungsi ini menggunakan grid yang ada dari modul obstacle untuk mengatur tata letak dan posisi rintangan secara merata di sepanjang bagian bawah layar.

Fungsi create.aliens: Fungsi ini menciptakan formasi alien pada awal setiap level. Fungsi ini mengatur posisi alien berdasarkan baris dan kolom, dan menentukan jenis alien (1, 2, atau 3) berdasarkan barisnya. Alien-alien ini kemudian ditambahkan ke grup alien.

Fungsi move.aliens: Fungsi ini menggerakkan formasi alien secara horizontal. Ketika alien mencapai tepi layar, arah pergerakannya akan berbalik, dan seluruh formasi bergerak turun sedikit.

Fungsi alien_move_down: Fungsi ini menurunkan formasi alien secara vertikal dan juga memeriksa apakah alien mencapai bagian bawah layar, yang akan memicu kondisi game over jika terjadi.

Fungsi alien_shoot_laser: Fungsi ini membuat alien menembakkan laser secara acak. Fungsi ini memilih alien secara acak dari grup alien,

membuat objek laser pada posisi alien tersebut, dan menambahkan laser ke grup laser alien.

Fungsi `create_mystery_ship`: Fungsi ini membuat kapal misteri dan menaruhnya dalam grup kapal misteri, kapal misteri ini bergerak dari sisi layar dan memberikan poin jika ditembak pemain.

Fungsi `create_extra_life`: Fungsi ini membuat objek power-up ekstra nyawa dan menambahkannya ke grup power-up ekstra nyawa, powerup akan memberikan ekstra nyawa jika pemain menabrak power up tersebut.

Fungsi `create_shield`: Fungsi ini membuat objek power-up perisai dan menambahkannya ke grup power-up perisai, powerup akan mengaktifkan shield sementara jika pemain menabrak power up tersebut.

Fungsi `check_for_collisions`: Fungsi ini memeriksa semua kemungkinan tabrakan dalam permainan. Ia memeriksa tabrakan antara laser pemain dengan alien dan kapal misteri, laser alien dengan pesawat pemain, alien dengan rintangan dan pesawat pemain, serta tabrakan dengan power-up. Fungsi ini juga menangani pengurangan nyawa, penambahan skor, dan aktivasi power-up perisai.

Fungsi `level_complete`: Fungsi ini dipanggil saat pemain berhasil menghabiskan semua alien di level saat ini. Fungsi ini meningkatkan level permainan, mengosongkan grup sprite alien, laser, dan kapal misteri, dan memanggil fungsi `create_aliens` untuk membuat formasi alien baru untuk level berikutnya.

Fungsi `game_over`: Fungsi ini mengatur status permainan ke 'game over' dengan mengatur variabel run menjadi false.

Fungsi `reset`: Fungsi ini mengembalikan permainan ke keadaan awal, mengeset ulang nyawa, skor, level, dan mengosongkan semua grup sprite. Fungsi ini juga membuat kembali rintangan dan alien untuk memulai permainan baru.

Fungsi check_for_highscore: Fungsi ini memeriksa apakah skor saat ini lebih tinggi dari skor tertinggi yang tersimpan. Jika iya, skor tertinggi akan diupdate dan disimpan ke file.

Fungsi load_highscore: Fungsi ini memuat skor tertinggi dari file pada awal permainan. Jika file tidak ditemukan, skor tertinggi akan diset ke 0.

Fungsi update_shield: Fungsi ini mengelola durasi aktifnya perisai.

Perisai akan dinonaktifkan jika waktu yang telah berlalu sejak aktivasi melebihi durasi perisai yang telah ditentukan.

- File *laser.py*

```
import pygame
from settings import LASER_COLOR, LASER_WIDTH, LASER_HEIGHT

class Laser(pygame.sprite.Sprite):
    def __init__(self, position, speed, screen_height):
        super().__init__()
        self.image = pygame.Surface((LASER_WIDTH, LASER_HEIGHT))
        self.image.fill(LASER_COLOR)
        self.rect = self.image.get_rect(center = position)
        self.speed = speed
        self.screen_height = screen_height

    def update(self):
        self.rect.y -= self.speed
        if self.rect.y > self.screen_height + LASER_HEIGHT or
self.rect.y < 0:
            self.kill()
```

import pygame: Baris ini mengimpor library Pygame, yang menyediakan alat dan fungsi penting untuk pengembangan permainan 2D, termasuk grafis, input, dan pengelolaan sprite. Dengan mengimpor Pygame, kita mendapatkan akses ke kelas dan metode yang diperlukan untuk membuat dan mengelola elemen-elemen visual permainan.

from settings import LASER_COLOR, LASER_WIDTH,

LASER_HEIGHT: Baris ini mengimpor variabel konfigurasi khusus

untuk laser dari modul "settings," memungkinkan kita untuk dengan mudah mengubah properti seperti warna, lebar, dan tinggi laser di satu tempat dan menerapkannya di seluruh kode, menjaga konsistensi dan kemudahan pemeliharaan.

Kelas Laser(pygame.sprite.Sprite): Kelas Laser mendefinisikan objek laser dalam permainan, mewarisi dari pygame.sprite.Sprite untuk memanfaatkan fitur pengelolaan sprite dari Pygame. Kelas ini menyimpan informasi seperti gambar laser, posisi, kecepatan, dan tinggi layar, serta memiliki metode untuk menginisialisasi objek dan memperbarui pergerakannya di setiap frame.

Fungsi __init__(self, position, speed, screen_height): Metode __init__ adalah konstruktor untuk kelas Laser, yang diaktifkan saat objek laser dibuat. Metode ini menginisialisasi atribut-atribut laser seperti gambar (berdasarkan variabel konfigurasi), posisi awal, kecepatan pergerakan, dan tinggi layar, menyiapkan objek laser untuk digunakan dalam permainan.

Fungsi update(self): Metode update bertanggung jawab untuk memindahkan objek laser setiap frame. Metode ini mengurangi koordinat Y laser (memindahkannya ke atas), dan memeriksa apakah laser sudah keluar dari layar. Jika laser sudah tidak terlihat, metode ini akan menghapusnya dari memori, membantu mengoptimalkan kinerja permainan.

- File *main.py*

```
import pygame, sys, random
from game import Game
from settings import SCREEN_WIDTH, SCREEN_HEIGHT, OFFSET, GREY, COLOR,
FONT_PATH, FONT_SIZE, ALIEN_SHOOT_DELAY, MYSTERY_SHIP_SPAWN_MIN,
MYSTERY_SHIP_SPAWN_MAX, EXTRA_LIFE_SPAWN_MIN, EXTRA_LIFE_SPAWN_MAX,
SHIELD_SPAWN_MIN, SHIELD_SPAWN_MAX, SHIELD_COLOR, SHIELD_THICKNESS,
SHIELD_V_OFFSET_TOP, SHIELD_V_WIDTH, SHIELD_V_HEIGHT

class Main:
```

```
def __init__(self):
    pygame.init()
    self.screen = pygame.display.set_mode((SCREEN_WIDTH + OFFSET,
SCREEN_HEIGHT + 2 * OFFSET))
    pygame.display.set_caption("Python Space Invaders by Khair")
    self.clock = pygame.time.Clock()
    self.game = Game(SCREEN_WIDTH, SCREEN_HEIGHT, OFFSET)
    self.font = pygame.font.Font(FONT_PATH, FONT_SIZE)
    self.score_text_surface = self.font.render("SCORE", False,
COLOR)
    self.highscore_text_surface = self.font.render("HIGH-SCORE",
False, COLOR)

    self.shoot_laser = pygame.USEREVENT
    pygame.time.set_timer(self.shoot_laser, ALIEN_SHOOT_DELAY)

    self.mystery_ship = pygame.USEREVENT + 1
    pygame.time.set_timer(self.mystery_ship,
random.randint(MYSTERY_SHIP_SPAWN_MIN, MYSTERY_SHIP_SPAWN_MAX))

    self.spawn_extra_life = pygame.USEREVENT + 2
    pygame.time.set_timer(self.spawn_extra_life,
random.randint(EXTRA_LIFE_SPAWN_MIN, EXTRA_LIFE_SPAWN_MAX))

    self.spawn_shield = pygame.USEREVENT + 3
    pygame.time.set_timer(self.spawn_shield,
random.randint(SHIELD_SPAWN_MIN, SHIELD_SPAWN_MAX))

def run(self):
    while True:
        self._handle_events()
        self._update()
        self._draw()

def _handle_events(self):
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            pygame.quit()
            sys.exit()
        if event.type == self.shoot_laser and self.game.run:
            self.game.alien_shoot_laser()

        if event.type == self.mystery_ship and self.game.run:
```

```
        self.game.create_mystery_ship()
        pygame.time.set_timer(self.mystery_ship,
random.randint(MYSTERY_SHIP_SPAWN_MIN, MYSTERY_SHIP_SPAWN_MAX))

        if event.type == self.spawn_extra_life and self.game.run:
            self.game.create_extra_life()
            pygame.time.set_timer(self.spawn_extra_life,
random.randint(EXTRA_LIFE_SPAWN_MIN, EXTRA_LIFE_SPAWN_MAX))

        if event.type == self.spawn_shield and self.game.run:
            self.game.create_shield()
            pygame.time.set_timer(self.spawn_shield,
random.randint(SHIELD_SPAWN_MIN, SHIELD_SPAWN_MAX))

    keys = pygame.key.get_pressed()
    if keys[pygame.K_SPACE] and not self.game.run:
        self.game.reset()

    def _update(self):
        if self.game.run:
            self.game.spaceship_group.update()
            self.game.move_aliens()
            self.game.alien_lasers_group.update()
            self.game.mystery_ship_group.update()
            self.game.extra_life_group.update()
            self.game.shield_group.update()
            self.game.check_for_collisions()
            self.game.update_shield()

    def _draw(self):
        self.screen.fill(GREY)

        # UI
        pygame.draw.rect(self.screen, COLOR, (10, 10, SCREEN_WIDTH +
OFFSET - 20, SCREEN_HEIGHT + 2 * OFFSET - 20), 2, 0, 60, 60, 60, 60)
        pygame.draw.line(self.screen, COLOR, (25, SCREEN_HEIGHT + 2 *
OFFSET - 70), (SCREEN_WIDTH + OFFSET - 25, SCREEN_HEIGHT + 2 * OFFSET -
70), 3)

        level_surface = self.font.render("LEVEL " + str(self.game.level), False, COLOR)
        if self.game.run:
```

```

        self.screen.blit(level_surface, (570, SCREEN_HEIGHT + 2 *
OFFSET - 60, 50, 50))
    else:
        game_over_surface = self.font.render("GAME OVER", False,
COLOR)
        self.screen.blit(game_over_surface, (570, SCREEN_HEIGHT + 2
* OFFSET - 60, 50, 50))

        x = 50
        for life in range(self.game.lives):
            self.screen.blit(self.game.spaceship_group.sprite.image,
(x, SCREEN_HEIGHT + 2 * OFFSET - 55))
            x += 50

        self.screen.blit(self.score_text_surface, (50, 15, 50, 50))
        formatted_score = str(self.game.score).zfill(5)
        score_surface = self.font.render(formatted_score, False, COLOR)
        self.screen.blit(score_surface, (50, 40, 50, 50))
        self.screen.blit(self.highscore_text_surface, (550, 15, 50,
50))
        formatted_highscore = str(self.game.highscore).zfill(5)
        highscore_surface = self.font.render(formatted_highscore,
False, COLOR)
        self.screen.blit(highscore_surface, (625, 40, 50, 50))

        self.game.spaceship_group.draw(self.screen)
        self.game.spaceship_group.sprite.lasers_group.draw(self.screen)
        for obstacle in self.game.obstacles:
            obstacle.blocks_group.draw(self.screen)
        self.game.aliens_group.draw(self.screen)
        self.game.alien_lasers_group.draw(self.screen)
        self.game.mystery_ship_group.draw(self.screen)
        self.game.extra_life_group.draw(self.screen)
        self.game.shield_group.draw(self.screen)

        if self.game.shield_active and self.game.run:
            spaceship = self.game.spaceship_group.sprite
            center_x = spaceship.rect.centerx
            top_y = spaceship.rect.top

            # Define the points for the inverted "V" shape
            point1 = (center_x, top_y - SHIELD_V_OFFSET_TOP) # Tip of
the "V"

```

```

        point2 = (center_x - SHIELD_V_WIDTH // 2, top_y -
SHIELD_V_OFFSET_TOP + SHIELD_V_HEIGHT) # Bottom-left
        point3 = (center_x + SHIELD_V_WIDTH // 2, top_y -
SHIELD_V_OFFSET_TOP + SHIELD_V_HEIGHT) # Bottom-right

    pygame.draw.polygon(self.screen, SHIELD_COLOR, [point1,
point2, point3], SHIELD_THICKNESS)

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

if __name__ == '__main__':
    main = Main()
    main.run()

```

import pygame, sys, random: Baris ini mengimpor library Pygame untuk permainan, modul sys untuk operasi sistem, dan modul random untuk menghasilkan angka acak, yang akan digunakan untuk berbagai fungsi dalam permainan seperti kemunculan objek secara acak.

from game import Game: Baris ini mengimpor kelas Game dari modul terpisah, yang kemungkinan besar berisi logika inti dari permainan Space Invaders, seperti gerakan alien, deteksi tabrakan, dan aturan permainan.

Ini memisahkan logika permainan dari kode manajemen permainan utama.

from settings import ...: Baris ini mengimpor berbagai konstanta dari modul "settings," seperti dimensi layar, warna, dan interval kemunculan event. Ini memusatkan konfigurasi permainan di satu tempat, membuatnya lebih mudah untuk diubah dan dikelola.

Kelas Main: Kelas Main adalah kelas utama yang mengelola keseluruhan alur permainan. Kelas ini bertanggung jawab untuk inisialisasi Pygame, membuat jendela permainan, menangani input pengguna, memperbarui logika permainan, dan menggambar semua elemen permainan di layar.

Fungsi __init__(self): Metode __init__ adalah konstruktur kelas Main. Metode ini menginisialisasi Pygame, membuat layar permainan, membuat objek Clock, menginisialisasi kelas Game, membuat objek Font untuk

teks, mengatur user event seperti tembakan alien, kemunculan kapal misteri, extra life dan shield serta mengatur timer untuk event tersebut.

Fungsi run(self): Metode run adalah loop utama permainan. Metode ini terus menerus memanggil _handle_events untuk menangani event, _update untuk memperbarui logika permainan, dan _draw untuk merender elemen-elemen permainan, memastikan permainan berjalan dengan lancar.

Fungsi _handle_events(self): Metode _handle_events menangani semua event Pygame, seperti input keyboard, user event dan event tutup jendela. Metode ini juga memanggil fungsi-fungsi yang sesuai dalam kelas Game ketika suatu event tertentu terjadi.

Fungsi _update(self): Metode _update bertanggung jawab untuk memperbarui semua elemen permainan. Metode ini akan memanggil fungsi update di semua objek permainan, memanggil fungsi move_aliens untuk memindahkan alien, memanggil fungsi check_for_collisions untuk menangani tabrakan dan memanggil fungsi update_shield untuk menangani shield.

Fungsi _draw(self): Metode _draw merender semua elemen permainan ke layar. Ini termasuk mengisi layar dengan warna latar belakang, menggambar UI, menggambar sprite permainan, menggambar shield, dan memperbarui tampilan layar.

if __name__ == '__main__': Blok ini memastikan bahwa kode di dalamnya hanya dijalankan jika file ini dijalankan sebagai program utama. Ini digunakan untuk membuat instance Main dan menjalankan loop utama permainan.

- File *obstacle.py*

```
import pygame
from settings import BLOCK_COLOR, BLOCK_SIZE

class Block(pygame.sprite.Sprite):
    def __init__(self, x, y):
```

import pygame: Baris ini mengimpor library Pygame, yang menyediakan fungsionalitas untuk membuat permainan 2D, termasuk pembuatan sprite dan pengelompokannya.

from settings import BLOCK_COLOR, BLOCK_SIZE: Baris ini mengimpor variabel konfigurasi khusus untuk blok dari modul "settings", seperti warna dan ukuran blok, yang memungkinkan kita untuk mengontrol properti visual blok dari satu lokasi terpusat.

Kelas Block(pygame.sprite.Sprite): Kelas Block mendefinisikan objek blok individual yang akan digunakan untuk membuat obstacle dalam permainan, mewarisi dari pygame.sprite.Sprite untuk memanfaatkan fitur manajemen sprite dan menggunakan parameter konfigurasi dari settings.

Fungsi __init__(self, x, y): Metode __init__ adalah konstruktur untuk kelas Block, yang menginisialisasi objek blok dengan membuat gambar pygame.Surface dengan ukuran dan warna yang ditentukan, serta menetapkan posisi awal blok berdasarkan koordinat x dan y.

grid = [...]: Variabel grid adalah struktur data berupa daftar bersarang yang merepresentasikan layout obstacle dalam bentuk pola blok. Nilai 1 menunjukkan keberadaan blok, sedangkan nilai 0 menunjukkan tidak ada blok, membentuk dasar visual obstacle.

Kelas Obstacle: Kelas Obstacle merepresentasikan sebuah obstacle dalam permainan, yang terdiri dari kumpulan objek blok. Kelas ini memiliki metode untuk membuat obstacle berdasarkan pola blok yang diFungsi dalam variabel grid.

Fungsi __init__(self, x, y): Metode __init__ adalah konstruktur untuk kelas Obstacle. Metode ini mengambil koordinat x dan y untuk menentukan posisi obstacle, membuat grup sprite untuk menyimpan objek-objek blok, dan membuat objek Block berdasarkan grid dan menambahkan mereka ke grup sprite.

- File *powerup.py*

```
import pygame
import random
from settings import EXTRA_LIFE_IMAGE_PATH, SHIELD_POWERUP_IMAGE_PATH,
SCREEN_WIDTH

class ExtraLife(pygame.sprite.Sprite):
    def __init__(self, screen_width, offset):
        super().__init__()
        original_image = pygame.image.load(EXTRA_LIFE_IMAGE_PATH)
```

```

        self.image = pygame.transform.scale(original_image, (30, 24))
        self.rect = self.image.get_rect(center=(random.randint(offset,
screen_width), -50))
        self.speed = 2

    def update(self):
        from settings import SCREEN_HEIGHT
        self.rect.y += self.speed
        if self.rect.top > SCREEN_HEIGHT:
            self.kill()

class Shield(pygame.sprite.Sprite):
    def __init__(self, screen_width, offset):
        super().__init__()
        original_image = pygame.image.load(SHIELD_POWERUP_IMAGE_PATH)
        self.image = pygame.transform.scale(original_image, (30,
38.65))
        self.rect = self.image.get_rect(center=(random.randint(offset,
screen_width), -50))
        self.speed = 3

    def update(self):
        from settings import SCREEN_HEIGHT
        self.rect.y += self.speed
        if self.rect.top > SCREEN_HEIGHT:
            self.kill()

```

import pygame: Baris ini mengimpor library Pygame, yang menyediakan fungsionalitas untuk membuat permainan 2D, termasuk manajemen sprite, gambar, dan transformasi.

import random: Baris ini mengimpor modul random, yang akan digunakan untuk menghasilkan angka acak untuk posisi awal power-up di layar.

from settings import EXTRA_LIFE_IMAGE_PATH,

SHIELD_POWERUP_IMAGE_PATH, SCREEN_WIDTH: Baris ini mengimpor path gambar dan lebar layar dari modul settings, yang memungkinkan konfigurasi eksternal dan konsisten untuk asset permainan.

Kelas ExtraLife(pygame.sprite.Sprite): Kelas ExtraLife mendefinisikan objek power-up extra life, mewarisi dari pygame.sprite.Sprite, untuk memanfaatkan fitur manajemen sprite. Kelas ini memiliki metode untuk inisialisasi objek dan memperbarui posisi objek di setiap frame permainan.

Fungsi __init__(self, screen_width, offset): Metode __init__ adalah konstruktor untuk kelas ExtraLife. Metode ini menginisialisasi objek dengan memuat dan menskalakan gambar dari file, menetapkan posisi awal acak di luar bagian atas layar, dan menetapkan kecepatan jatuhnya.

Fungsi update(self): Metode update digunakan untuk memperbarui posisi vertikal objek ExtraLife di setiap frame permainan, dan menghapusnya dari grup sprite jika sudah mencapai bagian bawah layar.

Kelas Shield(pygame.sprite.Sprite): Kelas Shield mendefinisikan objek power-up shield, mewarisi dari pygame.sprite.Sprite, untuk memanfaatkan fitur manajemen sprite. Kelas ini memiliki metode untuk inisialisasi objek dan memperbarui posisi objek di setiap frame permainan.

Fungsi __init__(self, screen_width, offset): Metode __init__ adalah konstruktor untuk kelas Shield. Metode ini menginisialisasi objek dengan memuat dan menskalakan gambar dari file, menetapkan posisi awal acak di luar bagian atas layar, dan menetapkan kecepatan jatuhnya.

Fungsi update(self): Metode update digunakan untuk memperbarui posisi vertikal objek Shield di setiap frame permainan, dan menghapusnya dari grup sprite jika sudah mencapai bagian bawah layar.

- File *spaceship.py*

```
import pygame
from laser import Laser
from settings import SPACESHIP_IMAGE_PATH, SCREEN_WIDTH, SCREEN_HEIGHT,
OFFSET, SPACESHIP_LASER_SPEED, LASER_SOUND_PATH

class Spaceship(pygame.sprite.Sprite):
    def __init__(self, screen_width, screen_height, offset):
        super().__init__()
```

```
    self.offset = offset
    self.screen_width = screen_width
    self.screen_height = screen_height
    self.image = pygame.image.load(SPACESHIP_IMAGE_PATH)
    self.rect = self.image.get_rect(midbottom = ((self.screen_width
+ self.offset)/2, self.screen_height))
    self.speed = 6
    self.lasers_group = pygame.sprite.Group()
    self.laser_ready = True
    self.laser_time = 0
    self.laser_delay = 300
    self.laser_sound = pygame.mixer.Sound(LASER_SOUND_PATH)

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

        if keys[pygame.K_RIGHT]:
            self.rect.x += self.speed

        if keys[pygame.K_LEFT]:
            self.rect.x -= self.speed

        if keys[pygame.K_SPACE] and self.laser_ready:
            self.laser_ready = False
            laser = Laser(self.rect.center, SPACESHIP_LASER_SPEED,
self.screen_height)
            self.lasers_group.add(laser)
            self.laser_time = pygame.time.get_ticks()
            self.laser_sound.play()

    def update(self):
        self.get_user_input()
        self.constrain_movement()
        self.lasers_group.update()
        self.recharge_laser()

    def constrain_movement(self):
        if self.rect.right > self.screen_width:
            self.rect.right = self.screen_width
        if self.rect.left < self.offset:
            self.rect.left = self.offset

    def recharge_laser(self):
```

```
if not self.laser_ready:
    current_time = pygame.time.get_ticks()
    if current_time - self.laser_time >= self.laser_delay:
        self.laser_ready = True

def reset(self):
    self.rect = self.image.get_rect(midbottom = ((self.screen_width
+ self.offset)/2, self.screen_height))
    self.lasers_group.empty()
```

import pygame: Baris ini mengimpor library Pygame, yang menyediakan fungsionalitas untuk membuat permainan 2D, termasuk manajemen sprite, input, dan suara.

from laser import Laser: Baris ini mengimpor kelas Laser dari modul laser, yang akan digunakan untuk membuat objek laser yang ditembakkan oleh pesawat luar angkasa.

**from settings import SPACESHIP_IMAGE_PATH,
SCREEN_WIDTH, SCREEN_HEIGHT, OFFSET,
SPACESHIP_LASER_SPEED, LASER_SOUND_PATH:** Baris ini mengimpor berbagai konstanta dari modul settings, seperti path gambar pesawat luar angkasa, dimensi layar, offset, kecepatan laser pesawat luar angkasa, dan path suara laser, untuk konfigurasi eksternal dan konsistensi.

Kelas Spaceship(pygame.sprite.Sprite): Kelas Spaceship mendefinisikan objek pesawat luar angkasa pemain, mewarisi dari pygame.sprite.Sprite, untuk memanfaatkan fitur manajemen sprite. Kelas ini memiliki metode untuk inisialisasi objek, menangani input pengguna, memperbarui posisi, mengelola tembakan laser dan batasan gerakan, dan me-reset pesawat.

Fungsi __init__(self, screen_width, screen_height, offset): Metode __init__ adalah konstruktor untuk kelas Spaceship. Metode ini menginisialisasi objek dengan memuat gambar pesawat, mengatur posisi awal, menetapkan kecepatan, membuat grup untuk laser pesawat,

menginisialisasi state laser ready, waktu laser, delay laser, dan memuat sound effect untuk laser.

Fungsi get_user_input(self): Metode get_user_input mengambil input dari pengguna untuk menggerakkan pesawat ke kiri atau kanan dan menembakkan laser. Metode ini memeriksa tombol yang ditekan, menggerakkan pesawat, dan membuat instance laser baru jika tombol spasi ditekan dan laser dalam kondisi ready, serta memainkan suara laser.

Fungsi update(self): Metode update dipanggil setiap frame permainan. Metode ini memanggil get_user_input, membatasi pergerakan pesawat, mengupdate posisi laser yang ditembakkan dan memanggil fungsi untuk me-recharge laser.

Fungsi constrain_movement(self): Metode constrain_movement digunakan untuk memastikan pesawat tidak bergerak keluar dari batas layar, dengan membatasi koordinat x dari rectangle pesawat.

Fungsi recharge_laser(self): Metode recharge_laser digunakan untuk mengelola delay tembakan laser. Metode ini mengecek apakah delay sudah habis, dan mengembalikan state laser ready menjadi True jika sudah.

Fungsi reset(self): Metode reset digunakan untuk mengembalikan posisi pesawat ke posisi awal dan mengosongkan grup laser pesawat.

C. LAMPIRAN

Permainan dasar

<https://github.com/educ8s/Python-Space-Invaders-Game-with-Pygame>

Lampiran AI

https://aistudio.google.com/app/prompts?state=%7B%22ids%22:%5B%221kzz_ME_Va7bJUF8FMNgP23R0wWFS87_x%22%5D,%22action%22:%22open%22,%22userId%22:%22115119950635860904125%22,%22resourceKeys%22:%7B%7D%7D&usp=sharing