

**Game *Snake Retro* Berbasis Pycairo dan Pygame Menggunakan
Konsep Dasar Grafika Komputer**



AHMAD AMIN AMRULLAH	212410103064
BRAMUDYA MELVAN IBRAHIM	232410103097
HARIS DWI RAMADHAN	232410103083

**PROGRAM STUDI INFORMATIKA
UNIVERSITAS JEMBER
2024**

Game *Snake Retro* Berbasis Pycairo dan Pygame Menggunakan Konsep Dasar Grafika Komputer

A. Deskripsi Umum

. Proyek akhir ini bertujuan untuk mengembangkan sebuah game sederhana berbasis *Snake'97*, yang mengaplikasikan konsep-konsep dasar Grafika Komputer menggunakan Pycairo dan Pygame. Game ini akan menerapkan teknik grafis sederhana, seperti rendering objek 2D dan pengaturan animasi dasar, untuk menciptakan pengalaman bermain yang klasik namun tetap menarik. Pycairo akan digunakan untuk mengelola elemen grafis yang berbasis vektor, sementara Pygame akan membantu dalam membangun interaksi dan mengatur elemen permainan.

Game ini dirancang untuk mengatasi tantangan klasik dalam pengembangan game sederhana: bagaimana menciptakan interaksi yang lancar, tampilan yang jelas, dan kontrol yang intuitif meskipun dalam format permainan yang minimalis. Melalui proyek ini, diharapkan pengguna dapat menikmati pengalaman yang nostalgik dan intuitif, sekaligus memahami bagaimana konsep grafika komputer dapat diterapkan untuk membangun aplikasi interaktif.

Permainan ini memiliki potensi untuk diterapkan sebagai media pembelajaran dasar dalam pemrograman grafika komputer, terutama bagi pemula yang ingin mempelajari pengembangan game. Selain itu, proyek ini menunjukkan bagaimana konsep grafika sederhana dapat menghasilkan aplikasi yang tetap menarik dan memiliki nilai hiburan serta edukasi.

BAB I

PENDAHULUAN

a. Latar Belakang

Game klasik Snake adalah salah satu game legendaris yang populer sejak era ponsel-ponsel pertama. Snake menawarkan gameplay yang sederhana namun adiktif, di mana pemain mengontrol seekor ular yang harus memakan objek tertentu untuk tumbuh lebih panjang, sambil menghindari tembok dan ekornya sendiri. Popularitas game ini berasal dari gameplay yang sederhana namun membutuhkan ketangkasan dan konsentrasi. Dengan kemajuan teknologi dan perangkat lunak grafis, game seperti Snake dapat direalisasikan kembali dalam bentuk modern yang memanfaatkan teknik grafika komputer yang lebih canggih namun tetap mempertahankan gaya retro yang ikonik.

Permasalahan utama dalam mengembangkan game sederhana ini adalah bagaimana menyusun elemen-elemen visual dan interaktif menggunakan teknik dasar grafika komputer tanpa mengurangi nilai nostalgia dari game aslinya. Dengan menggunakan Pycairo dan Pygame, teknologi grafika komputer yang didukung oleh bahasa pemrograman Python, pengembangan game Snake Retro ini dapat dilakukan secara efektif dan efisien. Pycairo memungkinkan pemrosesan grafis berbasis vektor yang optimal, sedangkan Pygame mendukung aspek interaktif seperti pengendalian karakter dan pengaturan frame rate, yang penting untuk mempertahankan kualitas gameplay.

Proyek ini juga relevan dengan materi Grafika Komputer karena memanfaatkan berbagai konsep dasar seperti penggambaran objek 2D, penggunaan vektor, pengaturan animasi, dan rendering berbasis Python. Dengan demikian, game ini bukan hanya untuk hiburan, tetapi juga berfungsi sebagai contoh pembelajaran bagi mahasiswa atau pemula yang ingin memahami aplikasi konsep grafika komputer dalam pengembangan game sederhana. Nilai tambah dari proyek ini adalah pembuatan game yang tidak hanya memupuk keterampilan teknis dalam pemrograman grafis tetapi juga membawa kembali pengalaman nostalgia yang menghibur bagi pemain.

b. Tujuan dan Manfaat

1. Tujuan: Tujuan dari proyek ini adalah mengembangkan sebuah game sederhana berbasis Snake dengan tampilan grafis retro, yang dapat dimainkan dengan lancar dan menyenangkan. Proyek ini juga bertujuan sebagai media pembelajaran penerapan konsep grafika komputer dengan menggunakan Pycairo dan Pygame.
2. Manfaat: Manfaat dari proyek ini mencakup penyediaan game klasik yang menyenangkan bagi pengguna serta menjadi contoh yang relevan bagi mahasiswa atau pengembang pemula yang ingin belajar tentang pemrograman game berbasis grafika komputer. Di lingkungan akademik, proyek ini bisa menjadi referensi untuk memahami bagaimana konsep dasar grafika komputer diaplikasikan dalam pembuatan game.

BAB 2

Tinjauan Pustaka

1. Teori Grafika Komputer Yang Relevan

Grafika komputer adalah bidang yang mempelajari cara merepresentasikan dan memanipulasi objek visual secara digital. Dalam pengembangan game seperti *Snake Retro*, teori grafika komputer yang relevan meliputi beberapa konsep dasar, termasuk:

1. Rendering 2D: Proses menggambar objek grafis dua dimensi pada layar. Dalam konteks *Snake Retro*, objek seperti ular dan makanan di-render dalam format 2D menggunakan pustaka Pycairo, yang mendukung gambar berbasis vektor sehingga menghasilkan visual yang lebih halus dan skalabilitas yang baik.

2. Animasi dan Frame Rate: Animasi adalah teknik untuk menampilkan gambar secara berurutan untuk menciptakan ilusi gerakan. *Frame rate* mengacu pada jumlah gambar yang ditampilkan per detik. Dalam game ini, animasi diterapkan untuk menggerakkan ular secara kontinu dan memberikan respons real-time atas masukan pemain, yang didukung oleh Pygame.

3. Koordinat Kartesius: Penggunaan sistem koordinat untuk mengontrol posisi objek di layar. Pada game *Snake Retro*, posisi ular dan makanan ditentukan oleh koordinat yang diatur di dalam area permainan, yang penting untuk mengatur logika pergerakan dan deteksi tabrakan.

2. Studi Literatur Aplikasi Serupa

1. Game *Snake'97*: Game *Snake'97* yang populer pada ponsel-ponsel klasik Nokia menggunakan grafis yang sangat sederhana namun intuitif. Pergerakan ular berbasis arah yang tetap (atas, bawah, kiri, dan kanan) dan gameplay yang adiktif menjadi inspirasi utama untuk proyek ini. Mengadaptasi konsep dari game ini, *Snake Retro* juga akan mempertahankan elemen gameplay yang mirip, namun dengan peningkatan pada segi grafis dan responsivitas untuk platform modern.

2. **Pustaka Pycairo dan Pygame dalam Pengembangan Game:**

Pycairo adalah pustaka untuk mengelola gambar vektor di lingkungan Python, yang mendukung rendering objek-objek dasar. Di sisi lain, Pygame adalah pustaka yang sering digunakan dalam pengembangan game sederhana dengan bahasa Python karena mendukung kontrol, animasi, dan manajemen *event* yang penting dalam game. Banyak aplikasi edukatif atau proyek hobi yang memanfaatkan Pygame untuk membangun game sederhana. Kombinasi Pycairo dan Pygame memungkinkan pengembang untuk memanfaatkan vektor dan bitmap grafis sekaligus, menciptakan tampilan yang menarik tanpa mengorbankan performa.

3. Studi Literatur Algoritma Lainnya

1. **Algoritma Pergerakan Sederhana:** Dalam game seperti *Snake Retro*, pergerakan ular ditangani dengan algoritma dasar yang mengikuti arah input pengguna (kiri, kanan, atas, bawah). Ular bergerak ke arah tertentu berdasarkan koordinat yang ditambahkan atau dikurangi setiap kali pemain memberikan input. Algoritma pergerakan ini sederhana namun efektif untuk game dengan grafis dan kontrol minimalis.

2. **Algoritma Deteksi Tabrakan:** Salah satu elemen penting dalam game *Snake* adalah mendeteksi tabrakan, baik dengan tembok maupun ekor ular sendiri. Algoritma ini mengecek apakah posisi kepala ular bertabrakan dengan objek tertentu di grid permainan. Jika terjadi tabrakan, permainan akan berhenti atau mengatur ulang ular sesuai aturan yang ditentukan.

3. **Algoritma Pseudorandom untuk Posisi Makanan:** Untuk menambah elemen tantangan, posisi makanan dalam game diatur secara acak di dalam grid. Algoritma ini memastikan bahwa makanan muncul di lokasi baru yang tidak bertabrakan dengan posisi ular, sehingga menambah variasi dalam permainan.

BAB III

METODOLOGI

1. Perancangan antarmuka (UI)

Antarmuka dalam game Snake Retro dirancang dengan tampilan minimalis dan retro, mengacu pada desain Snake'97 yang sederhana dan intuitif. Elemen utama UI meliputi:

- Area Permainan: Tampilan area permainan berbentuk kotak dengan grid, di mana ular dan makanan muncul dan bergerak.
- Tampilan Skor: Di bagian atas layar, ditampilkan skor pemain yang meningkat seiring ular memakan makanan.
- Tombol Navigasi: Kontrol menggunakan keyboard (panah atas, bawah, kiri, kanan) untuk mengatur pergerakan ular.
- Animasi Sederhana: Ular akan menampilkan animasi gerakan saat bergerak sesuai input pemain.

2. Fitur-fitur utama dalam aplikasi

- Pergerakan Ular: Ular akan bergerak terus-menerus sesuai arah yang ditentukan oleh pemain, dan arah tersebut dapat diubah menggunakan tombol panah.
- Makan dan Bertambah Panjang: Setiap kali ular memakan makanan, tubuhnya bertambah panjang. Ini menambah tingkat kesulitan, karena pemain harus menghindari ekor ular sendiri.
- Deteksi Tabrakan: Game akan berakhir jika kepala ular menabrak dinding atau ekornya sendiri. Fitur ini menambah tantangan bagi pemain untuk terus bertahan hidup selama mungkin.
- Randomisasi Posisi Makanan: Makanan akan muncul di posisi acak yang tidak bertabrakan dengan ular, sehingga menambah variasi dalam gameplay.

3. Library yang digunakan

- Pycairo: Digunakan untuk menggambar elemen-elemen vektor 2D dalam permainan seperti tubuh ular dan objek makanan. Pycairo mendukung grafik yang tajam dan halus.

- Pygame: Digunakan sebagai kerangka kerja utama untuk mengelola aspek-aspek permainan, seperti tampilan, event handling, dan pengaturan frame rate. Pygame juga mendukung animasi sederhana dan kontrol keyboard yang digunakan untuk pergerakan ular.

4. Rencana pengujian:

Black Box Testing: Pengujian ini akan dilakukan dengan memainkan game secara langsung untuk mengamati apakah fitur-fitur utama, seperti kontrol ular, deteksi tabrakan, dan tampilan skor, berfungsi sesuai harapan. Pengujian ini memastikan bahwa pemain dapat berinteraksi dengan game tanpa masalah teknis.

5. Timeline pengerjaan proyek atau gantt chart

1. **Minggu 1 (13 November - 19 November 2024):**

- Perancangan konsep game dan UI sederhana
- Menentukan layout area permainan, tampilan skor, dan desain elemen dasar game (ular, makanan, dinding).
- Menyusun diagram alur permainan dan rencana pengembangan.

2. **Minggu 2 (20 November - 23 November 2024):**

- Implementasi dasar game menggunakan Pycairo dan Pygame
- Pengaturan area permainan dan penggambaran objek ular serta makanan.
- Pengaturan kontrol pergerakan ular (menggunakan tombol panah).
- Mengimplementasikan pengaturan dasar frame rate dan pergerakan ular.

3. **Minggu 3 (24 November - 26 November 2024):**

- Pengembangan fitur tambahan
- Menambahkan logika deteksi tabrakan (dengan dinding dan ekor ular).
- Menambahkan algoritma untuk menempatkan makanan secara acak di grid.
- Pengaturan tampilan skor dan fitur game over.

4. **Minggu 4 (27 November 2024):**

- Pengujian dan Debugging

- Melakukan pengujian black box untuk memastikan fungsi-fungsi utama berjalan dengan baik.
- Melakukan perbaikan berdasarkan hasil pengujian
- Menyusun laporan akhir dan dokumentasi proyek.

BAB IV

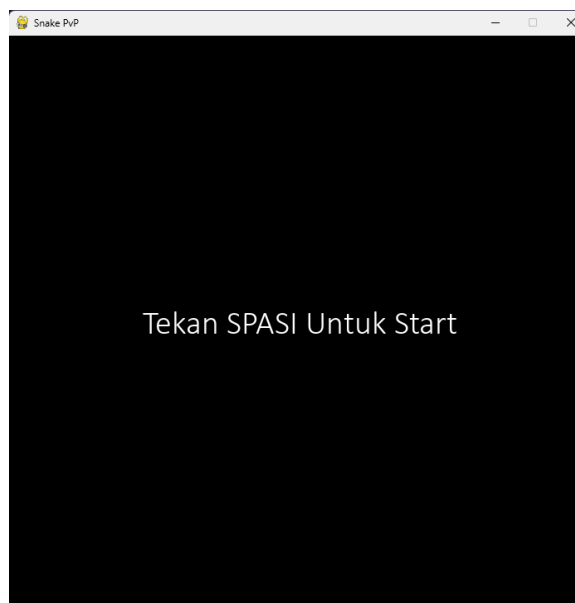
PEMBAHASAN

4.1 Tentang Game

Game ini adalah versi PvP dari Snake, di mana dua pemain mengendalikan ular masing-masing di layar, saling bersaing untuk makan makanan dan menghindari tabrakan. Pemain pertama yang menyebabkan ular lawan kalah (karena tabrakan atau keluar dari batas) adalah pemenangnya.

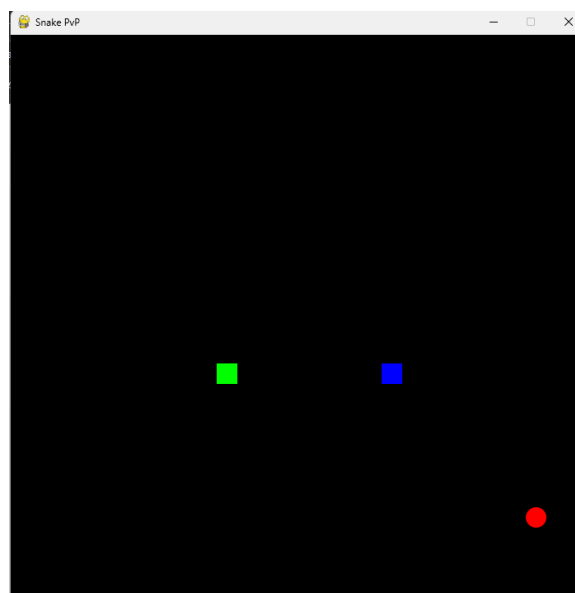
4.1.1 Page Start

Dalam page start ini player akan diperintahkan menekan tombol spasi untuk memulai game tersebut.



4.1.2 Page Permainan

Pada page ini player akan memulai game dengan ukuran ular 1 kotak dan makanan akan terspawn secara acak, para player harus berlomba - lomba untuk memakan makanan tersebut untuk tumbuh atau mencari strategi untuk mengalahkan lawan dengan cara lawan harus menabrak tubuh dari player



4.1.3 Page Winner

Pada page ini pemenang akan ditampilkan, dan player bisa menekan tombol spasi lagi untuk memulai ulang permainan



4.2 Penerapan Grafika Komputer

```
GAME_WIDTH = 700
GAME_HEIGHT = 700
SPEED = 5
SPACE_SIZE = 25
SNAKE_COLORS = [(0, 255, 0), (0, 0, 255)] #Player 1, Player 2
FOOD_COLOR = (255, 0, 0)
BACKGROUND_COLOR = (0, 0, 0)
```

Kode diatas adalah variabel yang akan digunakan nanti, seperti game width dan height yang digunakan untuk mengatur panjang dan lebar dari canvas, kemudian speed digunakan untuk mengatur kecepatan, kemudian space size digunakan untuk mengatur ukuran dari ular setiap kotaknya, kemudian ada snake colors untuk mengatur warna setiap ular dari masing-masing player, dan ada aja food color untuk mengatur warna makanan, dan terakhir ada background color untuk mengatur warna background dari game

```
class Snake:
    def __init__(self, color, start_x, start_y):
        self.body = [[start_x, start_y]]
        self.color = color
        self.direction = "up"
        self.growing = False
```

Kode di atas adalah class dari Snake, yang digunakan untuk merepresentasikan ular dalam permainan.

- `__init__` adalah metode inisialisasi untuk membuat objek Snake baru.
- `self.body` menyimpan posisi awal tubuh ular dalam bentuk list koordinat (x, y).
- `self.color` digunakan untuk menentukan warna ular.
- `self.direction` menyimpan arah pergerakan awal ular, yaitu "up" (ke atas).
- `self.growing` adalah flag untuk menandai apakah ular sedang tumbuh setelah makan makanan.
- Kode ini mengatur properti awal ular ketika permainan dimulai.

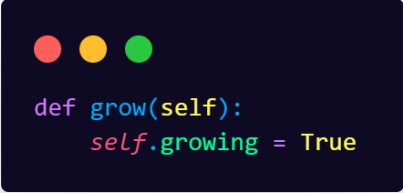
```
def move(self):
    x, y = self.body[0]
    if self.direction == "up":
        y -= SPACE_SIZE
    elif self.direction == "down":
        y += SPACE_SIZE
    elif self.direction == "left":
        x -= SPACE_SIZE
    elif self.direction == "right":
        x += SPACE_SIZE

    new_head = [x, y]
    self.body.insert(0, new_head)

    if not self.growing:
        self.body.pop()
    else:
        self.growing = False
```

Kode di atas adalah metode move dalam kelas Snake, yang digunakan untuk mengatur pergerakan ular di dalam permainan.

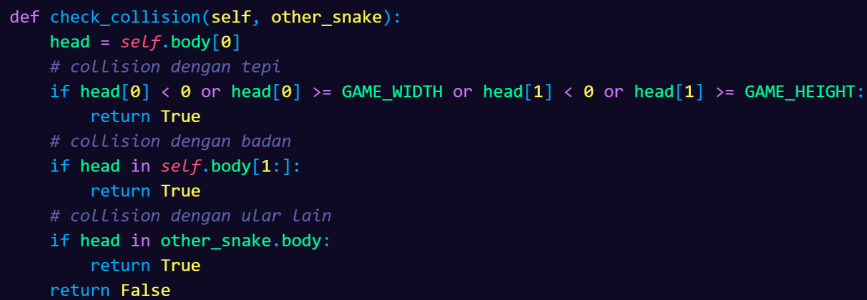
- Baris `x, y = self.body[0]` mengambil koordinat kepala ular (segmen pertama tubuh ular).
- Ada pengecekan `self.direction` untuk menentukan arah gerakan ular:
 - "up": Y berkurang sebesar `SPACE_SIZE` (bergerak ke atas).
 - "down": Y bertambah sebesar `SPACE_SIZE` (bergerak ke bawah).
 - "left": X berkurang sebesar `SPACE_SIZE` (bergerak ke kiri).
 - "right": X bertambah sebesar `SPACE_SIZE` (bergerak ke kanan).
- `new_head = [x, y]` membuat koordinat baru untuk kepala ular setelah bergerak.
- `self.body.insert(0, new_head)` menambahkan posisi kepala baru ke dalam tubuh ular.
- Jika ular tidak sedang tumbuh (`self.growing` adalah `False`), ekor ular dihapus dengan `self.body.pop()`, menjaga panjang tubuh tetap sama.
- Jika ular sedang tumbuh (`self.growing` adalah `True`), ekor tidak dihapus, dan flag `self.growing` diatur kembali menjadi `False`.



```
def grow(self):  
    self.growing = True
```

Kode di atas adalah metode `grow` dalam kelas `Snake`, yang digunakan untuk menandai bahwa ular harus bertambah panjang.

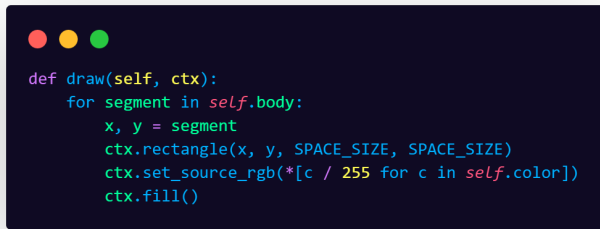
- `self.growing = True`: Mengatur atribut `growing` pada objek ular menjadi `True`.
- Saat metode `move` dijalankan setelah `grow`, atribut `growing` ini memastikan bahwa ekor ular tidak dihapus, sehingga tubuh ular bertambah panjang.



```
def check_collision(self, other_snake):  
    head = self.body[0]  
    # collision dengan tepi  
    if head[0] < 0 or head[0] >= GAME_WIDTH or head[1] < 0 or head[1] >= GAME_HEIGHT:  
        return True  
    # collision dengan badan  
    if head in self.body[1:]:  
        return True  
    # collision dengan ular lain  
    if head in other_snake.body:  
        return True  
    return False
```

Kode di atas adalah metode `check_collision` dalam kelas `Snake`, yang digunakan untuk memeriksa apakah ular mengalami tabrakan dengan elemen lain di dalam permainan.

- `head = self.body[0]`: Menyimpan posisi kepala ular yang ada di indeks pertama tubuh ular.
- `if head[0] < 0 or head[0] >= GAME_WIDTH or head[1] < 0 or head[1] >= GAME_HEIGHT`:: Memeriksa apakah kepala ular berada di luar batas permainan (dalam hal ini, batas canvas game).
- `if head in self.body[1:]`:: Memeriksa apakah kepala ular bertabrakan dengan tubuhnya sendiri. Jika kepala ular ada di dalam tubuh selain bagian pertama, berarti terjadi tabrakan.
- `if head in other_snake.body`:: Memeriksa apakah kepala ular bertabrakan dengan ular lain.
- `return True`: Jika salah satu kondisi tabrakan terpenuhi, fungsi akan mengembalikan `True`, menandakan ada tabrakan.
- `return False`: Jika tidak ada tabrakan yang terjadi, fungsi mengembalikan `False`.



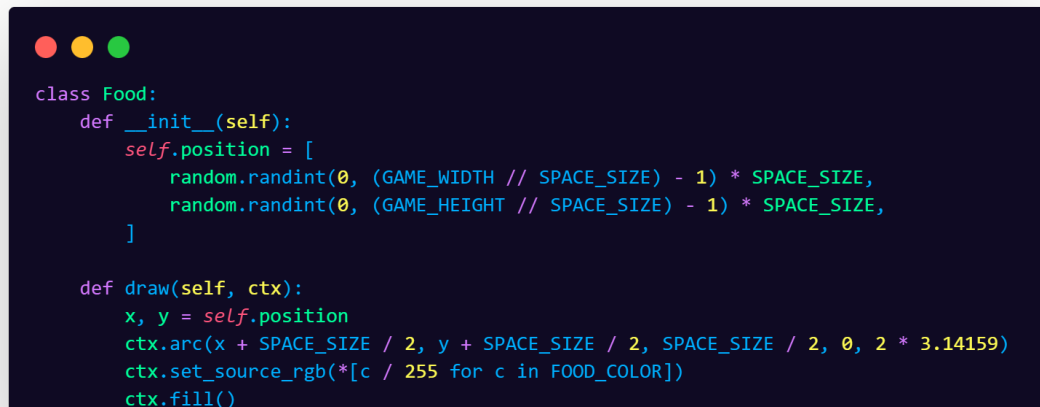
```

def draw(self, ctx):
    for segment in self.body:
        x, y = segment
        ctx.rectangle(x, y, SPACE_SIZE, SPACE_SIZE)
        ctx.set_source_rgb(*[c / 255 for c in self.color])
        ctx.fill()

```

Kode di atas adalah metode draw dalam kelas Snake, yang digunakan untuk menggambar tubuh ular pada layar menggunakan objek ctx dari pustaka Cairo.

- `for segment in self.body::` Melakukan iterasi pada setiap segmen tubuh ular yang ada dalam list `self.body`.
- `x, y = segment:` Mengambil posisi x dan y dari setiap segmen tubuh ular.
- `ctx.rectangle(x, y, SPACE_SIZE, SPACE_SIZE):` Menggambar sebuah persegi panjang untuk setiap segmen tubuh ular, dengan posisi x dan y serta ukuran yang ditentukan oleh `SPACE_SIZE` (ukuran setiap bagian tubuh ular).
- `ctx.set_source_rgb(*[c / 255 for c in self.color]):` Menetapkan warna untuk menggambar segmen tubuh ular dengan mengonversi nilai RGB (dari 0–255) menjadi nilai antara 0–1 yang diperlukan oleh Cairo.
- `ctx.fill():` Mengisi persegi panjang yang telah digambar dengan warna yang telah ditentukan.



```

class Food:
    def __init__(self):
        self.position = [
            random.randint(0, (GAME_WIDTH // SPACE_SIZE) - 1) * SPACE_SIZE,
            random.randint(0, (GAME_HEIGHT // SPACE_SIZE) - 1) * SPACE_SIZE,
        ]

    def draw(self, ctx):
        x, y = self.position
        ctx.arc(x + SPACE_SIZE / 2, y + SPACE_SIZE / 2, SPACE_SIZE / 2, 0, 2 * 3.14159)
        ctx.set_source_rgb(*[c / 255 for c in FOOD_COLOR])
        ctx.fill()

```

Kode di atas adalah implementasi dari kelas Food, yang bertugas untuk mengelola posisi dan menggambar makanan dalam permainan.

- `__init__(self):` Konstruktor untuk kelas Food, yang menetapkan posisi makanan secara acak di dalam area permainan.
- - `random.randint(0, (GAME_WIDTH // SPACE_SIZE) - 1) * SPACE_SIZE:` Menentukan posisi x makanan secara acak dalam rentang yang sesuai dengan lebar layar, dengan mempertimbangkan ukuran kotak (`SPACE_SIZE`).
 - `random.randint(0, (GAME_HEIGHT // SPACE_SIZE) - 1) * SPACE_SIZE:` Menentukan posisi y makanan secara acak dalam rentang yang sesuai dengan tinggi layar, dengan mempertimbangkan ukuran kotak (`SPACE_SIZE`).
 -
- `draw(self, ctx):` Metode untuk menggambar makanan pada layar menggunakan objek ctx dari pustaka Cairo.

- o `x, y = self.position`: Mengambil posisi x dan y makanan.
 - o `ctx.arc(x + SPACE_SIZE / 2, y + SPACE_SIZE / 2, SPACE_SIZE / 2, 0, 2 * 3.14159)`: Menggambar lingkaran untuk mewakili makanan, dengan posisi pusat di tengah kotak makanan dan jari-jari setengah dari ukuran kotak (`SPACE_SIZE / 2`).
 - o `ctx.set_source_rgb(*[c / 255 for c in FOOD_COLOR])`: Menetapkan warna makanan menggunakan nilai RGB, yang dikonversi ke dalam rentang 0–1 untuk Cairo.
 - o `ctx.fill()`: Mengisi lingkaran dengan warna yang telah ditetapkan.

```
def main():
    pygame.init()
    screen = pygame.display.set_mode((GAME_WIDTH, GAME_HEIGHT))
    pygame.display.set_caption("Snake PvP")
    clock = pygame.time.Clock()

    def reset_game():
        return Snake(SNAKE_COLORS[0], GAME_WIDTH // 4, GAME_HEIGHT // 2), \
            Snake(SNAKE_COLORS[1], 3 * GAME_WIDTH // 4, GAME_HEIGHT // 2), \
            Food(), True, None

    running = True
    snake1, snake2, food, paused, winner = reset_game()

    surface = cairo.ImageSurface(cairo.FORMAT_ARGB32, GAME_WIDTH, GAME_HEIGHT)
    ctx = cairo.Context(surface)

    while running:
        for event in pygame.event.get():
            if event.type == pygame.QUIT:
                running = False

            if event.type == pygame.KEYDOWN:
                if paused and event.key == pygame.K_SPACE:
                    if winner:
                        snake1, snake2, food, paused, winner = reset_game()
                    else:
                        paused = False
```

Kode di atas menunjukkan bagian awal dari logika permainan, yang mencakup inisialisasi permainan, pengelolaan input, dan pengaturan ulang permainan.

- `pygame.init()`: Menginisialisasi semua modul yang diperlukan oleh Pygame, seperti pengelolaan window, input, dan grafika.
- `screen = pygame.display.set_mode((GAME_WIDTH, GAME_HEIGHT))`: Membuat layar permainan dengan ukuran yang sudah ditentukan oleh `GAME_WIDTH` dan `GAME_HEIGHT`.
- `pygame.display.set_caption("Snake PvP")`: Memberikan judul pada jendela permainan.
- `clock = pygame.time.Clock()`: Membuat objek clock untuk mengatur kecepatan frame per detik (FPS) permainan.
- `reset_game()`: Fungsi ini akan mengembalikan kondisi permainan ke awal dengan memulai posisi ular pertama di bagian kiri layar dan ular kedua di bagian kanan layar. Juga, memulai makanan baru, mengatur permainan dalam keadaan dijeda (`paused = True`), dan tidak ada pemenang awal (`winner = None`).
- `snake1, snake2, food, paused, winner = reset_game()`: Menginisialisasi objek ular, makanan, status permainan, dan status pemenang.
- `surface = cairo.ImageSurface(cairo.FORMAT_ARGB32, GAME_WIDTH, GAME_HEIGHT)`: Membuat objek permukaan gambar menggunakan Cairo, dengan ukuran sesuai lebar dan tinggi game.
- `ctx = cairo.Context(surface)`: Membuat objek konteks untuk menggambar pada

permukaan gambar yang telah dibuat.

- while running:: Memulai loop utama permainan yang akan terus berjalan selama nilai running adalah True.
 - for event in pygame.event.get():: Memproses setiap event (misalnya input dari keyboard atau mouse).
 - if event.type == pygame.QUIT:: Jika event QUIT terdeteksi (misalnya ketika jendela permainan ditutup), maka running diubah menjadi False untuk menghentikan permainan.
 - if event.type == pygame.KEYDOWN:: Mengecek jika ada tombol yang ditekan.
 - if paused and event.key == pygame.K_SPACE:: Jika permainan sedang dijeda dan tombol spasi ditekan, maka:
 - Jika ada pemenang, maka permainan akan di-reset menggunakan reset_game().
 - Jika tidak ada pemenang, permainan akan dilanjutkan dengan mengubah status paused menjadi False.

```
# Player 1
if event.key == pygame.K_w and snake1.direction != "down":
    snake1.direction = "up"
if event.key == pygame.K_s and snake1.direction != "up":
    snake1.direction = "down"
if event.key == pygame.K_a and snake1.direction != "right":
    snake1.direction = "left"
if event.key == pygame.K_d and snake1.direction != "left":
    snake1.direction = "right"
# Player 2
if event.key == pygame.K_UP and snake2.direction != "down":
    snake2.direction = "up"
if event.key == pygame.K_DOWN and snake2.direction != "up":
    snake2.direction = "down"
if event.key == pygame.K_LEFT and snake2.direction != "right":
    snake2.direction = "left"
if event.key == pygame.K_RIGHT and snake2.direction != "left":
    snake2.direction = "right"
```

Kode di atas menangani input dari keyboard untuk mengubah arah pergerakan ular sesuai dengan tombol yang ditekan oleh masing-masing pemain. Kode ini memastikan bahwa ular tidak bisa berbalik arah langsung ke arah yang berlawanan dengan arah pergerakan sebelumnya.

- Untuk Player 1 (menggunakan tombol W, A, S, D):
 - if event.key == pygame.K_w and snake1.direction != "down": Jika tombol W ditekan dan ular 1 tidak bergerak ke bawah, maka ular 1 akan bergerak ke atas ("up").
 - if event.key == pygame.K_s and snake1.direction != "up": Jika tombol S ditekan dan ular 1 tidak bergerak ke atas, maka ular 1 akan bergerak ke bawah ("down").
 - if event.key == pygame.K_a and snake1.direction != "right": Jika tombol A ditekan dan ular 1 tidak bergerak ke kanan, maka ular 1 akan bergerak ke kiri ("left").
 - if event.key == pygame.K_d and snake1.direction != "left": Jika tombol D ditekan dan ular 1 tidak bergerak ke kiri, maka ular 1 akan bergerak ke kanan ("right").
- Untuk Player 2 (menggunakan tombol panah atas, bawah, kiri, kanan):
 - if event.key == pygame.K_UP and snake2.direction != "down": Jika tombol

panah ↑ ditekan dan ular 2 tidak bergerak ke bawah, maka ular 2 akan bergerak ke atas ("up").

- if event.key == pygame.K_DOWN and snake2.direction != "up": Jika tombol panah ↓ ditekan dan ular 2 tidak bergerak ke atas, maka ular 2 akan bergerak ke bawah ("down").
- if event.key == pygame.K_LEFT and snake2.direction != "right": Jika tombol panah ← ditekan dan ular 2 tidak bergerak ke kanan, maka ular 2 akan bergerak ke kiri ("left").
- if event.key == pygame.K_RIGHT and snake2.direction != "left": Jika tombol panah → ditekan dan ular 2 tidak bergerak ke kiri, maka ular 2 akan bergerak ke kanan ("right").

```
if paused:
    screen.fill(BACKGROUND_COLOR)
    font = pygame.font.SysFont("calibri", 35)
    text = font.render("Tekan SPASI Untuk Start" if not winner else f"{winner} Wins! Tekan Spasi untuk mulai ulang", True, (255, 255, 255))
    screen.blit(text, (GAME_WIDTH // 2 - text.get_width() // 2, GAME_HEIGHT // 2 - text.get_height() // 2))
    pygame.display.flip()
    clock.tick(60)
    continue

snake1.move()
snake2.move()

if snake1.body[0] == food.position:
    snake1.grow()
    food = Food()
if snake2.body[0] == food.position:
    snake2.grow()
    food = Food()

if snake1.check_collision(snake2):
    winner = "Player 2"
    paused = True
if snake2.check_collision(snake1):
    winner = "Player 1"
    paused = True
```

kode di atas menangani logika untuk sementara menghentikan permainan (pause) dan melakukan pengecekan apakah ada kemenangan atau tidak. Berikut penjelasan untuk setiap bagian:

- if paused:
 - Jika permainan dalam kondisi pause, kode di bawah ini akan dijalankan.
- screen.fill(BACKGROUND_COLOR)
 - Mengisi layar dengan warna latar belakang yang telah ditentukan sebelumnya.
- font = pygame.font.SysFont("calibri", 35)
 - Membuat objek font dengan tipe Calibri dan ukuran 35 untuk menampilkan teks di layar.
- text = font.render("Tekan SPASI Untuk Start" if not winner else f"{winner} Wins! Tekan Spasi untuk mulai ulang", True, (255, 255, 255))
Menentukan teks yang akan ditampilkan di layar:
 - Jika belum ada pemenang (not winner), maka teks yang ditampilkan adalah "Tekan SPASI Untuk Start".
 - Jika ada pemenang (winner), teks yang ditampilkan adalah nama pemenang dan pesan untuk mulai ulang.
- screen.blit(text, (GAME_WIDTH // 2 - text.get_width() // 2, GAME_HEIGHT // 2 - text.get_height() // 2))
 - Menampilkan teks yang telah dirender pada posisi tengah layar.
- pygame.display.flip()
 - Menampilkan perubahan yang telah dilakukan pada layar.
- clock.tick(60)
 - Mengatur agar game berjalan dengan frame rate 60 frame per detik saat game dalam kondisi pause.
- continue
 - Melanjutkan ke iterasi berikutnya dari loop while, sehingga permainan tidak akan dilanjutkan lebih jauh jika sedang pause.

- `snake1.move()` dan `snake2.move()`
 - Jika permainan tidak dalam keadaan pause, maka kedua ular akan bergerak sesuai dengan arah yang sudah ditentukan.
- `if snake1.body[0] == food.position:`
 - Mengecek apakah kepala ular 1 (segment pertama) bertabrakan dengan posisi makanan. Jika ya, maka ular 1 akan bertumbuh dan makanan akan di-reset ke posisi baru.
- `if snake2.body[0] == food.position:`
 - Mengecek apakah kepala ular 2 (segment pertama) bertabrakan dengan posisi makanan. Jika ya, maka ular 2 akan bertumbuh dan makanan akan di-reset ke posisi baru.
- `if snake1.check_collision(snake2):`
 - Mengecek apakah ular 1 bertabrakan dengan ular 2. Jika ya, maka permainan akan dihentikan sementara dan pemenang ditentukan sebagai "Player 2".
- `if snake2.check_collision(snake1):`
 - Mengecek apakah ular 2 bertabrakan dengan ular 1. Jika ya, maka permainan akan dihentikan sementara dan pemenang ditentukan sebagai "Player 1".

```

ctx.rectangle(0, 0, GAME_WIDTH, GAME_HEIGHT)
ctx.set_source_rgb(*[c / 255 for c in BACKGROUND_COLOR])
ctx.fill()

food.draw(ctx)
snake1.draw(ctx)
snake2.draw(ctx)

# Convert ARGB32 ke RGB24 karena ada masalah di pewarnaan
buffer = surface.get_data()
img_array = np.frombuffer(buffer, np.uint8).reshape((GAME_HEIGHT, GAME_WIDTH, 4))
img_rgb = img_array[:, :, [2, 1, 0]] # Swap channel R dan B

```

Kode di atas bertanggung jawab untuk menggambar elemen-elemen game ke dalam permukaan (surface) menggunakan pustaka **Cairo** dan kemudian menampilkan gambar tersebut di layar dengan **pygame**. Berikut penjelasan masing-masing bagian:

1. `ctx.rectangle(0, 0, GAME_WIDTH, GAME_HEIGHT)`
 - Menggambar sebuah persegi panjang yang mencakup seluruh area permainan (seluruh layar game), dimulai dari titik (0,0) dengan lebar `GAME_WIDTH` dan tinggi `GAME_HEIGHT`.
2. `ctx.set_source_rgb(*[c / 255 for c in BACKGROUND_COLOR])`
 - Menentukan warna untuk persegi panjang yang telah digambar. Warna ini diambil dari variabel `BACKGROUND_COLOR`, dan karena Cairo menggunakan format RGB (0-1), warna tersebut dibagi dengan 255 untuk mengubahnya ke format yang sesuai.
3. `ctx.fill()`
 - Mengisi persegi panjang yang telah digambar dengan warna latar belakang yang telah ditentukan.
4. `food.draw(ctx)`
 - Menggambar objek makanan di layar menggunakan metode `draw` dari kelas `Food`.
5. `snake1.draw(ctx)` dan `snake2.draw(ctx)`
 - Menggambar kedua ular (player 1 dan player 2) di layar menggunakan metode `draw` dari kelas `Snake` untuk masing-masing ular.
6. `buffer = surface.get_data()`
 - Mengambil data gambar (pixel) dari permukaan (surface) yang telah digambar

dengan Cairo dan menyimpannya ke dalam variabel buffer.

7. `img_array = np.frombuffer(buffer, np.uint8).reshape((GAME_HEIGHT, GAME_WIDTH, 4))`
 - Mengubah data gambar (dalam bentuk buffer) menjadi array NumPy dengan tipe data `uint8` (angka bulat antara 0 dan 255). Kemudian array ini diubah bentuknya sesuai dengan dimensi gambar, yakni `(GAME_HEIGHT, GAME_WIDTH, 4)`, di mana 4 mewakili empat channel (RGBA).
8. `img_rgb = img_array[:, :, [2, 1, 0]]`
 - Mengubah urutan channel warna dari RGBA (Red, Green, Blue, Alpha) menjadi RGB (Red, Green, Blue) dengan menukar posisi channel merah (R) dan biru (B), karena ada masalah dengan urutan warna pada Cairo.

```
pygame_surface = pygame.image.frombuffer(img_rgb.tobytes(), (GAME_WIDTH, GAME_HEIGHT), "RGB")
screen.blit(pygame_surface, (0, 0))
pygame.display.flip()
fps = SPEED
clock.tick(fps)
```

Dengan kode diatas, tampilan game diperbarui pada kecepatan yang sesuai, dan seluruh elemen game (seperti ular dan makanan) digambar dan ditampilkan di layar.

```
pygame_surface = pygame.image.frombuffer(img_rgb.tobytes(), (GAME_WIDTH, GAME_HEIGHT), "RGB")
```

- Mengonversi data gambar yang ada dalam `img_rgb` (array NumPy) menjadi format yang bisa digunakan oleh pygame.
- Fungsi `tobytes()` mengubah array NumPy menjadi byte stream, dan `pygame.image.frombuffer()` mengubah byte stream tersebut menjadi objek gambar (`pygame.Surface`) dengan dimensi `(GAME_WIDTH, GAME_HEIGHT)` dan format warna "RGB".

```
screen.blit(pygame_surface, (0, 0))
```

- Menampilkan (blit) gambar yang telah diproses (dalam bentuk objek `pygame_surface`) ke layar `screen`. Posisi `(0, 0)` menunjukkan bahwa gambar akan ditempatkan di pojok kiri atas layar.

```
pygame.display.flip()
```

- Menyegarkan layar untuk menampilkan perubahan terbaru yang telah dilakukan pada permukaan (surface). Fungsi ini penting untuk memperbarui tampilan grafis agar perubahan yang sudah digambar dapat dilihat oleh pemain.

```
fps = SPEED
```

- Mengatur frame rate atau kecepatan permainan, yang didasarkan pada nilai variabel `SPEED`. Dalam hal ini, `SPEED` adalah kecepatan permainan yang sudah ditentukan sebelumnya (5 dalam kasus ini).

```
clock.tick(fps)
```

- Mengatur jumlah frame per detik (FPS) yang diinginkan dalam permainan. Fungsi `clock.tick(fps)` memastikan bahwa game berjalan pada kecepatan yang diinginkan, diatur oleh nilai `fps` yang sudah didefinisikan sebagai `SPEED`.

Daftar Pustaka

Angel, E., & Shreiner, D. (2011). Interactive Computer Graphics: A Top-Down Approach with WebGL. Addison-Wesley.

Van Rossum, G., & Drake, F. L. (2009). Python 3 Reference Manual. CreateSpace Independent Publishing Platform.

Pygame Community. (2023). Pygame Documentation. Diakses dari <https://www.pygame.org/docs/>.

Cairo Graphics. (2023). Cairo Documentation. Diakses dari <https://www.cairographics.org/documentation/>.