Penjelasan Kode Program Simulasi Game Invader dengan Ursina

Kode berikut adalah implementasi sederhana dari permainan invader menggunakan pustaka `Ursina` di Python. Game ini melibatkan tank, musuh, dan rudal. Pemain dapat menggerakkan tank dan menembakkan rudal untuk menghancurkan musuh.

```
from ursina import *

app = Ursina()
```

Listing 1: Kode Simulasi Game Invader

Pustaka `ursina` digunakan untuk membuat game 3D dengan mudah. `app = Ursina()` membuat aplikasi Ursina yang akan menjalankan game kita.

Gambaran Umum

- **tank:** Merujuk pada objek tank pemain yang dikendalikan dalam permainan.
- **enemies:** Daftar yang berisi objek-objek musuh yang diinisialisasi dalam pola grid.
- **rudals:** Daftar yang akan menyimpan objek-objek rudal yang ditembakkan oleh tank.

```
class Tank(Entity):
    def __init__(self):
        super().__init__(
            model='cube',
            color=color.blue,
            scale=(0.5, 0.1, 0.5),
            position=(0, -3.5)
        )
        self.speed = 5

    def update(self):
        if held_keys['d'] and self.x < 3.5:
            self.x += self.speed * time.dt
        if held_keys['a'] and self.x > -3.5:
            self.x -= self.speed * time.dt
```

`Tank` adalah kelas yang mewakili tank pemain. Metode `update` digunakan untuk menggerakkan tank berdasarkan input tombol dari pemain.

```
-------------------------
def __init__(self):
-------------------------
Ini adalah definisi dari metode konstruktor untuk kelas Tank. Metode konstruktor digunakan untuk
menginisialisasi objek saat pertama kali dibuat.
Konstruktor dalam pemrograman adalah metode khusus yang digunakan untuk menginisialisasi objek dari suatu kelas.
Konstruktor dipanggil secara otomatis untuk melakukan inisialisasi awal objek tersebut.

-------------------------
super().__init__(
-------------------------
super() digunakan untuk mengakses metode dari kelas induk dari Tank, yaitu kelas Entity (diasumsikan sebagai
kelas dasar dalam Ursina untuk semua entitas). Dalam konteks ini, super().__init__() digunakan
untuk memanggil konstruktor dari kelas Entity, yang merupakan kelas dasar untuk entitas dalam game.

-------------------------
model='cube',
-------------------------
Parameter model menentukan bentuk atau model visual dari entitas. Di sini, tank direpresentasikan sebagai kubus.

-------------------------
color=color.blue,
-------------------------
Parameter color menentukan warna dari entitas. color.blue mengatur warna tank menjadi biru.

-------------------------
scale=(0.5, 0.1, 0.5),
-------------------------
Parameter scale menentukan skala atau ukuran entitas dalam tiga dimensi (x, y, z). Di sini, tank diperbesar
```

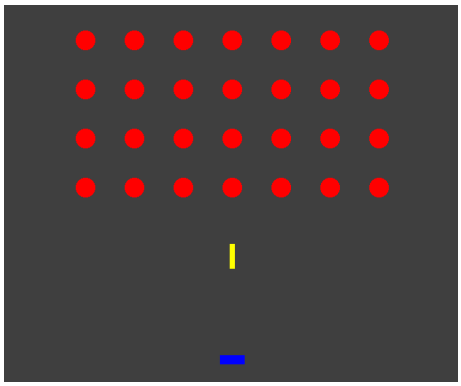
lima kali lipat dari ukuran standarnya dalam sumbu x dan z, dan 0.1 kali lipat dalam sumbu y.

---
position=(0, -3.5)

Parameter position menentukan posisi awal entitas dalam koordinat dunia. Koordinat (0, -3.5) menempatkan tank di tengah layar (sumbu x) dan sedikit di bagian bawah (sumbu y).

---
self.speed = 5

Ini adalah variabel anggota (attribute) speed dari objek Tank. Variabel ini digunakan untuk menentukan kecepatan pergerakan tank. Di sini, tank diinisialisasi dengan kecepatan 5.



```python
class Rudal(Entity):
    def __init__(self, position):
        super().__init__(
            model='quad',
            color=color.yellow,
            scale=(0.1, 0.5),
            position=position
        )
        self.speed = 4

    def update(self):
        self.y += self.speed * time.dt
        if self.y > 5:
            destroy(self)
            if self in rudals:
                rudals.remove(self)
```

`Rudal` adalah kelas yang mewakili rudal yang ditembakkan oleh tank. Rudal bergerak ke atas setiap frame dan dihancurkan ketika keluar dari layar.

---
model='quad'

Quad adalah singkatan dari quadrilateral, yang berarti bentuk dengan empat sisi. Dalam konteks Ursina, 'quad' sering digunakan untuk membuat objek yang terlihat seperti persegi panjang (rectangle)

---
update(self)
---
metode yang umum digunakan dalam game development untuk mengatur perubahan status atau perilaku objek dari waktu ke waktu.

---
self.y += self.speed * time.dt

Baris ini bertanggung jawab untuk menggerakkan objek (self) ke atas di sumbu y, berdasarkan kecepatan (self.speed) yang telah ditentukan untuk objek tersebut. time.dt adalah delta time, yaitu interval waktu antara setiap frame dalam permainan. Mengalikan kecepatan dengan time.dt memastikan bahwa pergerakan objek lebih lancar dan tidak tergantung pada kecepatas komputer atau perangkat.

---
if self.y > 5:

Ini adalah kondisi yang memeriksa apakah objek telah mencapai posisi y lebih dari 5. Dalam konteks permainan kita, ini mungkin berarti objek Rudal telah mencapai batas atas layar atau area tertentu dalam permainan.

```
----------------------
destroy(self)
----------------------
Fungsi destroy() digunakan untuk menghapus objek dari layar atau dunia permainan. Dalam kasus ini,
jika self.y > 5, objek Rudal akan dihapus dari permainan.

--------------------------------------
if self in rudals: rudals.remove(self)
--------------------------------------
Setelah menghancurkan objek Rudal, baris ini memeriksa apakah objek self masih ada dalam daftar rudals
(yang mungkin berisi semua rudal yang sedang aktif dalam permainan). Jika ya, maka objek self akan dihapus
dari daftar rudals.
```

```python
class Enemy(Entity):
    def __init__(self, position):
        super().__init__(
            model='circle',
            color=color.red,
            scale=(0.4, 0.4),
            position=position
        )
```

`Enemy` adalah kelas yang mewakili musuh. Musuh diinisialisasi sebagai lingkaran berwarna merah.

```python
tank = Tank()

enemies = [Enemy((i, j)) for i in range(-3, 4) for j in range(4)]
rudals = []
```

Kita membuat instance `Tank` dan daftar musuh `enemies`. Daftar `rudals` berisi rudal-rudal aktif.

```
------------------------
tank = Tank()
------------------------
Membuat sebuah instance dari kelas Tank. Instance Tank yang baru saja dibuat ini kemudian disimpan dalam
variabel tank. Variabel ini sekarang merujuk pada entitas tank di dalam permainan.
Ketika Tank() dipanggil, metode konstruktor __init__ dari kelas Tank akan dieksekusi. Tank akan diinisialisasi
dengan model cube, berwarna biru, dengan ukuran skala (0.5, 0.1, 0.5), dan diposisikan pada koordinat (0, -3.5).

-----------------------------------------------------------------------
enemies = [Enemy((i, j)) for i in range(-3, 4) for j in range(4)]
-----------------------------------------------------------------------
----------------------
List Comprehension:
----------------------
Ini adalah cara Python yang ringkas dan efisien untuk membuat daftar (list)
menggunakan loop dalam satu baris kode.

----------------------
Range dalam Loop:
----------------------
range(-3, 4): Loop ini menghasilkan nilai mulai dari -3 hingga 3 (termasuk), menghasilkan
total 7 nilai (-3, -2, -1, 0, 1, 2, 3).

----------------------
range(4):
----------------------
Loop ini menghasilkan nilai mulai dari 0 hingga 3 (termasuk), menghasilkan total 4 nilai (0, 1, 2, 3).

----------------------
Posisi Musuh:
----------------------
Koordinat x: Nilai i dari -3 hingga 3 menentukan posisi horizontal dari musuh.
Koordinat y: Nilai j dari 0 hingga 3 menentukan posisi vertikal dari musuh.

--------------------------
Inisialisasi Objek Enemy:
--------------------------
Setiap kombinasi (i, j) digunakan untuk menginisialisasi sebuah objek Enemy, yang kemudian ditempatkan pada
posisi (i, j).

----------------------
Daftar Enemies:
----------------------
```

```
Semua objek Enemy yang diinisialisasi dengan posisi yang ditentukan disimpan dalam daftar enemies.
Total Musuh: Karena kita memiliki 7 nilai untuk i dan 4 nilai untuk j, total musuh yang diinisialisasi adalah
7×4=28 musuh, yang tersebar di posisi grid dari (-3,0) hingga (3,3).
```

```
game_over = False
```

Variabel `game_over` melacak apakah permainan sudah selesai atau belum.

```
def update():
    global enemies, rudals, game_over

    if game_over:
        return

    enemies_to_remove = []
    rudals_to_remove = []

    for enemy in enemies:
        if distance(enemy, tank) < 1:
            quit()

        for rudal in rudals:
            if distance(rudal, enemy) < 0.4:
                rudals_to_remove.append(rudal)
                enemies_to_remove.append(enemy)
                break

    for rudal in rudals_to_remove:
        if rudal in rudals:
            destroy(rudal)
            rudals.remove(rudal)

    for enemy in enemies_to_remove:
        if enemy in enemies:
            destroy(enemy)
            enemies.remove(enemy)

    if len(enemies) == 0:
        game_over = True
        show_game_over()
```

Fungsi `update` adalah fungsi global yang memperbarui status permainan setiap frame. Jika tank dan musuh bertabrakan, permainan berakhir. Jika rudal dan musuh bertabrakan, keduanya dihancurkan. Permainan berakhir ketika semua musuh hancur.

```
--------------------------------------
enemies_to_remove = []
rudals_to_remove = []
--------------------------------------
Daftar ini akan digunakan untuk melacak musuh dan rudal yang perlu dihapus dari permainan.
```

```
def input(key):
    if key == 'space' and not game_over:
        rudal = Rudal((tank.x, tank.y + 0.5))
        rudals.append(rudal)
```

Fungsi `input` menangani input dari pemain. Ketika tombol 'space' ditekan, sebuah rudal ditembakkan jika permainan belum selesai.

```
--------------------------------------
input(key)
--------------------------------------
adalah fungsi khusus di Ursina yang dipanggil secara otomatis setiap kali ada input dari
pengguna, seperti menekan tombol keyboard atau klik mouse.

Parameter key:
Parameter key mewakili tombol yang ditekan oleh pengguna pada saat input terjadi.

Pengecekan Input:
--------------------------------------
if key == 'space' and not game_over::
--------------------------------------
```

```
Baris ini memeriksa apakah tombol yang ditekan adalah spasi ('space')
dan apakah permainan belum berakhir (not game_over).
Jika kedua kondisi ini terpenuhi, kode di dalam blok if akan dieksekusi.

Membuat Rudal Baru (Rudal):
--------------------------------------
rudal = Rudal((tank.x, tank.y + 0.5)):
--------------------------------------
Baris ini membuat sebuah objek Rudal baru dengan menginisialisasi
posisi baru untuk rudal tersebut.
Posisi x diambil dari posisi tank (tank.x).
Posisi y dinaikkan sebesar 0.5 dari posisi tank (tank.y + 0.5).
Objek Rudal yang baru dibuat kemudian disimpan dalam list rudals menggunakan perintah rudals.append(rudal).
```

```
def show_game_over():
    game_over_text = Text(
        text='Game Over',
        origin=(0, 0),
        scale=3,
        background=True
    )
    game_over_text.position = (0, 0)
```

Fungsi `show_game_over` menampilkan teks "Game Over" di tengah layar ketika permainan berakhir.

```
app.run()
```

`app.run()` memulai loop utama dari game, menjalankan semua logika permainan dan pembaruan tampilan secara terus-menerus.

Ukuran layar

```
(-3.5, 5)              (3.5, 5)
+----------------------+
|                      |
|                      |
|                      |
|                      |
|                      |
|                      |
|                      |
|                      |
|                      |
|                      |
|                      |
|                      |
|                      |
|                      |
|                      |
+----------------------+
(-3.5, -3.5)           (3.5, -3.5)
```

Analisis Batas Layar

- Tank bergerak di sepanjang sumbu x antara -3.5 dan 3.5.
- Musuh ditempatkan dalam rentang posisi x dari -3 hingga 3 dan dalam rentang posisi y dari 0 hingga 3.
- Rudal bergerak ke atas di sepanjang sumbu y, dan dihancurkan ketika posisinya melebihi $y = 5$.

Fungsi Distance

Cara Kerja `distance` di Ursina

- **Entitas atau Koordinat:** Fungsi `distance` dapat bekerja dengan dua jenis input dalam Ursina:
  - *Entitas:* Objek-objek yang memiliki atribut `position`, yang menyimpan posisi entitas dalam ruang tiga dimensi.
  - *Vektor Posisi:* Vektor yang merepresentasikan koordinat posisi langsung dalam ruang 3D.
- **Jarak Euclidean:** Fungsi ini biasanya menghitung jarak Euclidean, yaitu jarak garis lurus antara dua titik dalam ruang tiga dimensi. Jarak Euclidean antara dua titik (x_1, y_1, z_1) dan (x_2, y_2, z_2) didefinisikan sebagai:

$$\text{distance} = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2}$$

  Di mana:
  - (x_1, y_1, z_1) adalah koordinat titik pertama.
  - (x_2, y_2, z_2) adalah koordinat titik kedua.

Cara Kerja `Entity` di Ursina

Ketika Anda membuat sebuah `Entity` di Ursina, Anda dapat menentukan berbagai properti dan perilaku. Berikut adalah beberapa atribut dan konsep kunci yang terkait dengan `Entity`:

Atribut Utama

- `model`: Menentukan bentuk geometris dari entitas. Bisa berupa model 3D standar atau model yang diimpor.
- `color`: Menentukan warna dari entitas. Bisa berupa warna solid atau tekstur.
- `position`: Menentukan posisi entitas dalam ruang 3D.
- `scale`: Mengatur ukuran entitas. Bisa skala seragam atau tidak seragam.
- `rotation`: Menentukan rotasi entitas di dalam ruang 3D.
- `texture`: Menentukan tekstur yang diterapkan pada model entitas.

Metode Utama

- `__init__()`: Konstruktor yang digunakan untuk menginisialisasi sebuah entitas dengan properti-properti tertentu.
- `update()`: Fungsi ini dipanggil setiap frame. Digunakan untuk mengupdate logika permainan atau perilaku entitas.
- `input()`: Digunakan untuk menangani input dari pengguna, seperti menekan tombol atau gerakan mouse.