**Laporan Program Python**

**Praktikum Lab – Class dan Turtle**

**Mata Kuliah:**

Pemrograman Berorientasi Objek

## Oleh:

Gerry Moeis Mahardika Dwi Putra

23091397164

2023E

[https://github.com/gerrymoeis/pbo-](https://github.com/gerrymoeis/pbo-1)2



# PROGRAM STUDI MANAJEMEN INFORMATIKA FAKULTAS VOKASI

**UNIVERSITAS NEGERI SURABAYA 2024**

## Laporan Tugas 1: Membuat Game Uler-uleran

|  |  |
| --- | --- |
| **Langkah** | **Praktikum** |
| **1.** | import pygame, sys, random, time  pygame.init()  Disini saya import modul pygame yang telah diinstall, juga modul bawaan seperti sys, random, dan time.  Lalu lakukan init. Untuk menginisiasi semua komponen atau fungsi dalam pygame.  class GameObject:      def \_\_init\_\_(self, color, position, size):          self.color = color          self.position = position          self.size = size  Selanjutnya, saya buat class GameObject. Sebagai blueprint parent pembuatan object maupun class inheritance lainnya.  Atribut yang ditambahkan adalah parameter umum. |
|  | def draw(self, game\_window):          pygame.draw.rect(game\_window, self.color, pygame.Rect(self.position[0], self.position[1], self.size, self.size))    Dalam class GameObject kita buat metode utama yaitu draw. Yang mana bertujuan untuk menggambar objek yang diharapkan sesuai atributnya. |

|  |  |
| --- | --- |
| **2.** | class Snake(GameObject):      def \_\_init\_\_(self, color, position, size, speed=10):          super().\_\_init\_\_(color, position, size)          self.body = [list(position), [position[0] - size, position[1]], [position[0] - size, position[1]]]          self.speed = speed          self.direction = 'RIGHT'          self.change\_to = self.direction  Sekarang kita buat Inheritance dari GameObject yaitu Snake. Objek Snake ini akan memiliki atribut dari GameObject juga tambahan beberapa atribut yang spesifik untuk kebutuhannya.  def change\_direction(self, direction):          if direction == 'UP' and self.direction != 'DOWN':              self.change\_to = 'UP'          if direction == 'DOWN' and self.direction != 'UP':              self.change\_to = 'DOWN'          if direction == 'LEFT' and self.direction != 'RIGHT':              self.change\_to = 'LEFT'          if direction == 'RIGHT' and self.direction != 'LEFT':              self.change\_to = 'RIGHT'  Lalu method selanjutnya pada class Snake adalah change\_direction, yang bertujuan menyimpan data variabel arah untuk digunakan nantinya.  def move(self):          self.direction = self.change\_to          if self.direction == 'UP':              self.position[1] -= self.speed          if self.direction == 'DOWN':              self.position[1] += self.speed          if self.direction == 'LEFT':              self.position[0] -= self.speed          if self.direction == 'RIGHT':              self.position[0] += self.speed          self.body.insert(0, list(self.position))  Nah berikutnya ada method move. Yang mana mengatur pergerakan Snake nantinya. Arah dan pergerakan dari kode sebelumnya di terapkan pada method ini, lalu ditambah kecepatan sesuai arah gerakan.  def shrink(self):          self.body.pop()  Method shrink disini bertujuan untuk menghapus ekor snake, karena nanti saat di gambar, jika ekor tidak dihapus, maka panjang snake akan terus menambah.  def draw(self, game\_window):          for pos in self.body:              pygame.draw.rect(game\_window, self.color, pygame.Rect(pos[0], pos[1], self.size, self.size))  Nah untuk method draw pada blueprint GameObject sebelumnya, disini kita mengubahnya sesuai kebutuhan kita, yaitu dengan looping yang bertujuan menggambar masing-masing bagian Snake.  def check\_collision(self, frame\_size\_x, frame\_size\_y):          if self.position[0] < 0 or self.position[0] > frame\_size\_x - self.size or self.position[1] < 0 or self.position[1] > frame\_size\_y - self.size:              return True            for block in self.body[1:]:              if self.position[0] == block[0] and self.position[1] == block[1]:                  return True            return False  Kemudian penting kita tambahkan method check\_collision untuk melakukan cek bilamana Snake menabrak atau bersinggungan dengan GameObject lain. |
| **3.** | class Apple(GameObject):      def \_\_init\_\_(self, frame\_size\_x, frame\_size\_y, color):          self.position = [random.randrange(1, (frame\_size\_x // 10)) \* 10, random.randrange(1, (frame\_size\_y // 10)) \* 10]          super().\_\_init\_\_(color, self.position, 10)  Sekarang kita buat class Apple yang juga inheritance dari GameObject.  Untuk inisiasi atribut. Disini kita mengkustomisasi atribut position.  def respawn(self, frame\_size\_x, frame\_size\_y):          self.position = [random.randrange(1, (frame\_size\_x // 10)) \* 10, random.randrange(1, (frame\_size\_y // 10)) \* 10]  Nah method yang kita tambahkan pada class Apple ini ialah respawn. Bertujuan untuk memunculkan apple kembali, nanti untuk mechanicnya akan dijelaskan pada langkah-langkah berikutnya. |
| **4.** | class Game:      def \_\_init\_\_(self, apple\_amount):          self.frame\_size\_x = 720          self.frame\_size\_y = 480          self.game\_window = pygame.display.set\_mode((self.frame\_size\_x, self.frame\_size\_y))          pygame.display.set\_caption('Aowkwk')          self.fps\_controller = pygame.time.Clock()          self.snake = Snake(pygame.Color(0, 255, 0), [100, 50], 10)          self.score = 0          self.apples = []          for \_ in range(apple\_amount):              self.apples.append(Apple(self.frame\_size\_x, self.frame\_size\_y, pygame.Color(random.randint(0, 255), random.randint(0, 255), random.randint(0, 255))))          self.arena\_position = (50, 50)          self.arena\_size = (self.frame\_size\_x - 100, self.frame\_size\_y - 100)  Oke, class terakhir yang kita butuhkan adalah class Game. Objek ini nantinya akan berfungsi untuk mengatur keseluruhan mechanic game termasuk layar tampilan permainannya.  Terdapat banyak atribut yang didefinisikan di class ini. Mulai dari ukuran layar, window, snake, apple, hingga arena.  def draw\_arena(self):          pygame.draw.rect(self.game\_window, pygame.Color(0, 0, 0), pygame.Rect(self.arena\_position[0], self.arena\_position[1], self.arena\_size[0], self.arena\_size[1]), 2)  Pertama kita buat method draw\_arena untuk menggambar kotak arena nantinya.  def game\_over(self):          my\_font = pygame.font.SysFont('Times New Roman', 90)          game\_over\_surface = my\_font.render('YOU DIED', True, pygame.Color(255, 0, 0))          game\_over\_rect = game\_over\_surface.get\_rect()          game\_over\_rect.midtop = (self.frame\_size\_x / 2, self.frame\_size\_y / 4)          self.game\_window.fill(pygame.Color(0, 0, 0))          self.game\_window.blit(game\_over\_surface, game\_over\_rect)          pygame.display.flip()          time.sleep(3)          pygame.quit()          sys.exit()  Lalu method game\_over, yang mengatur bilamana Snake menabrak dinding atau menabrak dirinya sendiri maka ditampilkan Game Over dan di exit.  def show\_score(self):          score\_font = pygame.font.SysFont('Times New Roman', 20)          score\_surface = score\_font.render('Score : ' + str(self.score), True, pygame.Color(0, 0, 0))          score\_rect = score\_surface.get\_rect()          score\_rect.midtop = (72, 15)          self.game\_window.blit(score\_surface, score\_rect)  Selanjutnya method show\_score untuk menampilkan jumlah score yang diperoleh. Disini ditetapkan font, warna, dan format teks yang akan ditampilkan.  def run(self):          while True:              for event in pygame.event.get():                  if event.type == pygame.QUIT:                      pygame.quit()                      sys.exit()                  elif event.type == pygame.KEYDOWN:                      if event.key == pygame.K\_UP:                          self.snake.change\_direction('UP')                      if event.key == pygame.K\_DOWN:                          self.snake.change\_direction('DOWN')                      if event.key == pygame.K\_LEFT:                          self.snake.change\_direction('LEFT')                      if event.key == pygame.K\_RIGHT:                          self.snake.change\_direction('RIGHT')                      if event.key == pygame.K\_ESCAPE:                          pygame.event.post(pygame.event.Event(pygame.QUIT))                self.snake.move()              if self.snake.position in map(lambda apple: apple.position, self.apples):                  eated\_apple = [apple for apple in self.apples if apple.position == self.snake.position][0]                  eated\_apple.respawn(self.arena\_size[0], self.arena\_size[1])                  self.score += 1              else:                  self.snake.shrink()                self.game\_window.fill(pygame.Color(255, 255, 255))              self.draw\_arena()              self.snake.draw(self.game\_window)              self.show\_score()              for apple in self.apples:                  apple.draw(self.game\_window)              if self.snake.check\_collision(self.frame\_size\_x, self.arena\_size[1] + self.arena\_position[1]):                  self.game\_over()                pygame.display.update()              self.fps\_controller.tick(10)  Terakhir method paling penting yaitu run. Method yang mengatur keseluruhan jalannya permainan dan mechanicnya.  Untuk penjelasan masing-masing kode akan diterangkan lebih detail di bawah:  while True:              for event in pygame.event.get():                  if event.type == pygame.QUIT:                      pygame.quit()                      sys.exit()                  elif event.type == pygame.KEYDOWN:                      if event.key == pygame.K\_UP:                          self.snake.change\_direction('UP')                      if event.key == pygame.K\_DOWN:                          self.snake.change\_direction('DOWN')                      if event.key == pygame.K\_LEFT:                          self.snake.change\_direction('LEFT')                      if event.key == pygame.K\_RIGHT:                          self.snake.change\_direction('RIGHT')                      if event.key == pygame.K\_ESCAPE:                          pygame.event.post(pygame.event.Event(pygame.QUIT))                self.snake.move()  Tujuan kode di atas untuk menggerakkan snake berdasarkan input panah pada keyboard.  if self.snake.position in map(lambda apple: apple.position, self.apples):                  eated\_apple = [apple for apple in self.apples if apple.position == self.snake.position][0]                  eated\_apple.respawn(self.arena\_size[0], self.arena\_size[1])                  self.score += 1              else:                  self.snake.shrink()  Lalu disini kita buat kondisi yang mana untuk mengecek bilamana Snake memakan apple (posisi keduanya bersinggungan / collide). Maka score ditambah dan apple di respawn.  Tetapi bilamana snake sedang tidak memakan apple. Maka snake di shrink agar ukuran snake tetap sama.  self.game\_window.fill(pygame.Color(255, 255, 255))              self.draw\_arena()              self.snake.draw(self.game\_window)              self.show\_score()              for apple in self.apples:                  apple.draw(self.game\_window)  Kode di atas adalah kumpulan kode untuk menampilkan UI. Seperti menggambar arena, snake, apple, dsb.  if self.snake.check\_collision(self.frame\_size\_x, self.arena\_size[1] + self.arena\_position[1]):                  self.game\_over()  Nah disini kita cek terlebih dahulu soal collision snake. Bilamana snake menabrak dinding arena maka tampilkan pesan Game Over.  pygame.display.update()              self.fps\_controller.tick(10)  Terakhir, kode di atas bertujuan untuk mengatur frame pada window pygame, disitu window akan di update dan fps ditentukan. |

**Laporan Tugas 2: Penambahan Apple**

|  |  |
| --- | --- |
| **Langkah** | **Praktikum** |
| **1.** | self.apples = []          for \_ in range(apple\_amount):              self.apples.append(Apple(self.frame\_size\_x, self.frame\_size\_y, pygame.Color(random.randint(0, 255), random.randint(0, 255), random.randint(0, 255))))  Untuk kode ini sebelumnya telah dilampirkan. Tetapi akan saya jelaskan lebih lanjut.  Jadi disini saya membuat list apples sebagai atribut class Game. Lalu saya loop dari parameter yang saya tambahkan pada class Game, untuk menambah Objek Apple sejumlah apple\_amount.  if self.snake.position in map(lambda apple: apple.position, self.apples):                  eated\_apple = [apple for apple in self.apples if apple.position == self.snake.position][0]                  eated\_apple.respawn(self.arena\_size[0], self.arena\_size[1])  Berikutnya untuk mechanic snake memakan apple, perlu kita konfigurasi agar hanya apple yang dimakan itu bisa respawn.  Teknisnya disini saya menggunakan fungsi map, lalu lambda function. Lalu kita tampung dulu posisi eated\_apple dalam variable. Lalu kita respawn eated\_apple tersebut.  for apple in self.apples:                  apple.draw(self.game\_window)  Terakhir kita gambar masing-masing apple pada layar window pygame menggunakan looping.  **Output:**  **Game:**    **Game Over:** |

***--#BeraksiBerprestasiBersinergi--***