Nama : Zulfi Masyita Resia Reswati

NIM : V3920063

Kelas : TI E

Prodi : D3 Teknik Informatika

Matkul : Praktikum Game Development

1. Urutan Kode Program

*#PART A*

import pygame, sys, random

class Block(pygame.sprite.Sprite):

    def \_\_init\_\_(self,path,x\_pos,y\_pos):

        super().\_\_init\_\_()

        self.image = pygame.image.load(path)

        self.rect = self.image.get\_rect(center = (x\_pos,y\_pos))

*#PART E*

class Player(Block):

    def \_\_init\_\_(self,path,x\_pos,y\_pos,speed):

        super().\_\_init\_\_(path,x\_pos,y\_pos)

        self.speed = speed

        self.movement = 0

    def screen\_constrain(self):

        if self.rect.top <= 0:

            self.rect.top = 0

        if self.rect.bottom >= screen\_height:

            self.rect.bottom = screen\_height

    def update(self,ball\_group):

        self.rect.y += self.movement

        self.screen\_constrain()

*#PART C*

class Ball(Block):

    def \_\_init\_\_(self,path,x\_pos,y\_pos,speed\_x,speed\_y,paddles):

        super().\_\_init\_\_(path,x\_pos,y\_pos)

        self.speed\_x = speed\_x \* random.choice((-1,1))

        self.speed\_y = speed\_y \* random.choice((-1,1))

        self.paddles = paddles

        self.active = False

        self.score\_time = 0

    def update(self):

        if self.active:

            self.rect.x += self.speed\_x

            self.rect.y += self.speed\_y

            self.collisions()

        else:

            self.restart\_counter()

*#PART G*

    def collisions(self):

        if self.rect.top <= 0 or self.rect.bottom >= screen\_height:

            pygame.mixer.Sound.play(plob\_sound)

            self.speed\_y \*= -1

        if pygame.sprite.spritecollide(self,self.paddles,False):

            pygame.mixer.Sound.play(plob\_sound)

            collision\_paddle = pygame.sprite.spritecollide(self,self.paddles,False)[0].rect

            if abs(self.rect.right - collision\_paddle.left) < 10 and self.speed\_x > 0:

                self.speed\_x \*= -1

            if abs(self.rect.left - collision\_paddle.right) < 10 and self.speed\_x < 0:

                self.speed\_x \*= -1

            if abs(self.rect.top - collision\_paddle.bottom) < 10 and self.speed\_y < 0:

                self.rect.top = collision\_paddle.bottom

                self.speed\_y \*= -1

            if abs(self.rect.bottom - collision\_paddle.top) < 10 and self.speed\_y > 0:

                self.rect.bottom = collision\_paddle.top

                self.speed\_y \*= -1

*#PART B*

    def reset\_ball(self):

        self.active = False

        self.speed\_x \*= random.choice((-1,1))

        self.speed\_y \*= random.choice((-1,1))

        self.score\_time = pygame.time.get\_ticks()

        self.rect.center = (screen\_width/2,screen\_height/2)

        pygame.mixer.Sound.play(score\_sound)

*# PART M*

    def restart\_counter(self):

        current\_time = pygame.time.get\_ticks()

        countdown\_number = 3

        if current\_time - self.score\_time <= 700:

            countdown\_number = 3

        if 700 < current\_time - self.score\_time <= 1400:

            countdown\_number = 2

        if 1400 < current\_time - self.score\_time <= 2100:

            countdown\_number = 1

        if current\_time - self.score\_time >= 2100:

            self.active = True

        time\_counter = basic\_font.render(str(countdown\_number),True,accent\_color)

        time\_counter\_rect = time\_counter.get\_rect(center = (screen\_width/2,screen\_height/2 + 50))

        pygame.draw.rect(screen,bg\_color,time\_counter\_rect)

        screen.blit(time\_counter,time\_counter\_rect)

*#PART I*

class Opponent(Block):

    def \_\_init\_\_(self,path,x\_pos,y\_pos,speed):

        super().\_\_init\_\_(path,x\_pos,y\_pos)

        self.speed = speed

    def update(self,ball\_group):

        if self.rect.top < ball\_group.sprite.rect.y:

            self.rect.y += self.speed

        if self.rect.bottom > ball\_group.sprite.rect.y:

            self.rect.y -= self.speed

        self.constrain()

    def constrain(self):

        if self.rect.top <= 0: self.rect.top = 0

        if self.rect.bottom >= screen\_height: self.rect.bottom = screen\_height

*#PART H*

class GameManager:

    def \_\_init\_\_(self,ball\_group,paddle\_group):

        self.player\_score = 0

        self.opponent\_score = 0

        self.ball\_group = ball\_group

        self.paddle\_group = paddle\_group

    def run\_game(self):

*# Drawing the game objects*

        self.paddle\_group.draw(screen)

        self.ball\_group.draw(screen)

*# Updating the game objects*

        self.paddle\_group.update(self.ball\_group)

        self.ball\_group.update()

        self.reset\_ball()

        self.draw\_score()

*# PART J*

    def reset\_ball(self):

        if self.ball\_group.sprite.rect.right >= screen\_width:

            self.opponent\_score += 1

            self.ball\_group.sprite.reset\_ball()

        if self.ball\_group.sprite.rect.left <= 0:

            self.player\_score += 1

            self.ball\_group.sprite.reset\_ball()

    def draw\_score(self):

        player\_score = basic\_font.render(str(self.player\_score),True,accent\_color)

        opponent\_score = basic\_font.render(str(self.opponent\_score),True,accent\_color)

        player\_score\_rect = player\_score.get\_rect(midleft = (screen\_width / 2 + 40,screen\_height/2))

        opponent\_score\_rect = opponent\_score.get\_rect(midright = (screen\_width / 2 - 40,screen\_height/2))

        screen.blit(player\_score,player\_score\_rect)

        screen.blit(opponent\_score,opponent\_score\_rect)

*# PART D*

pygame.mixer.pre\_init(44100,-16,2,512)

pygame.init()

clock = pygame.time.Clock()

*# Main Window*

screen\_width = 720

screen\_height = 480

screen = pygame.display.set\_mode((screen\_width,screen\_height))

pygame.display.set\_caption('Pong')

*# Global Variables*

bg\_color = pygame.Color('#2F373F')

accent\_color = (27,35,43)

basic\_font = pygame.font.Font('freesansbold.ttf', 32)

plob\_sound = pygame.mixer.Sound("pong.ogg")

score\_sound = pygame.mixer.Sound("score.ogg")

middle\_strip = pygame.Rect(screen\_width/2 - 2,0,4,screen\_height)

*# PART F*

player = Player('Paddle.png',screen\_width - 20,screen\_height/2,5)

opponent = Opponent('Paddle.png',20,screen\_width/2,5)

paddle\_group = pygame.sprite.Group()

paddle\_group.add(player)

paddle\_group.add(opponent)

ball = Ball('Ball.png',screen\_width/2,screen\_height/2,4,4,paddle\_group)

ball\_sprite = pygame.sprite.GroupSingle()

ball\_sprite.add(ball)

game\_manager = GameManager(ball\_sprite,paddle\_group)

*# PART L*

while True:

    for event in pygame.event.get():

        if event.type == pygame.QUIT:

            pygame.quit()

            sys.exit()

        if event.type == pygame.KEYDOWN:

            if event.key == pygame.K\_UP:

                player.movement -= player.speed

            if event.key == pygame.K\_DOWN:

                player.movement += player.speed

        if event.type == pygame.KEYUP:

            if event.key == pygame.K\_UP:

                player.movement += player.speed

            if event.key == pygame.K\_DOWN:

                player.movement -= player.speed

*# PART K*

*# Background Stuff*

    screen.fill(bg\_color)

    pygame.draw.rect(screen,accent\_color,middle\_strip)

*# Run the game*

    game\_manager.run\_game()

*# Rendering*

    pygame.display.flip()

    clock.tick(120)

1. Langkah Selanjutnya adalah , identifikasi pada bagian manakah implementasi AI pada Program game tersebut. Jelaskan!

Jawab :

Yang merupakan implementasi AI pada program game ini adalah bagian class Ball(block) dan class game manager.

* Pada class Ball(block) yaitu implementasi AI dalam class ini adalah pada fungsi collisions, reset\_ball dan restart\_counter karena pada fungsi ini adalah untuk mendeteksi adanya tabrakan antara 2 objek atau lebih yang dideklarasikan menggunakan suatu kondisi.
* Sedangkan pada class game manager yaitu dalam kelas ini terdapat implementasi AI pada run\_game, reset\_ball dan draw score. Ini karena pada run\_game perlu pengamatan dalam lingkungan game yaitu menggambarkan objek dalam game kemudian mengupdate situasi yang harus dilakukan objek

1. Jelaskan Bagaimana Alur AI yang digunakan pada program tersebut !

Jawab :

Class Ball memiliki alur sebagai berikut:

* Fungsi collisions

Fungsi ini adalah untuk mendeteksi adanya tabrakan antara 2 objek atau lebih yang dideklarasikan menggunakan suatu kondisi

* Fungsi reset\_ball

fungsi yang menambahkan keputusan diikuti dengan deklarasi baru

* Fungsi restart\_counter

fungsi yang mengatur ulang jarak counter dengan mengindetifikasikan lingkungan dari class objek dan membuat kondisi baru

Class GameManager memiliki alur sebagai berikut :

* Fungsi run\_game

disini perlu pengamatan dalam lingkungan game yaitu menggambarkan objek dalam game kemudian mengupdate situasi yang harus dilakukan objek

* Fungsi reset\_ball

terdapat kondisi yang memungkinkan objek untuk membuat suatu perhitungan dari kejadian yang dilakukan oleh objek

* Fungsi draw\_score

fungsi ini adalah bentuk hasil keputusan dari sebuah kondisi yang telah diamati dari lingkungannya