МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ОДЕСЬКА ПОЛІТЕХНІКА»

ІНСТИТУТ КОМП’ЮТЕРНИХ СИСТЕМ

КАФЕДРА ІЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Протокол лабораторної роботи №6

З дисципліни «Основи програмування»

на тему:

«Створення ігрової програми засобами бібліотеки Pygame»

Варіант 23

Виконав студент групи АС-211:

Пасека Ілля Вікторович

Перевірила:

д. Комлева Н.О.

Одеса 2021

**Зміст**

Завдання..................................................................................................................3

Код …....................................................................................................................3-7

Робота програми......................................................................................................7

Висновоки................................................................................................................7

**Завдання**

Необхідно придумати концепцію гри та створити ігрову програму за власними принципами та алгоритмом роботи з використанням спрайтів, можливостей руху спрайтів, налаштування кольорів та ін. Як основу взяти шаблон для ігрового проекту .

**Код**

**Файл main:**

import pygame

import math

from settings import \*

from classes import Player

from classes import Zombie

def signum(x):

    return 1 if (x > 0) else -1 if (x < 0) else 0

pygame.init()

sc = pygame.display.set\_mode((Screen\_height, Screen\_widht), pygame.DOUBLEBUF | pygame.HWSURFACE)

pygame.display.set\_caption(" Shooter ")

pygame.mixer.music.load("Hell on Earth.mp3")

pygame.mixer.music.play(-1)

clock = pygame.time.Clock()

player = Player()

wall\_1 = pygame.image.load("block.jpg")

wall\_1\_rect = wall\_1.get\_rect(center = (200, 200))

wall\_2 = pygame.image.load("block.jpg")

wall\_2\_rect = wall\_2.get\_rect(center = (1000, 200))

wall\_3 = pygame.image.load("block.jpg")

wall\_3\_rect = wall\_3.get\_rect(center = (200, 600))

wall\_4 = pygame.image.load("block.jpg")

wall\_4\_rect = wall\_3.get\_rect(center = (1000, 600))

wall\_5 = pygame.image.load("block.jpg")

wall\_5\_rect = wall\_3.get\_rect(center = (600, 400))

bullets = []

zomb\_spawn = (((0,1366), (-800,-400)),

              ((0,1366), (1200,1600)),

              ((-800,-400), (0,800)),

              ((1766,2166), (0,800)))

zomb\_spawn\_sec = random.randint(0, 3)

zombies = [Zombie(random.randint(zomb\_spawn[zomb\_spawn\_sec][0][0], zomb\_spawn[zomb\_spawn\_sec][0][1]),

                  random.randint(zomb\_spawn[zomb\_spawn\_sec][1][0], zomb\_spawn[zomb\_spawn\_sec][1][1]),

                  zombie\_speed) for i in range(zombie\_count)]

while Running\_flag:

    for event in pygame.event.get():

        if event.type == pygame.QUIT:

            Running\_flag = False

    sc.fill("darkgrey")

    sold\_surf = pygame.image.load("soldier.png")

    sold\_rect = sold\_surf.get\_rect(center=player.pos)

    sold\_surf = pygame.transform.rotate(sold\_surf, player.sold\_angle)

    sc.blit(wall\_1, wall\_1\_rect)

    sc.blit(wall\_2, wall\_2\_rect)

    sc.blit(wall\_3, wall\_3\_rect)

    sc.blit(wall\_4, wall\_4\_rect)

    sc.blit(wall\_5, wall\_5\_rect)

    sc.blit(sold\_surf, sold\_rect)

    # sc.blit(zomb\_surf, zomb\_rect)

    for zomb in zombies:

        rot = pygame.transform.rotate(zomb.zomb\_surf, math.degrees(zomb.angle))

        sc.blit(rot, zomb.zomb\_rect)

    #pygame.draw.circle(sc, "green" ,player.pos, 16)

    #pygame.draw.line(sc, "green", player.pos, (player.x + height \* math.cos(player.angle),

                                                                           # player.y + height\* math.sin(player.angle)))

    for bullet in bullets:

        bullet.update()

    for bullet in bullets:

        bullet.draw(sc)

    for bullet in bullets:

        if bullet.rect.centerx < 0 or bullet.rect.centerx > 1366 or \

                bullet.rect.centery < 0 or bullet.rect.centery > 768:

            bullets.remove(bullet)

        if wall\_1\_rect.colliderect(bullet.rect):

            bullets.remove(bullet)

        if wall\_2\_rect.colliderect(bullet.rect):

            bullets.remove(bullet)

        if wall\_3\_rect.colliderect(bullet.rect):

            bullets.remove(bullet)

        if wall\_4\_rect.colliderect(bullet.rect):

            bullets.remove(bullet)

        if wall\_5\_rect.colliderect(bullet.rect):

            bullets.remove(bullet)

    for zomb in zombies:

        for bullet in bullets:

            if zomb.zomb\_rect.colliderect(bullet.rect):

                bullets.remove(bullet)

                zomb.health = zomb.health - b\_damage

                print(zomb.health)

                if zomb.health <= 0:

                    print("zombie killed")

                    killed\_zombies+=1

                    zombies.remove(zomb)

                    break

    for zomb in zombies:

        if zomb.zomb\_rect.colliderect(sold\_rect):

            soldier\_hp= soldier\_hp-zombie\_aim

            print (soldier\_hp)

            if soldier\_hp <=0 :

                print("YOU WAS KILLED, TRY AGAIN")

                Running\_flag = False

                break

    for zomb in zombies :

        zomb.zomb\_update(player.pos)

        print(zomb.angle, math.degrees(zomb.angle))

    f = pygame.font.SysFont(None, 24)

    sc\_text= f.render("HP: ", 1, "red")

    sc\_text\_rect= sc\_text.get\_rect(center=(Screen\_height//2, 10))

    sc.blit(sc\_text, sc\_text\_rect)

    f1 = pygame.font.SysFont(None, 24)

    sc\_text1= f1.render(str(int(soldier\_hp)), 1, "red")

    sc\_text\_rect1= sc\_text1.get\_rect(center=(Screen\_height//2+27, 10))

    sc.blit(sc\_text1, sc\_text\_rect1)

    f2 = pygame.font.SysFont(None, 24)

    sc\_text2= f2.render("Killled zombies:", 1, "blue")

    sc\_text\_rect2= sc\_text2.get\_rect(center=(Screen\_height//2, 760))

    sc.blit(sc\_text2, sc\_text\_rect2)

    f3 = pygame.font.SysFont(None, 24)

    sc\_text3= f3.render(str(killed\_zombies), 1, "blue")

    sc\_text\_rect3= sc\_text3.get\_rect(center=(Screen\_height//2+78, 760))

    sc.blit(sc\_text3, sc\_text\_rect3)

    # debug

    #pygame.draw.circle(sc, (0, 255, 0), (sold\_rect.centerx, sold\_rect.centery), 1)

    #pygame.draw.circle(sc, (0, 255, 0), (wall\_1\_rect.centerx, wall\_1\_rect.centery), 1)

    #pygame.draw.line(sc, (0, 255, 0),

    #                 (sold\_rect.centerx, sold\_rect.centery),

    #                 (wall\_1\_rect.centerx, wall\_1\_rect.centery), 2)

    #pygame.draw.line(sc, (0, 0, 255),

    #                 (sold\_rect.centerx, sold\_rect.centery - signum(sold\_rect.centery - wall\_1\_rect.centery) \* sold\_rect.height / 2),

    #                 (wall\_1\_rect.centerx, wall\_1\_rect.centery), 2)

    #pygame.draw.line(sc, (0, 0, 255),

    #                 (sold\_rect.centerx - signum(sold\_rect.centerx - wall\_1\_rect.centerx) \* sold\_rect.width / 2, sold\_rect.centery),

    #                 (wall\_1\_rect.centerx, wall\_1\_rect.centery), 2)

    ##

    if len(zombies) == 0:

        if player.delay > 3:

            player.delay -= 1

        if zombie\_speed <= 4:

            zombie\_speed += 1

        if zombie\_count <= 50:

            zombie\_count += 4

        zomb\_spawn\_sec = random.randint(0, 3)

        zombies = [Zombie(random.randint(zomb\_spawn[zomb\_spawn\_sec][0][0], zomb\_spawn[zomb\_spawn\_sec][0][1]),

                          random.randint(zomb\_spawn[zomb\_spawn\_sec][1][0], zomb\_spawn[zomb\_spawn\_sec][1][1]),

                          zombie\_speed) for i in range(zombie\_count)]

    for zomb in zombies:

        if wall\_1\_rect.colliderect(zomb.zomb\_rect):

            pos = [zomb.zomb\_rect.centerx - wall\_1\_rect.centerx, zomb.zomb\_rect.centery - wall\_1\_rect.centery]

            if (abs(pos[0]) / (wall\_1\_rect.width / 2)) > (abs(pos[1]) / (wall\_1\_rect.height / 2)):

                pos[0] = signum(pos[0])

                pos[1] = 0

                zomb.pos[0] += zombie\_speed \* pos[0]

            if (abs(pos[0]) / (wall\_1\_rect.width / 2)) < (abs(pos[1]) / (wall\_1\_rect.height / 2)):

                pos[1] = signum(pos[1])

                pos[0] = 0

                zomb.pos[1] += zombie\_speed \* pos[1]

        if wall\_2\_rect.colliderect(zomb.zomb\_rect):

            pos = [zomb.zomb\_rect.centerx - wall\_2\_rect.centerx, zomb.zomb\_rect.centery - wall\_2\_rect.centery]

            if (abs(pos[0]) / (wall\_2\_rect.width / 2)) > (abs(pos[1]) / (wall\_2\_rect.height / 2)):

                pos[0] = signum(pos[0])

                pos[1] = 0

                zomb.pos[0] += zombie\_speed \* pos[0]

            if (abs(pos[0]) / (wall\_2\_rect.width / 2)) < (abs(pos[1]) / (wall\_2\_rect.height / 2)):

                pos[1] = signum(pos[1])

                pos[0] = 0

                zomb.pos[1] += zombie\_speed \* pos[1]

        if wall\_3\_rect.colliderect(zomb.zomb\_rect):

            pos = [zomb.zomb\_rect.centerx - wall\_3\_rect.centerx, zomb.zomb\_rect.centery - wall\_3\_rect.centery]

            if (abs(pos[0]) / (wall\_3\_rect.width / 2)) > (abs(pos[1]) / (wall\_3\_rect.height / 2)):

                pos[0] = signum(pos[0])

                pos[1] = 0

                zomb.pos[0] += zombie\_speed \* pos[0]

            if (abs(pos[0]) / (wall\_3\_rect.width / 2)) < (abs(pos[1]) / (wall\_3\_rect.height / 2)):

                pos[1] = signum(pos[1])

                pos[0] = 0

                zomb.pos[1] += zombie\_speed \* pos[1]

        if wall\_4\_rect.colliderect(zomb.zomb\_rect):

            pos = [zomb.zomb\_rect.centerx - wall\_4\_rect.centerx, zomb.zomb\_rect.centery - wall\_4\_rect.centery]

            if (abs(pos[0]) / (wall\_4\_rect.width / 2)) > (abs(pos[1]) / (wall\_4\_rect.height / 2)):

                pos[0] = signum(pos[0])

                pos[1] = 0

                zomb.pos[0] += zombie\_speed \* pos[0]

            if (abs(pos[0]) / (wall\_4\_rect.width / 2)) < (abs(pos[1]) / (wall\_4\_rect.height / 2)):

                pos[1] = signum(pos[1])

                pos[0] = 0

                zomb.pos[1] += zombie\_speed \* pos[1]

        if wall\_5\_rect.colliderect(zomb.zomb\_rect):

            pos = [zomb.zomb\_rect.centerx - wall\_5\_rect.centerx, zomb.zomb\_rect.centery - wall\_5\_rect.centery]

            if (abs(pos[0]) / (wall\_5\_rect.width / 2)) > (abs(pos[1]) / (wall\_5\_rect.height / 2)):

                pos[0] = signum(pos[0])

                pos[1] = 0

                zomb.pos[0] += zombie\_speed \* pos[0]

            if (abs(pos[0]) / (wall\_5\_rect.width / 2)) < (abs(pos[1]) / (wall\_5\_rect.height / 2)):

                pos[1] = signum(pos[1])

                pos[0] = 0

                zomb.pos[1] += zombie\_speed \* pos[1]

    if wall\_1\_rect.colliderect(sold\_rect):

        temp\_pos = [sold\_rect.centerx - wall\_1\_rect.centerx, sold\_rect.centery - wall\_1\_rect.centery]

        if (abs(temp\_pos[0]) / (wall\_1\_rect.width / 2)) > (abs(temp\_pos[1]) / (wall\_1\_rect.height / 2)):

            temp\_pos[0] = signum(temp\_pos[0])

            temp\_pos[1] = 0

            player.x += player\_speed \* temp\_pos[0]

        if (abs(temp\_pos[0]) / (wall\_1\_rect.width / 2)) < (abs(temp\_pos[1]) / (wall\_1\_rect.height / 2)):

            temp\_pos[1] = signum(temp\_pos[1])

            temp\_pos[0] = 0

            player.y += player\_speed \* temp\_pos[1]

    if wall\_2\_rect.colliderect(sold\_rect):

        temp\_pos = [sold\_rect.centerx - wall\_2\_rect.centerx, sold\_rect.centery - wall\_2\_rect.centery]

        if (abs(temp\_pos[0]) / (wall\_2\_rect.width / 2)) > (abs(temp\_pos[1]) / (wall\_2\_rect.height / 2)):

            temp\_pos[0] = signum(temp\_pos[0])

            temp\_pos[1] = 0

            player.x += player\_speed \* temp\_pos[0]

        if (abs(temp\_pos[0]) / (wall\_2\_rect.width / 2)) < (abs(temp\_pos[1]) / (wall\_2\_rect.height / 2)):

            temp\_pos[1] = signum(temp\_pos[1])

            temp\_pos[0] = 0

            player.y += player\_speed \* temp\_pos[1]

    if wall\_3\_rect.colliderect(sold\_rect):

        temp\_pos = [sold\_rect.centerx - wall\_3\_rect.centerx, sold\_rect.centery - wall\_3\_rect.centery]

        if (abs(temp\_pos[0]) / (wall\_3\_rect.width / 2)) > (abs(temp\_pos[1]) / (wall\_3\_rect.height / 2)):

            temp\_pos[0] = signum(temp\_pos[0])

            temp\_pos[1] = 0

            player.x += player\_speed \* temp\_pos[0]

        if (abs(temp\_pos[0]) / (wall\_3\_rect.width / 2)) < (abs(temp\_pos[1]) / (wall\_3\_rect.height / 2)):

            temp\_pos[1] = signum(temp\_pos[1])

            temp\_pos[0] = 0

            player.y += player\_speed \* temp\_pos[1]

    if wall\_4\_rect.colliderect(sold\_rect):

        temp\_pos = [sold\_rect.centerx - wall\_4\_rect.centerx, sold\_rect.centery - wall\_4\_rect.centery]

        if (abs(temp\_pos[0]) / (wall\_4\_rect.width / 2)) > (abs(temp\_pos[1]) / (wall\_4\_rect.height / 2)):

            temp\_pos[0] = signum(temp\_pos[0])

            temp\_pos[1] = 0

            player.x += player\_speed \* temp\_pos[0]

        if (abs(temp\_pos[0]) / (wall\_4\_rect.width / 2)) < (abs(temp\_pos[1]) / (wall\_4\_rect.height / 2)):

            temp\_pos[1] = signum(temp\_pos[1])

            temp\_pos[0] = 0

            player.y += player\_speed \* temp\_pos[1]

    if wall\_5\_rect.colliderect(sold\_rect):

        temp\_pos = [sold\_rect.centerx - wall\_5\_rect.centerx, sold\_rect.centery - wall\_5\_rect.centery]

        if (abs(temp\_pos[0]) / (wall\_5\_rect.width / 2)) > (abs(temp\_pos[1]) / (wall\_5\_rect.height / 2)):

            temp\_pos[0] = signum(temp\_pos[0])

            temp\_pos[1] = 0

            player.x += player\_speed \* temp\_pos[0]

        if (abs(temp\_pos[0]) / (wall\_5\_rect.width / 2)) < (abs(temp\_pos[1]) / (wall\_5\_rect.height / 2)):

            temp\_pos[1] = signum(temp\_pos[1])

            temp\_pos[0] = 0

            player.y += player\_speed \* temp\_pos[1]

    player.movement(bullets)

    pygame.display.update()

    clock.tick(FPS)

pygame.quit()

**Файл classes:**

from settings import \*

import pygame

import math

import time

class Bullet:

    def \_\_init\_\_(self, bullet\_angle, start\_pos):

        self.coords = start\_pos

        self.angle = bullet\_angle

        self.image = pygame.image.load("e4b.png").convert\_alpha()

        self.rect = self.image.get\_rect(center = self.coords)

    def update(self):

        self.coords[0] += bullet\_speed \* math.cos(self.angle)

        self.coords[1] += bullet\_speed \* math.sin(self.angle)

        self.rect.centerx = self.coords[0]

        self.rect.centery = self.coords[1]

    def draw(self, screen):

        screen.blit(self.image, self.rect)

class Zombie:

    def \_\_init\_\_(self, posx, posy, speed):

        self.pos = [posx, posy]

        self.zomb\_surf = pygame.image.load("zombie.png")

        self.zomb\_rect = self.zomb\_surf.get\_rect(center=[self.pos[0], self.pos[1]])

        self.health = 100

        self.speed = speed

        self.angle = 0

    def zomb\_update(self, sold\_pos):

        self.angle = math.atan2(self.pos[0] - sold\_pos[0], self.pos[1] - sold\_pos[1]) + math.pi/2

        self.pos[0] += math.cos(self.angle) \* self.speed

        self.pos[1] -= math.sin(self.angle) \* self.speed

        self.zomb\_rect.centerx = self.pos[0]

        self.zomb\_rect.centery = self.pos[1]

        return self.pos

class Player:

    def \_\_init\_\_(self):

        self.x = player\_x

        self.y = player\_y

        self.angle = player\_angle

        self.sold\_angle = sol\_angle

        self.counter = 0

        self.delay = player\_delay

    @property

    def pos(self):

        return self.x, self.y

    def movement(self, bullets):

        sin\_a = math.sin(self.angle)

        cos\_a = math.cos(self.angle)

        keys = pygame.key.get\_pressed()

        if keys[pygame.K\_w]:

            self.y -= player\_speed

            if self.y < 0:

                self.y = 0

        if keys[pygame.K\_s]:

            self.y += player\_speed

            if self.y > 768:

                self.y = 768

        if keys[pygame.K\_a]:

            self.x -= player\_speed

            if self.x < 0:

                self.x = 0

        if keys[pygame.K\_d]:

            self.x += player\_speed

            if self.x > 1366:

                self.x = 1366

        if keys[pygame.K\_LEFT]:

            self.sold\_angle += 4

            self.angle -= 0.06981317008

        if keys[pygame.K\_RIGHT]:

            self.sold\_angle -= 4

            self.angle += 0.06981317008

        if keys[pygame.K\_SPACE]:

            self.shoot(bullets)

    def shoot(self, bullets):

        if self.counter % self.delay == 0:

            bullets.append(Bullet(self.angle, [self.x + height \* math.cos(self.angle),

                                               self.y + height \* math.sin(self.angle)]))

        self.counter += 1

        self.counter %= FPS

**Файл settings:**

import math

import random

Running\_flag = True

FPS=60

Screen\_height = 1366

Screen\_widht = 768

#player position

player\_x =  700

player\_y =  700

player\_speed = 4

player\_angle = 0

height = 30

# bullet

bullet\_speed = 10

scatter = random.random()\*(0.1745329252+0.1745329252)-0.1745329252

#soldier

sol\_angle = 0

soldier\_hp=100

zombie\_aim=0.03

player\_delay=6

#ZOMBIE

zombie\_start\_pos= (400, 400)

b\_damage=50

zombie\_angle=0

zombie\_health=100

killed\_zombies=0

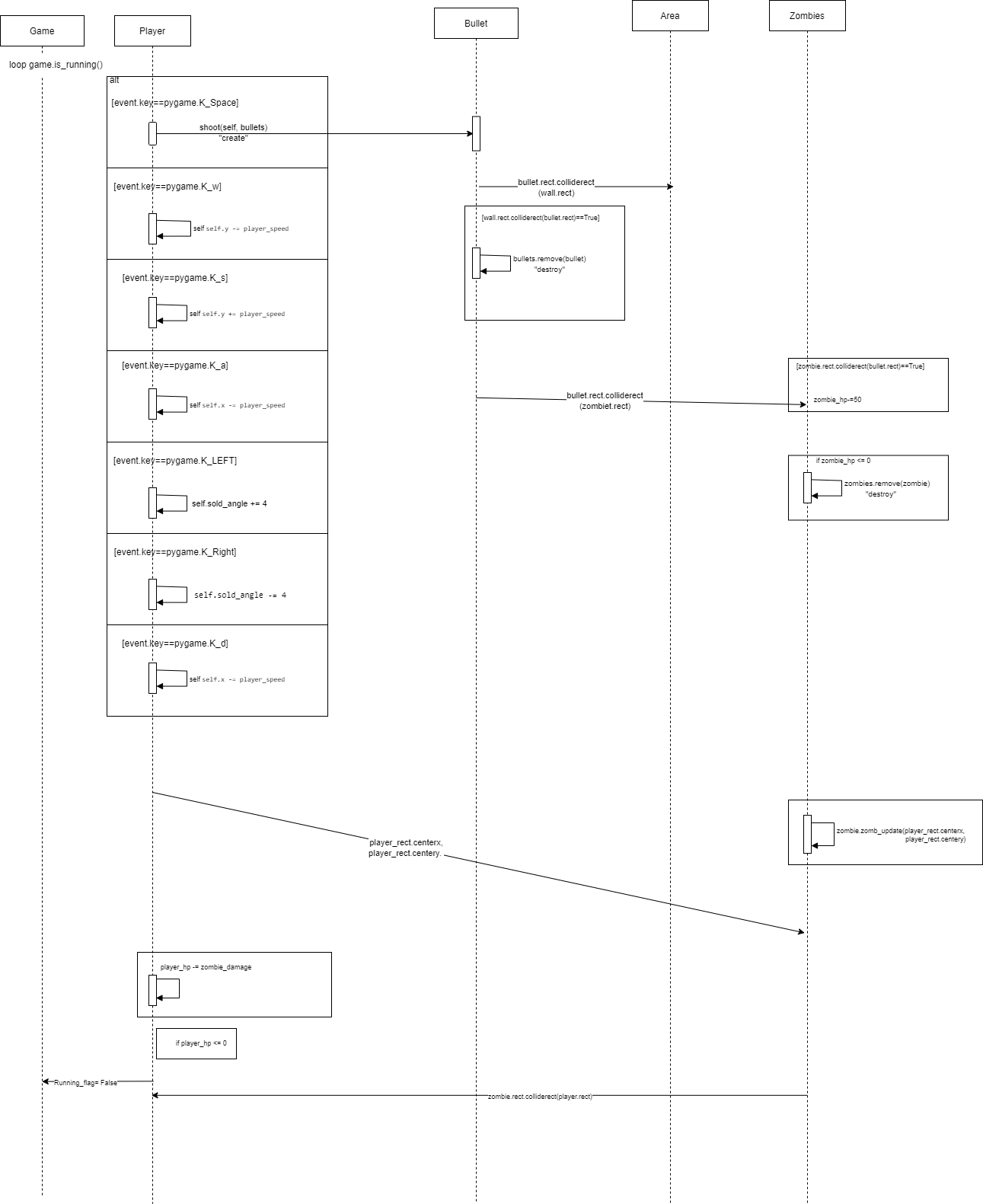
zombie\_speed=2

zombie\_count=10

**Приклад викоонання програми:**

Приклад виконання лабораторної роботи див. у відео

Sequence Diagram



З матеріалами лабораторної роботи можна ознайомитись за посиланням: https://drive.google.com/drive/u/0/folders/1vK\_b\_SeLI8OxTEDzMlFa2WP\_xeqomCmr

**Висновки:**

В цій лабораторній роботі я навчився створювати ігри з використанням бібліотеки pygame. Вивчив базові медоди роботи цієї бібліотеки та створив власну гру.