MENTOR : PROF. GORAN BONETA

Autor : Karlo Brkić
2.5 razred, Gimnazija Andrije Mohorovičića

ERIK BALL

TEHNIČKA DOKUMENTACIJA

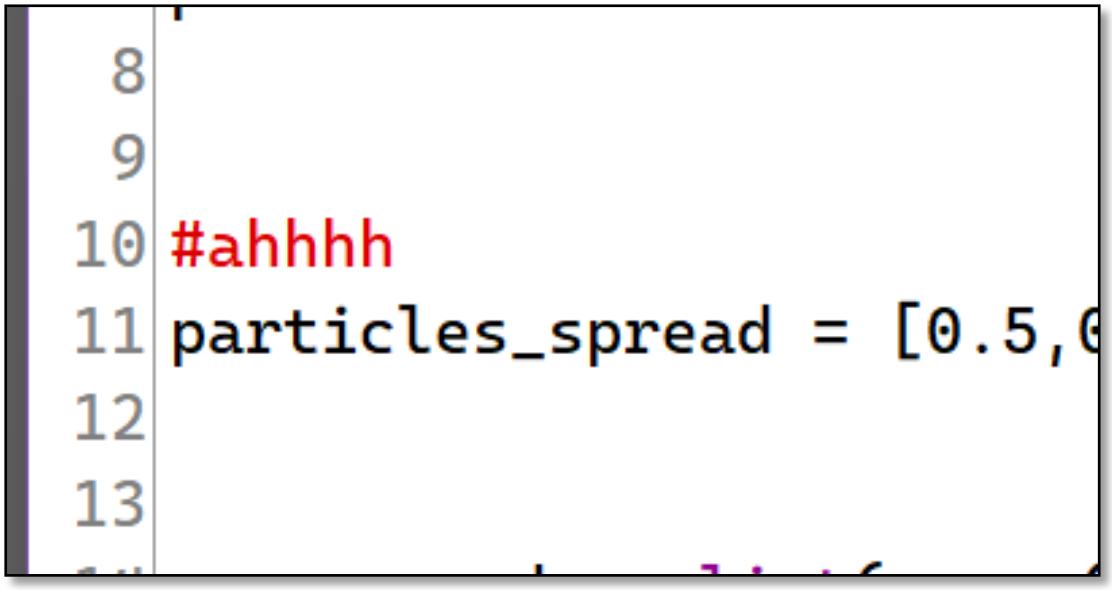
Uvod

Ovaj program dio je projektnog zadatka iz informatike koji se provodio u drugoj polovici svibnja i prvoj polovici lipnja. 100% programa je napisano u Pythonu, a multimedijske datoteke su obrađivane u svojim programima.

Program je većinom originalna kreacija gospodina Karla Brkića (mene), nijedan komad koda nije copy-pastean. Naravno, s obzirom da sam potpuni početnik u Pygameu, tutorijali su bivali praćeni, ali dao sam si truda sve shvatiti i ništa slijepo ne kopirati.

Nema specifične inspiracije za igru, možemo ju svrstati u top-down shootere, kao što su *The Binding of Isaac* ili *Hotline Miami.* Mehanike igre su vrlo jednostavne, nema kompleksnih sustava kao što su leveli ili atributi neprijatelja, stoga sličnosti s navedenim igrama počinju i završavaju u njihovom žanru.

Možemo pomicati našeg igrača i gađati neprijatelje koji umiru pri pogotku, ali se pojavljuju u većim brojevima kako igra teče. Cilj igre je preživjeti što dulje i postići što veći *score*, gađajući neprijatelje. Igra sadrži pozadinsku glazbu te zvučne efekte pri pogotku ili pri smrti.

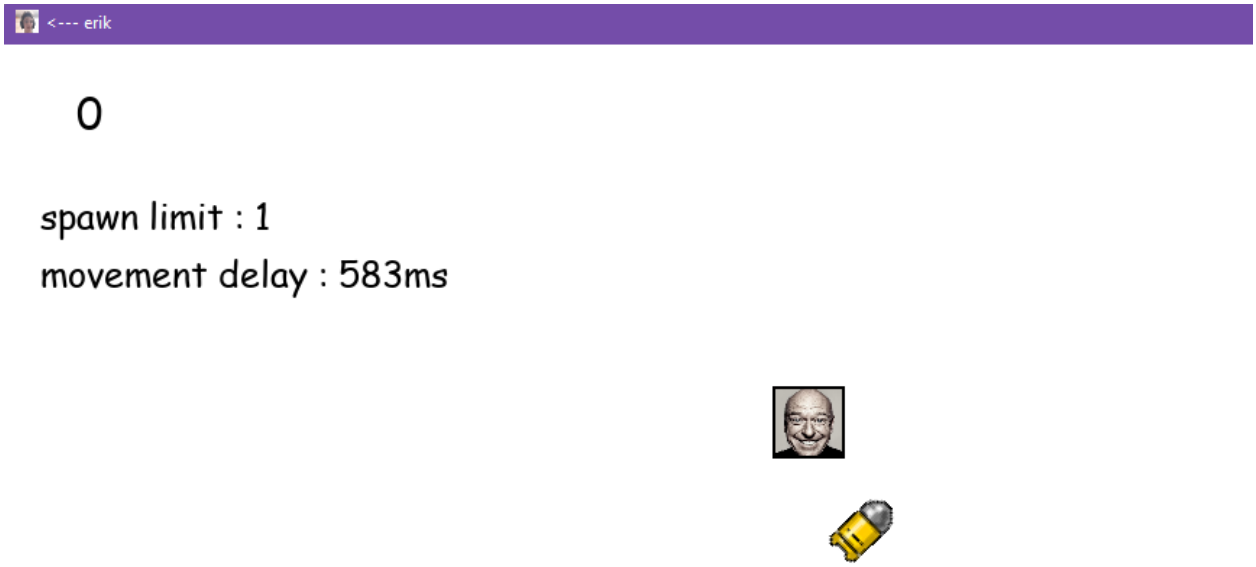


„Production value“

Opis rada

Program pokrećemo double klikom na datoteku “damn.py” unutar foldera “virus final build”. Pri pokretanju igra odmah kreće te imamo kontrolu nad našim likom. Pojavljuje se prvi neprijatelj koji će nas ubiti ako nas dotakne. Možemo ga pogoditi, nakon čega će se stvoriti još jedan.

Napomena : potreban je Python i modul Pygame (“pip install pygame” u cmd)



„Erik bozo promašuje“

Ujebemu

Kako ubijamo neprijatelje, tako nam se score povećava, ali s time i broj te brzina neprijatelja. Igra postaje prilično teška kod bodova preko 200. Ne brinite, ne treba previše vremena za zaraditi toliko bodova.

„Erik u problemima“

U gornjem lijevom kutu, zajedno sa bodovima, možemo vidjeti dvije vrijednosti : spawn limit i movement delay. Spawn limit govori koliko neprijatelja može biti živo odjednom, a movement delay govori koliko milisekundi neprijatelji zastaju pri kretanju. Ove dvije vrijednosti se povećavaju s brojem ubijenih neprijatelja.

Game over

Igra završava kada nas dotakne jedan od neprijatelja. Gubimo svu kontrolu nad Erikom te ne možemo više pucati. Jedino što možemo pritisnuti je tipka Escape za zatvoriti igru.

(Escape je moguće pritisnuti u bilo kojem trenutku.)

Tehnički opis rada

Za početak, importamo potrebne module i postavljamo osnovne varijable.

```
import pygame
import sys
import math
import random
pygame.init()
running=True
pritisnuoexit=0

```

```
#sound

deathsound = pygame.mixer.Sound("hitmarker.wav")
pygame.mixer.Sound.set_volume (deathsound, 0.01)
gameover = pygame.mixer.Sound("gameover.wav")
pygame.mixer.Sound.set_volume (gameover, 0.01)


if random.randint (1,2) == 1 :
    music = pygame.mixer.music.load("kass_theme.mp3")
    pygame.mixer.music.set_volume (0.01)
else:
    music = pygame.mixer.music.load("gangplank_galleon.mp3")
    pygame.mixer.music.set_volume (0.01)

pygame.mixer.music.play(-1)

```

Importamo potrebne zvukove i glazbu. Imamo 50/50 generator koji određuje koja od dvije pozadinske pjesme će se pustiti.

Kreiramo klasu particles (efekti).

Pygame nema ugrađenu funkciju particlesa, stoga ju sami kreiramo i koristimo attribute metaka koji se pojavljuju kasnije. Uglavnom, kreirat ćemo nasumični broj točkica sa nasumičnom brzinom kada metak pogodi neprijatelja.

```
#particles-----

#particle kad metak pogodi
class Particle():
    def __init__(self, x, y):
        self.x = x
        self.y = y
        self.velx = math.cos(bullet.angle) * 0.16 * bullet.speed * random.choice(particles_spread)
        self.vely = math.sin(bullet.angle) * 0.17 * bullet.speed * random.choice(particles_spread)
        self.lifetime = 0

    def draw(self, display):
        self.lifetime += 1
        if self.lifetime < (random.randrange(0,100)) :
            self.x += self.velx
            self.y += self.vely
            pygame.draw.circle(display, (0,0,0), (self.x, self.y), 3.5)

```

```
#player-------------
quandale=pygame.image.load("mrik.png")

class Player(pygame.sprite.Sprite):
    def __init__(self, x, y, width, height):
```

Stvaramo klasu igrača, importamo sliku i definiramo mu osnovne attribute, koje neću prikazivati.

```
def main(self):
    self.brzx=0
    self.brzy=0
    if self.D_press and not self.A_press and player.x + 50 < displaywid :
        self.brzx = self.brzina
    if self.A_press and not self.D_press and player.x + 50 > 0:
        self.brzx = -self.brzina
    if self.W_press and not self.S_press and player.y + 50 > 0:
        self.brzy = -self.brzina
    if self.S_press and not self.W_press and player.y + 50 < displayhei:
        self.brzy = self.brzina

    self.x = self.x + self.brzx
    self.y = self.y + self.brzy
```

Jednostavan movement igrača koji ga pomakne određeni broj piksela svaki frame.

Što se događa kada igrač umre. Izgubi kontrolu, score se prikaže na sredini ekrana, prikažemo hitboxeve njega i neprijatelja koji ga je ubio, zaustavljamo muziku, puštamo game over sound.

```
#kad je player dotaknut (game over) ------------------------------------------------------------------------
    def hit(self):

        self.dead= True
        display.fill((0,0,0))
        player.W_press = False
        player.A_press = False
        player.S_press = False
        player.D_press = False
        scoretext = fontscore.render(str(int(score)), 1, (255,255,255))
        press_esc = fontscore.render("press ESC", 1, (255,255,255))
        display.blit(press_esc,(1440/2 - 58,1280/2 + 40 ))
        display.blit(scoretext, (1440/2,1280/2))
        display.blit(pygame.transform.scale(enemy.animation_images[enemy.animation_count//64], (50, 50)), (e
        pygame.draw.rect(display, (255,0,0), self.hitbox, 3)
        pygame.draw.rect(display, (255,0,0), enemy.hitbox, 2)
        enemy.animation_count = -1
        pygame.mixer.music.stop()
        if self.play_gameover == True:
            gameover.play()
            self.play_gameover = False
```

```
class Enemy1 :
    def __init__(self, x, y) :
```

Klasa Neprijatelj, i osnovne osobine. (nisu prikazane)

Movement i animacije neprijatelja. Offset je random udaljenost od igrača koja postoji kako on ne bi cijelo vrijeme išao točno prema njemu, nego malo oko njega. Provjeravamo je li x i y vrijednost neprijatelja veća ili manja od x i y vrijednosti igrača, neprijatelj se miče u sukladu s time.

```
def main(self, display):
    if self.animation_count + 1 == 128 :
        self.animation_count = 0
    self.animation_count+= 1

    if self.reset_offset == 0:
        self.offset_x = random.randrange(-200, 200)
        self.offset_y = random.randrange(-200, 200)
        self.reset_offset=random.randrange(120, 150)
    else:
        self.reset_offset -= 1
        #movement enemya, disableaj da se ne mice------------

    if self.animation_count > movement_delay :
        if player.x + self.offset_x > self.x  :
            self.x += 2
        elif player.x + self.offset_x < self.x  :
            self.x -= 2
        if player.y + self.offset_y > self.y  :
            self.y += 2
        elif player.y + self.offset_y < self.y :
            self.y -= 2
```

```
#kada metak pogodi enemya
    def hit(self) :
        bullet.visible = False
        for i in range (random.randrange(15,30)):
            particles.append(Particle(enemy.x + 25, enemy.y + 25))
        if self.health > 0 :
            self.health -= 1
        else:
            self.visible = False
            enemies.pop(enemies.index(enemy))
        deathsound.play()
```

Kada je neprijatelj pogođen. Metak nestaje, neprijatelj nestaje, stvaraju se particlesi, pušta se death sound.

```
#metak-------------
bullets = []
metak = pygame.image.load("metak.png")
class PlayerBullet:
    def __init__(self, x, y, mouse_x, mouse_y):
        self.x = x
        self.y = y
        self.mouse_x = mouse_x
        self.mouse_y = mouse_y
        self.speed = 45
        self.angle = math.atan2(mouse_y - self.y, mouse_x - self.x)
        self.velx = math.cos(self.angle) * self.speed
        self.vely = math.sin(self.angle) * self.speed
        self.hitbox = (self.x-5, self.y-5, 25, 25)
        self.lifetime = 150
        self.visible = True


    def draw(self,draw):
        self.x += (self.velx)
        self.y += (self.vely)
        metakrotated = pygame.transform.rotate(metak, 270 - self.angle * 57.29)
        display.blit(metakrotated, (self.x-10, self.y-10))
        self.hitbox = (self.x-5, self.y-5, 25, 25)

```

Metak. Kalkuliramo kut kuda će se ispaliti uz pomoć trigonometrije i pozicije miša. Potrebno je rotirati sliku metka u skladu s tim kutom kako bi on “gledao” tamo kuda ide.

Main loop, bla bla, standardno sve

```
movement_delay = 70 - 4.5 * (score//20)
spawn_limit = 1
if score > 50 :
    spawn_limit = 6 + score//25
else:
    spawn_limit += score//7


while len(enemies) < spawn_limit:
        enemies.append(Enemy1(random.choice(spawn_randx), random.choice(spawn_randy)))
```

Ovdje se reguliraju spawn limit i movement delay s obzirom na score.

Neprijatelje stavljamo u listu čija dužina neće biti veća od spawn limita. Spawnamo ih u random koordinati izvan ekrana.

```
for bullet in bullets:
    if bullet.lifetime <= 0:
        bullets.pop(bullets.index(bullet))
        for i in range (random.randrange(7,20)):
            bulletparticles.append(BulletParticle(bullet.x, bullet.y))
    if bullet.visible:
        bullet.draw(display)
    for enemy in enemies :
        if enemy.visible and bullet.visible :
            if (bullet.hitbox[1] + 3 < enemy.hitbox [1] + enemy.hitbox[3] and bullet.hitbox[1] + 3  > enemy.hitbox [1]) or (bullet.hitbox[1] + 20 < enemy.hitbox [1] + e
                if bullet.hitbox[0] + 3 > enemy.hitbox[0] and bullet.hitbox[0] + 3  < enemy.hitbox[0] + enemy.hitbox[2] or (bullet.hitbox[0] + 20 > enemy.hitbox[0] and
                    enemy.hit()
                    score+= 1


for enemy in enemies:
    enemy.main(display)
    if enemy.visible :
        if player.hitbox[1] < enemy.hitbox [1] + enemy.hitbox[3] and player.hitbox[1] + player.hitbox [3]> enemy.hitbox [1]:
                if player.hitbox[0] + player.hitbox[2] > enemy.hitbox[0] and player.hitbox[0] < enemy.hitbox[0] + enemy.hitbox[2]:
                    player.hit()
```

Janky ahh homemade provjera diraju li se hitbox neprijatelja i metka. Ako da : pokreni efekt enemy.hit, povećaj score.
Također provjera diraju li se igrač i neprijatelj. Ako da : pokreni player.hit.

Ostatak main loopa je nezanimljiv (provjera koje tipke su pritisnute, prikaz teksta, uzimanje pozicije miša, yea)