#ОСНОВНОЙ КОД

#библиотеки и коды  
import pygame  
import random #спавн врагов и доск  
import os #для работы с файлом операционки  
from pygame import mixer #mp3  
from spritesheet import SpriteSheet #спрайт для движка картинок  
from enemy import Enemy #враги  
  
#иницилизируем саму игру  
mixer.init()  
pygame.init()  
  
#параметры для окна 400 на 600  
SCREEN\_WIDTH = 400  
SCREEN\_HEIGHT = 600  
  
#создаём окно с названием Вверх!  
screen = pygame.display.set\_mode((SCREEN\_WIDTH, SCREEN\_HEIGHT))  
pygame.display.set\_caption('Вверх!')  
  
#частота на 60 фпс  
clock = pygame.time.Clock()  
FPS = 60  
  
#музыка через mixer для фона, прыжка и смерти с установкой громкости  
pygame.mixer.music.load('assets/music.mp3')  
pygame.mixer.music.set\_volume(0.6)  
pygame.mixer.music.play(-1, 0.0)  
jump\_fx = pygame.mixer.Sound('assets/jump.mp3')  
jump\_fx.set\_volume(0.5)  
death\_fx = pygame.mixer.Sound('assets/death.mp3')  
death\_fx.set\_volume(0.5)  
  
  
#переменные игровые  
SCROLL\_THRESH = 200  
GRAVITY = 1  
MAX\_PLATFORMS = 10  
scroll = 0  
bg\_scroll = 0  
game\_over = False  
score = 0  
fade\_counter = 0  
  
#памятка последнего результа, запись в txt файл  
if os.path.exists('score.txt'):  
 with open('score.txt', 'r') as file:  
 high\_score = int(file.read())  
else:  
 high\_score = 0  
  
#задаём цвета rgb  
WHITE = (255, 255, 255)  
BLACK = (0, 0, 0)  
PANEL = (163, 100, 240)  
  
#шрифт  
font\_small = pygame.font.SysFont('Courier New', 20)  
font\_big = pygame.font.SysFont('Courier New', 24)  
  
#загрузка пикч  
jumpy\_image = pygame.image.load('assets/jump.png').convert\_alpha()  
bg\_image = pygame.image.load('assets/bg.png').convert\_alpha()  
platform\_image = pygame.image.load('assets/wood.png').convert\_alpha()  
#комета  
comet\_sheet\_img = pygame.image.load('assets/comet.png').convert\_alpha()  
comet\_sheet = SpriteSheet(comet\_sheet\_img)  
  
  
#def для вывода текста на экран  
def draw\_text(text, font, text\_col, x, y):  
 img = font.render(text, True, text\_col)  
 screen.blit(img, (x, y))  
  
#очки в верхней панели  
def draw\_panel():  
 pygame.draw.rect(screen, PANEL, (0, 0, SCREEN\_WIDTH, 30))  
 pygame.draw.line(screen, WHITE, (0, 30), (SCREEN\_WIDTH, 30), 2)  
 draw\_text('SCORE: ' + str(score), font\_small, WHITE, 0, 0)  
  
  
#def фона  
def draw\_bg(bg\_scroll):  
 screen.blit(bg\_image, (0, 0 + bg\_scroll))  
 screen.blit(bg\_image, (0, -600 + bg\_scroll))  
  
#Вальтер  
class Player():  
 def \_\_init\_\_(self, x, y):  
 #размер вальтера 50 на 50  
 self.image = pygame.transform.scale(jumpy\_image, (45, 45))  
 #размер для соприкосновения с доской  
 self.width = 25  
 self.height = 40  
 self.rect = pygame.Rect(0, 0, self.width, self.height)  
 self.rect.center = (x, y)  
 self.vel\_y = 0  
 self.flip = False  
  
 def move(self):  
 #сброс переменных  
 scroll = 0  
 dx = 0  
 dy = 0  
  
 #управление  
 key = pygame.key.get\_pressed()  
 if key[pygame.K\_a]:  
 dx = -10  
 self.flip = True  
 if key[pygame.K\_d]:  
 dx = 10  
 self.flip = False  
  
 #гравитация(для прыжка)  
 self.vel\_y += GRAVITY  
 dy += self.vel\_y  
  
 #вальтер не может выходить за пределы экрана  
 if self.rect.left + dx < 0:  
 dx = -self.rect.left  
 if self.rect.right + dx > SCREEN\_WIDTH:  
 dx = SCREEN\_WIDTH - self.rect.right  
  
  
 #квадрат сталкивается с платформой(проверка)  
 for platform in platform\_group:  
 #столкновение по Y  
 if platform.rect.colliderect(self.rect.x, self.rect.y + dy, self.width, self.height):  
 #находится ли над платформой  
 if self.rect.bottom < platform.rect.centery:  
 if self.vel\_y > 0:  
 self.rect.bottom = platform.rect.top  
 dy = 0  
 self.vel\_y = -20  
 jump\_fx.play()  
  
 #отскочил ли игрок в верхнюю часть экрана  
 if self.rect.top <= SCROLL\_THRESH:  
 #если игрок прыгает  
 if self.vel\_y < 0:  
 scroll = -dy  
  
 #при прыжке вальтера обновляется положение квадрата  
 self.rect.x += dx  
 self.rect.y += dy + scroll  
  
 #маску(в этом случае картинка в квадрате) обновляем  
 self.mask = pygame.mask.from\_surface(self.image)  
  
 return scroll  
  
 def draw(self):  
 screen.blit(pygame.transform.flip(self.image, self.flip, False), (self.rect.x - 12, self.rect.y - 5))  
  
#сама платформа(доски)  
class Platform(pygame.sprite.Sprite):  
 def \_\_init\_\_(self, x, y, width, moving):  
 pygame.sprite.Sprite.\_\_init\_\_(self)  
 self.image = pygame.transform.scale(platform\_image, (width, 10))  
 self.moving = moving  
 self.move\_counter = random.randint(0, 50)  
 self.direction = random.choice([-1, 1])  
 self.speed = random.randint(1, 2)  
 self.rect = self.image.get\_rect()  
 self.rect.x = x  
 self.rect.y = y  
  
 def update(self, scroll):  
 #перемещение плафтормы осуществляется до середины экрана и возвращается обратно  
 if self.moving == True:  
 self.move\_counter += 1  
 self.rect.x += self.direction \* self.speed  
  
 #изменение направления платформы, если она полностью сдвинулась или ударилась о стену  
 if self.move\_counter >= 100 or self.rect.left < 0 or self.rect.right > SCREEN\_WIDTH:  
 self.direction \*= -1  
 self.move\_counter = 0  
  
 #обновляем вертикальное положение доски  
 self.rect.y += scroll  
  
 #есть ли доска на экране  
 if self.rect.top > SCREEN\_HEIGHT:  
 self.kill()  
  
#делаем игрока из класса Игрок  
jumpy = Player(SCREEN\_WIDTH // 2, SCREEN\_HEIGHT - 150)  
  
#формируем спрайты  
platform\_group = pygame.sprite.Group()  
enemy\_group = pygame.sprite.Group()  
  
#стартовая платформа будет длиннее обычной и без движения.Положение-середина низ  
platform = Platform(SCREEN\_WIDTH // 2 - 50, SCREEN\_HEIGHT - 50, 100, False)  
platform\_group.add(platform)  
  
#заводим цикл игры  
run = True  
while run:  
  
 clock.tick(FPS)  
  
 if game\_over == False:  
 scroll = jumpy.move()  
  
 #фон  
 bg\_scroll += scroll  
 if bg\_scroll >= 600:  
 bg\_scroll = 0  
 draw\_bg(bg\_scroll)  
  
 #создаем платформы рандомно  
 if len(platform\_group) < MAX\_PLATFORMS:  
 p\_w = random.randint(40, 60)  
 p\_x = random.randint(0, SCREEN\_WIDTH - p\_w)  
 p\_y = platform.rect.y - random.randint(80, 120)  
 p\_type = random.randint(1, 2)  
 if p\_type == 1 and score > 500:  
 p\_moving = True  
 else:  
 p\_moving = False  
 platform = Platform(p\_x, p\_y, p\_w, p\_moving)  
 platform\_group.add(platform)  
  
 #и обновляем положение  
 platform\_group.update(scroll)  
  
 #кометы генерируем на очках выше 1500  
 if len(enemy\_group) == 0 and score > 1500:  
 enemy = Enemy(SCREEN\_WIDTH, 100, comet\_sheet, 1.5)  
 enemy\_group.add(enemy)  
  
 #обновляем кометы  
 enemy\_group.update(scroll, SCREEN\_WIDTH)  
  
 #обновление очков  
 if scroll > 0:  
 score += scroll  
  
 #обозначение предыдущего успеха в достижении очков(линия белая с самым высоким очком) будет брать из txt  
 pygame.draw.line(screen, WHITE, (0, score - high\_score + SCROLL\_THRESH), (SCREEN\_WIDTH, score - high\_score + SCROLL\_THRESH), 3)  
 draw\_text('HIGH SCORE', font\_small, WHITE, SCREEN\_WIDTH - 130, score - high\_score + SCROLL\_THRESH)  
  
 #загрузка на окно спрайты(картинки)  
 platform\_group.draw(screen)  
 enemy\_group.draw(screen)  
 jumpy.draw()  
  
 #сама панель игровая  
 draw\_panel()  
  
 #основа для чека конца игры  
 if jumpy.rect.top > SCREEN\_HEIGHT:  
 game\_over = True  
 death\_fx.play()  
 #столкновение с кометами тоже конец игры  
 if pygame.sprite.spritecollide(jumpy, enemy\_group, False):  
 if pygame.sprite.spritecollide(jumpy, enemy\_group, False, pygame.sprite.collide\_mask):  
 game\_over = True  
 death\_fx.play()  
 else:  
 if fade\_counter < SCREEN\_WIDTH:  
 fade\_counter += 5  
 for y in range(0, 6, 2):  
 pygame.draw.rect(screen, BLACK, (0, y \* 100, fade\_counter, 100))  
 pygame.draw.rect(screen, BLACK, (SCREEN\_WIDTH - fade\_counter, (y + 1) \* 100, SCREEN\_WIDTH, 100))  
 else:  
 draw\_text('GAME OVER!', font\_big, WHITE, 130, 200)  
 draw\_text('SCORE: ' + str(score), font\_big, WHITE, 130, 250)  
 draw\_text('PRESS SPACE TO PLAY AGAIN', font\_big, WHITE, 40, 300)  
 #обновление высоких очков  
 if score > high\_score:  
 high\_score = score  
 with open('score.txt', 'w') as file:  
 file.write(str(high\_score))  
 key = pygame.key.get\_pressed()  
 if key[pygame.K\_SPACE]:  
 #сброс переменных  
 game\_over = False  
 score = 0  
 scroll = 0  
 fade\_counter = 0  
 #перемещение вальтера  
 jumpy.rect.center = (SCREEN\_WIDTH // 2, SCREEN\_HEIGHT - 150)  
 #перемещение комет  
 enemy\_group.empty()  
 #перемещение платформ  
 platform\_group.empty()  
 #выявление стартовой доски  
 platform = Platform(SCREEN\_WIDTH // 2 - 50, SCREEN\_HEIGHT - 50, 100, False)  
 platform\_group.add(platform)  
  
  
 #обрабатываем события(управление)  
 for event in pygame.event.get():  
 if event.type == pygame.QUIT:  
 #обновляем выоские очки в txt  
 if score > high\_score:  
 high\_score = score  
 with open('score.txt', 'w') as file:  
 file.write(str(high\_score))  
 run = False  
  
  
 #окно обновления  
 pygame.display.update()  
  
  
#запуск  
pygame.quit()

#КОД ДЛЯ КОМЕТЫ

import pygame  
import random  
  
class Enemy(pygame.sprite.Sprite):  
 def \_\_init\_\_(self, SCREEN\_WIDTH, y, sprite\_sheet, scale):  
 pygame.sprite.Sprite.\_\_init\_\_(self)  
 #определение переменных  
 self.animation\_list = []  
 self.frame\_index = 0  
 self.update\_time = pygame.time.get\_ticks()  
 self.direction = random.choice([-1, 1])  
 if self.direction == 1:  
 self.flip = True  
 else:  
 self.flip = False  
  
 #загружаем спрайты с spritesheet  
 animation\_steps = 8  
 for animation in range(animation\_steps):  
 image = sprite\_sheet.get\_image(animation, 32, 32, scale, (0, 0, 0))  
 image = pygame.transform.flip(image, self.flip, False)  
 image.set\_colorkey((0, 0, 0))  
 self.animation\_list.append(image)  
   
 #комета состоит из нескольких пнг внутри одного пнг.выбираем начальную картинку и делаем прямоугольник  
 self.image = self.animation\_list[self.frame\_index]  
 self.rect = self.image.get\_rect()  
  
 if self.direction == 1:  
 self.rect.x = 0  
 else:  
 self.rect.x = SCREEN\_WIDTH  
 self.rect.y = y  
  
 def update(self, scroll, SCREEN\_WIDTH):  
 #анимация обновляется  
 ANIMATION\_COOLDOWN = 50  
 #обновление изображения в зависимости от текущего кадра  
 self.image = self.animation\_list[self.frame\_index]  
 #достаточно ли времени прошло с момента последнего обновления  
 if pygame.time.get\_ticks() - self.update\_time > ANIMATION\_COOLDOWN:  
 self.update\_time = pygame.time.get\_ticks()  
 self.frame\_index += 1  
 #если анимация закончилась, вернемся к началу(1)  
 if self.frame\_index >= len(self.animation\_list):  
 self.frame\_index = 0  
  
 #перемещение кометы  
 self.rect.x += self.direction \* 2  
 self.rect.y += scroll  
  
 #исчез ли  
 if self.rect.right < 0 or self.rect.left > SCREEN\_WIDTH:  
 self.kill()

#КОД ДЛЯ СПРАЙТОВ

import pygame  
#сборка спрайтов,картинки в пнг формате,игнор фона  
class SpriteSheet():  
 def \_\_init\_\_(self, image):  
 self.sheet = image  
#конверт в один формат пикселей  
 def get\_image(self, frame, width, height, scale, colour):  
 image = pygame.Surface((width, height)).convert\_alpha()  
 image.blit(self.sheet, (0, 0), ((frame \* width), 0, width, height))  
 image = pygame.transform.scale(image, (int(width \* scale), int(height \* scale)))  
 image.set\_colorkey(colour) #игнор фона  
  
 return image