import random  
import pygame  
from pygame.locals import \*  
import time  
  
pygame.init()  
  
  
class Dino():  
 def \_\_init\_\_(self):  
 self.Img = pygame.image.load("sprites/dino\_.png")  
 self.WIDTH, self.HEIGHT = 44, 48  
 self.Img = pygame.transform.scale(self.Img, (self.WIDTH, self.HEIGHT))  
 self.image = self.Img  
 self.x = 20  
 self.y = 170  
 self.g = -0.25 # Gravity  
 self.up = 7 # Initial upward velocity  
 self.t = 0 # time  
 self.hitbox = pygame.Rect(self.x + 5, self.y, self.WIDTH - 15, self.HEIGHT - 5)  
  
 self.runImg1 = pygame.image.load("sprites/dino\_1.png")  
 self.runImg2 = pygame.image.load("sprites/dino\_2.png")  
 self.runImg1 = pygame.transform.scale(self.runImg1, (self.WIDTH, self.HEIGHT))  
 self.runImg2 = pygame.transform.scale(self.runImg2, (self.WIDTH, self.HEIGHT))  
  
 self.duck1 = pygame.image.load("sprites/dino\_ducking1.png")  
 self.duck2 = pygame.image.load("sprites/dino\_ducking2.png")  
 self.duck1 = pygame.transform.scale(self.duck1, (self.WIDTH + 15, self.HEIGHT))  
 self.duck2 = pygame.transform.scale(self.duck2, (self.WIDTH + 15, self.HEIGHT))  
 self.is\_ducking = False  
  
 self.duckImgs = [self.duck1, self.duck2]  
  
 self.runImgs = [self.runImg1, self.runImg2]  
  
 self.jump\_sound = pygame.mixer.Sound("jump.wav")  
 self.count = 0  
 self.jumping = False  
  
 def jump(self):  
 self.y -= self.up # Start jumping  
 self.jumping = True  
 self.jump\_sound.play()  
  
 def update(self):  
 if self.y < 170: # check if jumping  
 self.up = self.up + self.g \* self.t # v = u + at  
 self.y -= self.up  
 self.t += 0.12 # incrementing time  
  
 if self.y > 170: # check if the jump is complete and resetting all variables  
 self.y = 170  
 self.t = 0  
 self.up = 7  
 self.jumping = False  
  
 if self.is\_ducking:  
 self.hitbox = pygame.Rect(self.x + 5, self.y + 20, self.WIDTH + 12, self.HEIGHT - 20)  
 self.image = self.duckImgs[int(self.count) % 2]  
 self.count += 0.2  
 elif self.jumping:  
 self.hitbox = pygame.Rect(self.x + 5, self.y, self.WIDTH - 15, self.HEIGHT - 5)  
 self.image = self.Img  
 else:  
 self.hitbox = pygame.Rect(self.x + 5, self.y, self.WIDTH - 17, self.HEIGHT - 5)  
 self.image = self.runImgs[int(self.count) % 2]  
 self.count += 0.15  
  
 def draw(self, screen):  
 screen.blit(self.image, (self.x, self.y))  
 # pygame.draw.rect(screen, (255, 0, 0), self.hitbox, 2) #In case you want to see the hitbox  
  
  
class Ptera():  
 def \_\_init\_\_(self):  
 self.width, self.height = 50, 40  
 self.im1 = pygame.image.load("sprites/ptera1.png")  
 self.im2 = pygame.image.load("sprites/ptera2.png")  
  
 self.im1 = pygame.transform.scale(self.im1, (self.width, self.height))  
 self.im2 = pygame.transform.scale(self.im2, (self.width, self.height))  
  
 self.flaps = [self.im1, self.im2]  
  
 self.image = self.im1  
  
 self.altitudes = [175, 150, 110]  
 self.x = random.randint(750, 1000) # generate a random position for ptera  
 self.y = random.choice(self.altitudes)  
  
 self.speed = 5  
 self.count = 0  
 self.is\_ptera = True  
 self.is\_cactus = False  
  
 self.hitbox = (self.x, self.y + 10, self.width, self.height - 12)  
  
 def update(self):  
 self.image = self.flaps[int(self.count) % 2] # Flapping mechanism  
 self.count += 0.1  
  
 self.x -= self.speed  
  
 if self.x < 50:  
 self.allowed = True  
  
 self.hitbox = pygame.Rect(self.x, self.y + 10, self.width, self.height - 12)  
  
 def draw(self, screen):  
 screen.blit(self.image, (self.x, self.y))  
 # pygame.draw.rect(screen, (255, 0, 0), self.hitbox, 2) #In case you want to see the hitbox  
  
  
class Cactus():  
 def \_\_init\_\_(self):  
 self.image0 = pygame.image.load("sprites/cacti-small.png")  
 self.image1 = pygame.image.load("sprites/cacti-big.png")  
 self.width0 = 45  
 self.height = 44  
 self.width1 = 65  
 self.image0 = pygame.transform.scale(self.image0, (self.width0, self.height))  
 self.image1 = pygame.transform.scale(self.image1, (self.width1, self.height))  
 self.is\_cactus = True  
 self.is\_ptera = False  
  
 self.image, self.width = random.choice([[self.image0, self.width0], [self.image1, self.width1]])  
  
 self.x = random.randint(720, 1000)  
 self.y = 175  
 self.speed = 4  
  
 self.hitbox = pygame.Rect(self.x, self.y, self.width, self.height)  
  
 def update(self):  
 self.x -= self.speed  
  
 self.hitbox = pygame.Rect(self.x, self.y, self.width, self.height)  
  
 def draw(self, screen):  
 screen.blit(self.image, (self.x, self.y))  
  
  
class Ground():  
 def \_\_init\_\_(self):  
 self.ground\_length = 1202  
 self.image1 = pygame.image.load("sprites/ground.png")  
 self.image1\_x = 0  
 self.image1\_y = 200  
 self.image2 = pygame.image.load("sprites/ground.png")  
 self.image2\_x = self.image1\_x + self.ground\_length  
 self.image2\_y = self.image1\_y  
 self.speed = 4  
  
 def draw(self, screen):  
 screen.blit(self.image1, (self.image1\_x, self.image1\_y))  
 screen.blit(self.image2, (self.image2\_x, self.image2\_y))  
  
 def update(self):  
 self.image1\_x -= self.speed  
 self.image2\_x -= self.speed  
  
 if self.image1\_x + self.ground\_length < 0:  
 self.image1\_x = self.image2\_x + self.ground\_length  
 elif self.image2\_x + self.ground\_length < 0:  
 self.image2\_x = self.image1\_x + self.ground\_length  
  
  
class Cloud():  
  
 def \_\_init\_\_(self):  
 self.image = pygame.image.load("sprites/cloud.png")  
 self.WIDTH, self.HEIGHT = 70, 40  
 self.image = pygame.transform.scale(self.image, (self.WIDTH, self.HEIGHT))  
 self.speed = 1  
 self.x = 600  
 self.y = 50  
  
 def update(self):  
 self.x -= self.speed  
  
 if self.x < -self.WIDTH:  
 self.x = 600  
 self.y = random.randint(10, 100)  
  
 def draw(self, screen):  
 screen.blit(self.image, (self.x, self.y))  
  
  
# ------------------------------------------  
# -----------------------------------------  
# AGREGADO -----------------------------  
#cargo sprites de botones  
imagen1=pygame.image.load("sprites/jugar1.png")  
imagen2=pygame.image.load("sprites/jugar2.png")  
imagen3=pygame.image.load("sprites/salir1.png")  
imagen4=pygame.image.load("sprites/salir2.png")  
  
  
# creo un rect que siga al maus para detectar la colicion con el boton  
class Cursor(pygame.Rect):  
 def \_\_init\_\_(self):  
 pygame.Rect.\_\_init\_\_(self,0,0,1,1)  
 def update(self):  
 self.left,self.top=pygame.mouse.get\_pos()  
  
# creamos el sprite boton  
class Boton(pygame.sprite.Sprite):  
 def \_\_init\_\_(self,imagen1,imagen2,x,y):  
 self.imagen\_normal=imagen1  
 self.imagen\_seleccion=imagen2  
 self.imagen\_actual=self.imagen\_normal  
 self.rect=self.imagen\_actual.get\_rect()  
 self.rect.left,self.rect.top=(x,y)  
  
 def update(self,pantalla,cursor):  
 if cursor.colliderect(self.rect):  
 self.imagen\_actual=self.imagen\_seleccion  
 else: self.imagen\_actual=self.imagen\_normal  
  
 pantalla.blit(self.imagen\_actual,self.rect)  
  
  
boton1=Boton(imagen1,imagen2,300,25)  
cursor1=Cursor()  
boton2=Boton(imagen3,imagen4,300,150)  
  
  
# ------------------------------------------------  
# --------------------------------------------------  
# ---------------------------------------------------  
  
def game():  
  
 screen = pygame.display.set\_mode((700, 250))  
 clock = pygame.time.Clock()  
 font = pygame.font.Font("freesansbold.ttf", 20)  
 check\_point = pygame.mixer.Sound("checkPoint.wav")  
 death\_sound = pygame.mixer.Sound("die.wav")  
 pygame.display.set\_caption("Dino Run")  
  
 dino\_icon = pygame.image.load("sprites/dino.png")  
 pygame.display.set\_icon(dino\_icon)  
  
 game\_over = pygame.image.load("sprites/game\_over.png")  
 replay\_button = pygame.image.load("sprites/replay\_button.png")  
 logo = pygame.image.load("sprites/logo.png")  
 GREY = (240, 240, 240)  
 WHITE = (255, 255, 255)  
  
 ground = Ground()  
 dino = Dino()  
 cloud = Cloud()  
 obstacles = [Cactus()]  
 obstacle\_start = time.time()  
 minimum\_time = 1.5  
  
 running = False  
 play\_game = True  
 dead = False  
 high\_score\_value = 0  
 FPS = 85  
  
 while play\_game:  
 if not dead:  
 screen.fill(GREY)  
 ground.draw(screen)  
 screen.blit(dino.image, (dino.x, dino.y))  
  
 # ---ELIMINAR----  
 screen.blit(logo, (200, 70))  
 # ---ELIMINAR----  
  
 # ----AGREGAR----  
 #agregamos los botones  
 cursor1.update()  
 boton1.update(screen,cursor1)  
 boton2.update(screen,cursor1)  
 #detectamos el click sobre el boton  
  
  
  
 # ----AGREGAR----  
  
 pygame.display.update()  
  
 for event in pygame.event.get():  
 if event.type==pygame.MOUSEBUTTONDOWN:  
 if cursor1.colliderect(boton1.rect):  
 running = True  
 ground = Ground()  
 dino = Dino()  
 obstacles = [Cactus()]  
 obstacle\_start = time.time()  
 dead = False  
 running = True  
 score\_value = 0  
 elif cursor1.colliderect(boton2.rect):  
 pygame.quit()  
 play\_game = False  
  
 if event.type == pygame.QUIT:  
 pygame.quit()  
 play\_game = False  
 ## -----ELIMINAR----------  
 ## -----ELIMINAR----------  
 ## -----ELIMINAR----------  
 elif event.type == pygame.KEYDOWN:  
 if event.key == pygame.K\_SPACE:  
 running = True  
 ground = Ground()  
 dino = Dino()  
 obstacles = [Cactus()]  
 obstacle\_start = time.time()  
 dead = False  
 running = True  
 score\_value = 0  
 #--------------------  
 #\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
 #\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
  
 while running:  
 clock.tick(FPS) # Controlling Frames Per Second  
  
 score = font.render("Score: " + str(int(score\_value)), True, (200, 200, 200))  
 score\_value += 0.25  
 high\_score\_value = max(high\_score\_value, score\_value)  
 high\_score = font.render("High Score: " + str(int(high\_score\_value)), True, (200, 200, 200))  
 screen.fill(GREY)  
  
 # Event handling  
 for event in pygame.event.get():  
 if event.type == pygame.QUIT:  
 pygame.quit()  
 elif event.type == pygame.KEYDOWN:  
 if event.key == pygame.K\_SPACE:  
 dino.jump()  
 elif event.key == pygame.K\_DOWN:  
 dino.is\_ducking = True  
 elif event.type == pygame.KEYUP:  
 if event.key == pygame.K\_DOWN:  
 dino.is\_ducking = False  
  
 ground.update()  
 ground.draw(screen)  
  
 cloud.update()  
 cloud.draw(screen)  
  
 dino.update()  
 dino.draw(screen)  
  
 for obstacle in obstacles:  
 if obstacle.is\_cactus:  
 obstacle.speed = ground.speed  
 elif obstacle.is\_ptera:  
 obstacle.speed = ground.speed + 1  
 obstacle.update()  
 obstacle.draw(screen)  
  
 screen.blit(score, (550, 30))  
 screen.blit(high\_score, (350, 30))  
  
 # Add new obstacle  
 if time.time() - obstacle\_start > minimum\_time + random.randrange(0, 30) / 10:  
 obstacle\_start = time.time()  
 if score\_value > 500.0:  
 ptera\_probability = random.random() # Generate a random float from 0 to 1.0  
 if ptera\_probability < 0.2: # 20% probability that ptera is spawned  
 obstacles.append(Ptera())  
 obstacles[-1].speed = ground.speed + 1  
 else:  
 obstacles.append(Cactus()) # 80% probability of a cactus (duh)  
 obstacles[-1].speed = ground.speed # Synchronise the speed  
 else:  
 obstacles.append(Cactus())  
 obstacles[-1].speed = ground.speed  
  
 if int(score\_value) > 0 and int(score\_value) % 100 == 0 and int(  
 score\_value) % 3 == 0: # Increase game speed after score crosses a multiple of 300  
 ground.speed += 0.25  
 for obstacle in obstacles:  
 if obstacle.is\_cactus:  
 obstacle.speed = ground.speed  
 elif obstacle.is\_ptera:  
 obstacle.speed = ground.speed + 1  
  
 if score\_value > 1 and score\_value % 100 == 0: # Checkpoint sound after score crosses a multiple of 100  
 check\_point.play()  
  
 if dino.hitbox.colliderect(obstacles[0].hitbox): # Collision detection with closest cactus  
 death\_sound.play()  
 dead = True  
 screen.blit(game\_over, (170, 70))  
 screen.blit(replay\_button, (340, 100))  
  
 # print(obstacles)  
 if obstacles[0].x < -30:  
 obstacles.pop(0)  
 if obstacles == []:  
 obstacles.append(Cactus())  
 obstacle\_start = time.time()  
  
 pygame.display.update()  
  
 if dead:  
 del dino  
 del ground  
 del obstacles  
 running = False  
  
  
game()