author\_\_ = "Google Bot"

import os

import sys

import pygame

import random

from pygame import \*

pygame.init()

scr\_size = (width,height) = (600,150)

FPS = 60

gravity = 0.6

black = (0,0,0)

white = (255,255,255)

background\_col = (235,235,235)

high\_score = 0

screen = pygame.display.set\_mode(scr\_size)

clock = pygame.time.Clock()

pygame.display.set\_caption("Google Bot")

jump\_sound = pygame.mixer.Sound('sprites/jump.wav')

die\_sound = pygame.mixer.Sound('sprites/die.wav')

checkPoint\_sound = pygame.mixer.Sound('sprites/checkPoint.wav')

def load\_image(

name,

sizex=-1,

sizey=-1,

colorkey=None,

):

fullname = os.path.join('sprites', name)

image = pygame.image.load(fullname)

image = image.convert()

if colorkey is not None:

if colorkey is -1:

colorkey = image.get\_at((0, 0))

image.set\_colorkey(colorkey, RLEACCEL)

if sizex != -1 or sizey != -1:

image = pygame.transform.scale(image, (sizex, sizey))

return (image, image.get\_rect())

def load\_sprite\_sheet(

sheetname,

nx,

ny,

scalex = -1,

scaley = -1,

colorkey = None,

):

fullname = os.path.join('sprites',sheetname)

sheet = pygame.image.load(fullname)

sheet = sheet.convert()

sheet\_rect = sheet.get\_rect()

sprites = []

sizex = sheet\_rect.width/nx

sizey = sheet\_rect.height/ny

for i in range(0,ny):

for j in range(0,nx):

rect = pygame.Rect((j\*sizex,i\*sizey,sizex,sizey))

image = pygame.Surface(rect.size)

image = image.convert()

image.blit(sheet,(0,0),rect)

if colorkey is not None:

if colorkey is -1:

colorkey = image.get\_at((0,0))

image.set\_colorkey(colorkey,RLEACCEL)

if scalex != -1 or scaley != -1:

image = pygame.transform.scale(image,(scalex,scaley))

sprites.append(image)

sprite\_rect = sprites[0].get\_rect()

return sprites,sprite\_rect

def disp\_gameOver\_msg(retbutton\_image,gameover\_image):

retbutton\_rect = retbutton\_image.get\_rect()

retbutton\_rect.centerx = width / 2

retbutton\_rect.top = height\*0.52

gameover\_rect = gameover\_image.get\_rect()

gameover\_rect.centerx = width / 2

gameover\_rect.centery = height\*0.35

screen.blit(retbutton\_image, retbutton\_rect)

screen.blit(gameover\_image, gameover\_rect)

def extractDigits(number):

if number > -1:

digits = []

i = 0

while(number/10 != 0):

digits.append(number%10)

number = int(number/10)

digits.append(number%10)

for i in range(len(digits),5):

digits.append(0)

digits.reverse()

return digits

class Dino():

def \_\_init\_\_(self,sizex=-1,sizey=-1):

self.images,self.rect = load\_sprite\_sheet('dino.png',5,1,sizex,sizey,-1)

self.images1,self.rect1 = load\_sprite\_sheet('dino\_ducking.png',2,1,59,sizey,-1)

self.rect.bottom = int(0.98\*height)

self.rect.left = width/15

self.image = self.images[0]

self.index = 0

self.counter = 0

self.score = 0

self.isJumping = False

self.isDead = False

self.isDucking = False

self.isBlinking = False

self.movement = [0,0]

self.jumpSpeed = 11.5

self.stand\_pos\_width = self.rect.width

self.duck\_pos\_width = self.rect1.width

def draw(self):

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

def checkbounds(self):

if self.rect.bottom > int(0.98\*height):

self.rect.bottom = int(0.98\*height)

self.isJumping = False

def update(self):

if self.isJumping:

self.movement[1] = self.movement[1] + gravity

if self.isJumping:

self.index = 0

elif self.isBlinking:

if self.index == 0:

if self.counter % 400 == 399:

self.index = (self.index + 1)%2

else:

if self.counter % 20 == 19:

self.index = (self.index + 1)%2

elif self.isDucking:

if self.counter % 5 == 0:

self.index = (self.index + 1)%2

else:

if self.counter % 5 == 0:

self.index = (self.index + 1)%2 + 2

if self.isDead:

self.index = 4

if not self.isDucking:

self.image = self.images[self.index]

self.rect.width = self.stand\_pos\_width

else:

self.image = self.images1[(self.index)%2]

self.rect.width = self.duck\_pos\_width

self.rect = self.rect.move(self.movement)

self.checkbounds()

if not self.isDead and self.counter % 7 == 6 and self.isBlinking == False:

self.score += 1

if self.score % 100 == 0 and self.score != 0:

if pygame.mixer.get\_init() != None:

checkPoint\_sound.play()

self.counter = (self.counter + 1)

class Cactus(pygame.sprite.Sprite):

def \_\_init\_\_(self,speed=5,sizex=-1,sizey=-1):

pygame.sprite.Sprite.\_\_init\_\_(self,self.containers)

self.images,self.rect = load\_sprite\_sheet('cacti-small.png',3,1,sizex,sizey,-1)

self.rect.bottom = int(0.98\*height)

self.rect.left = width + self.rect.width

self.image = self.images[random.randrange(0,3)]

self.movement = [-1\*speed,0]

def draw(self):

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

def update(self):

self.rect = self.rect.move(self.movement)

if self.rect.right < 0:

self.kill()

class Ptera(pygame.sprite.Sprite):

def \_\_init\_\_(self,speed=5,sizex=-1,sizey=-1):

pygame.sprite.Sprite.\_\_init\_\_(self,self.containers)

self.images,self.rect = load\_sprite\_sheet('ptera.png',2,1,sizex,sizey,-1)

self.ptera\_height = [height\*0.82,height\*0.75,height\*0.60]

self.rect.centery = self.ptera\_height[random.randrange(0,3)]

self.rect.left = width + self.rect.width

self.image = self.images[0]

self.movement = [-1\*speed,0]

self.index = 0

self.counter = 0

def draw(self):

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

def update(self):

if self.counter % 10 == 0:

self.index = (self.index+1)%2

self.image = self.images[self.index]

self.rect = self.rect.move(self.movement)

self.counter = (self.counter + 1)

if self.rect.right < 0:

self.kill()

class Ground():

def \_\_init\_\_(self,speed=-5):

self.image,self.rect = load\_image('ground.png',-1,-1,-1)

self.image1,self.rect1 = load\_image('ground.png',-1,-1,-1)

self.rect.bottom = height

self.rect1.bottom = height

self.rect1.left = self.rect.right

self.speed = speed

def draw(self):

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

screen.blit(self.image1,self.rect1)

def update(self):

self.rect.left += self.speed

self.rect1.left += self.speed

if self.rect.right < 0:

self.rect.left = self.rect1.right

if self.rect1.right < 0:

self.rect1.left = self.rect.right

class Cloud(pygame.sprite.Sprite):

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

pygame.sprite.Sprite.\_\_init\_\_(self,self.containers)

self.image,self.rect = load\_image('cloud.png',int(90\*30/42),30,-1)

self.speed = 1

self.rect.left = x

self.rect.top = y

self.movement = [-1\*self.speed,0]

def draw(self):

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

def update(self):

self.rect = self.rect.move(self.movement)

if self.rect.right < 0:

self.kill()

class Scoreboard():

def \_\_init\_\_(self,x=-1,y=-1):

self.score = 0

self.tempimages,self.temprect = load\_sprite\_sheet('numbers.png',12,1,11,int(11\*6/5),-1)

self.image = pygame.Surface((55,int(11\*6/5)))

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

if x == -1:

self.rect.left = width\*0.89

else:

self.rect.left = x

if y == -1:

self.rect.top = height\*0.1

else:

self.rect.top = y

def draw(self):

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

def update(self,score):

score\_digits = extractDigits(score)

self.image.fill(background\_col)

for s in score\_digits:

self.image.blit(self.tempimages[s],self.temprect)

self.temprect.left += self.temprect.width

self.temprect.left = 0

def introscreen():

temp\_dino = Dino(44,47)

temp\_dino.isBlinking = True

gameStart = False

temp\_ground,temp\_ground\_rect = load\_sprite\_sheet('ground.png',15,1,-1,-1,-1)

temp\_ground\_rect.left = width/20

temp\_ground\_rect.bottom = height

logo,logo\_rect = load\_image('logo.png',300,140,-1)

logo\_rect.centerx = width\*0.6

logo\_rect.centery = height\*0.6

while not gameStart:

if pygame.display.get\_surface() == None:

print("Couldn't load display surface")

return True

else:

for event in pygame.event.get():

if event.type == pygame.QUIT:

return True

if event.type == pygame.KEYDOWN:

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

temp\_dino.isJumping = True

temp\_dino.isBlinking = False

temp\_dino.movement[1] = -1\*temp\_dino.jumpSpeed

temp\_dino.update()

if pygame.display.get\_surface() != None:

screen.fill(background\_col)

screen.blit(temp\_ground[0],temp\_ground\_rect)

if temp\_dino.isBlinking:

screen.blit(logo,logo\_rect)

temp\_dino.draw()

pygame.display.update()

clock.tick(FPS)

if temp\_dino.isJumping == False and temp\_dino.isBlinking == False:

gameStart = True

def gameplay():

global high\_score

gamespeed = 4

startMenu = False

gameOver = False

gameQuit = False

playerDino = Dino(44,47)

new\_ground = Ground(-1\*gamespeed)

scb = Scoreboard()

highsc = Scoreboard(width\*0.78)

counter = 0

cacti = pygame.sprite.Group()

pteras = pygame.sprite.Group()

clouds = pygame.sprite.Group()

last\_obstacle = pygame.sprite.Group()

Cactus.containers = cacti

Ptera.containers = pteras

Cloud.containers = clouds

retbutton\_image,retbutton\_rect = load\_image('replay\_button.png',35,31,-1)

gameover\_image,gameover\_rect = load\_image('game\_over.png',190,11,-1)

temp\_images,temp\_rect = load\_sprite\_sheet('numbers.png',12,1,11,int(11\*6/5),-1)

HI\_image = pygame.Surface((22,int(11\*6/5)))

HI\_rect = HI\_image.get\_rect()

HI\_image.fill(background\_col)

HI\_image.blit(temp\_images[10],temp\_rect)

temp\_rect.left += temp\_rect.width

HI\_image.blit(temp\_images[11],temp\_rect)

HI\_rect.top = height\*0.1

HI\_rect.left = width\*0.73

while not gameQuit:

while startMenu:

pass

while not gameOver:

if pygame.display.get\_surface() == None:

print("Couldn't load display surface")

gameQuit = True

gameOver = True

else:

for event in pygame.event.get():

if event.type == pygame.QUIT:

gameQuit = True

gameOver = True

if event.type == pygame.KEYDOWN:

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

if playerDino.rect.bottom == int(0.98\*height):

playerDino.isJumping = True

if pygame.mixer.get\_init() != None:

jump\_sound.play()

playerDino.movement[1] = -1\*playerDino.jumpSpeed

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

if not (playerDino.isJumping and playerDino.isDead):

playerDino.isDucking = True

if event.type == pygame.KEYUP:

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

playerDino.isDucking = False

for c in cacti:

c.movement[0] = -1\*gamespeed

if pygame.sprite.collide\_mask(playerDino,c):

playerDino.isDead = True

if pygame.mixer.get\_init() != None:

die\_sound.play()

for p in pteras:

p.movement[0] = -1\*gamespeed

if pygame.sprite.collide\_mask(playerDino,p):

playerDino.isDead = True

if pygame.mixer.get\_init() != None:

die\_sound.play()

if len(cacti) < 2:

if len(cacti) == 0:

last\_obstacle.empty()

last\_obstacle.add(Cactus(gamespeed,40,40))

else:

for l in last\_obstacle:

if l.rect.right < width\*0.7 and random.randrange(0,50) == 10:

last\_obstacle.empty()

last\_obstacle.add(Cactus(gamespeed, 40, 40))

if len(pteras) == 0 and random.randrange(0,200) == 10 and counter > 500:

for l in last\_obstacle:

if l.rect.right < width\*0.8:

last\_obstacle.empty()

last\_obstacle.add(Ptera(gamespeed, 46, 40))

if len(clouds) < 5 and random.randrange(0,300) == 10:

Cloud(width,random.randrange(height/5,height/2))

playerDino.update()

cacti.update()

pteras.update()

clouds.update()

new\_ground.update()

scb.update(playerDino.score)

highsc.update(high\_score)

if pygame.display.get\_surface() != None:

screen.fill(background\_col)

new\_ground.draw()

clouds.draw(screen)

scb.draw()

if high\_score != 0:

highsc.draw()

screen.blit(HI\_image,HI\_rect)

cacti.draw(screen)

pteras.draw(screen)

playerDino.draw()

pygame.display.update()

clock.tick(FPS)

if playerDino.isDead:

gameOver = True

if playerDino.score > high\_score:

high\_score = playerDino.score

if counter%700 == 699:

new\_ground.speed -= 1

gamespeed += 1

counter = (counter + 1)

if gameQuit:

break

while gameOver:

if pygame.display.get\_surface() == None:

print("Couldn't load display surface")

gameQuit = True

gameOver = False

else:

for event in pygame.event.get():

if event.type == pygame.QUIT:

gameQuit = True

gameOver = False

if event.type == pygame.KEYDOWN:

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

gameQuit = True

gameOver = False

if event.key == pygame.K\_RETURN or event.key == pygame.K\_SPACE:

gameOver = False

gameplay()

highsc.update(high\_score)

if pygame.display.get\_surface() != None:

disp\_gameOver\_msg(retbutton\_image,gameover\_image)

if high\_score != 0:

highsc.draw()

screen.blit(HI\_image,HI\_rect)

pygame.display.update()

clock.tick(FPS)

pygame.quit()

quit()

def main():

isGameQuit = introscreen()

if not isGameQuit:

gameplay()

main()