```
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()
```