# -\*- coding: cp1252 -\*-

# Faz as importações necessarias

#import tela\_inicial

import pygame, math, os

from random import randrange

import random

from pygame.locals import \*

## importa o arquivo que contem as funções basicas

import Funcoes

# Define a tela para ser iniciada sempre no centro

os.environ['SDL\_VIDEO\_CENTERED'] = '1'

#Inicia pygame

pygame.init()

#Inicia a font do pygame

pygame.font.init()

#Inicializa o mixer

#pygame.mixer.pre\_init(44100, 32, 2, 4096)

#ost = pygame.mixer.Sound('resources\\teste.wav')

# Classe que representa a tela

class Tela():

def \_\_init\_\_(self):

self.nomeTela = 'Hangeki'

self.altura = Funcoes.alt\_disp

self.largura = Funcoes.larg\_disp

#self.background = 'resources\\background1.jpg'

#Classe responsavel por gerenciar o jogo

class Game():

def \_\_init\_\_(self):

'''

1 = ativo

2 = pausado

3 = gameOver

'''

self.status = 1

self.score = 0

self.font\_name = 'NasalizationRg-Regular'#pygame.font.get\_default\_font()

self.game\_font = pygame.font.SysFont(self.font\_name, 20)

self.HPAsteroid = 1

self.fase = 1

self.asteroids = 2

self.modo = 1

self.bossWave = 5

#ost.play()

def atualizaScore(self,ponto):

self.score += ponto

# Classe da nave

class Nave(pygame.sprite.Sprite):

def \_\_init\_\_(self):

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

#Carregando imagens das animações pois o blit ferra

self.animSprite0 = pygame.image.load("Nave\_0.png")

self.imagemMatriz = pygame.image.load("Nave\_0.png")

self.imageF = pygame.image.load("Nave\_0.png")

self.image = self.imagemMatriz

self.altura = self.image.get\_height()-4

self.largura = self.image.get\_width()-4

self.angulo = 0

self.x = tela.largura/2

self.y = tela.altura/2

self.speed = 0

self.vidas = 10

self.rect = pygame.Rect(self.x, self.y, self.largura-10, self.altura-10)

self.rady = 0

self.radx = 0

self.anim = 0

self.countAnim = 0

self.tiraVida = False

#metodo que ira atualizar a posição da nave, rotaciona-la e anima-la

def atualiza(self):

self.anim = 0

self.imagemMatriz = self.animSprite0

self.image = self.imagemMatriz

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

rot\_image = pygame.transform.rotate(self.imagemMatriz, self.angulo)

rot\_rect = orig\_rect.copy()

rot\_rect.center = rot\_image.get\_rect().center

self.image = rot\_image.subsurface(rot\_rect).copy()

self.rect = pygame.Rect(self.x, self.y, self.largura, self.altura)

def viraEsquerda(self):

if self.angulo < 45:

self.angulo += 5

def viraDireita(self):

if self.angulo > - 45:

self.angulo -= 5

def shoot(self):

tiros.append(Tiro((tela.largura/2),(tela.altura - nave.altura), self.angulo))

#Classe responsavel por gerenciar os asteroids

class Asteroid(pygame.sprite.Sprite):

def \_\_init\_\_(self):

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

self.tamx = 78

self.tamy = 78

self.imagemMatriz = pygame.image.load("resources\\asteroid.png")

self.imagemMatriz = self.imagemMatriz.convert\_alpha()

self.image = self.imagemMatriz

self.angulo = randrange(360)

self.x = randrange(tela.largura)

self.y = randrange(357)

self.altura = 74

self.largura = 74

self.speed = 2

self.dano = 0

#self.contTiro = 250

self.contTiro = random.randrange(250,500)

self.disparar = self.contTiro

self.rect = pygame.Rect(self.x, self.y, self.largura-10, self.altura-10)

#self.rangeDisp = tela

if self.y > 0:

self.x = -25

def moveAsteroids(self):

real\_angulo=self.angulo+90

#real\_angulo\*=math.pi/180

#rady=-math.sin(real\_angulo)

#radx=math.cos(real\_angulo)

radx=-1

rady= 0

if (rady<0):

self.y+=rady\*self.speed

if (rady>0):

self.y+=+rady\*self.speed

if (radx<0):

self.x+=radx\*self.speed

if (radx>0):

self.x+=+radx\*self.speed

def verificaDisparo(self):

if self.disparar > 0:

self.disparar = self.disparar - 1

if self.disparar == 0:

self.shoot()

self.disparar = self.contTiro

def atualizaAsteroids(self):

self.imagemMatriz = pygame.transform.scale(self.imagemMatriz, (self.tamx,self.tamy) )

self.image = self.imagemMatriz

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

rot\_image = pygame.transform.rotate(self.imagemMatriz, self.angulo)

rot\_rect = orig\_rect.copy()

rot\_rect.center = rot\_image.get\_rect().center

self.image = rot\_image.subsurface(rot\_rect).copy()

if (self.x < -self.largura-self.tamx):

self.x = tela.largura

if (self.x > tela.largura+self.tamx):

self.x = -self.largura

if (self.y < -self.altura-self.tamy):

self.y = tela.altura

if (self.y > tela.altura+self.tamy):

self.y = -self.altura

self.rect = pygame.Rect(self.x, self.y, self.largura-10, self.altura-10)

def shoot(self):

tirosInimigo.append(TiroInimigo((self.x),(self.y), 180))

#Classe responsavel por gerenciar os tiros da nave

class Tiro(pygame.sprite.Sprite):

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

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

self.imagemMatriz = pygame.image.load("resources\\projetil.png")

self.imagemMatriz = self.imagemMatriz.convert\_alpha()

self.image = self.imagemMatriz

self.altura = 31

self.largura = 5

self.angulo = angulo

self.speed = speedrace

self.x = x

self.y = y

self.rect = pygame.Rect(self.x+34, self.y+37, self.largura, self.altura)

def atualiza(self):

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

rot\_image = pygame.transform.rotate(self.imagemMatriz, self.angulo)

rot\_rect = orig\_rect.copy()

rot\_rect.center = rot\_image.get\_rect().center

self.image = rot\_image.subsurface(rot\_rect).copy()

self.rect = pygame.Rect(self.x+34, self.y+37, self.largura, self.altura)

def disparo(self):

real\_angulo=self.angulo+90

real\_angulo\*=math.pi/180

rady=-math.sin(real\_angulo)

radx=math.cos(real\_angulo)

if (rady<0):

self.y+=rady\*self.speed

if (rady>0):

self.y+=+rady\*self.speed

if (radx<0):

self.x+=radx\*self.speed

if (radx>0):

self.x+=+radx\*self.speed

#Classe responsavel por gerenciar os tiros da nave

class TiroInimigo(pygame.sprite.Sprite):

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

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

self.imagemMatriz = pygame.image.load("resources\\projetil.png")

self.imagemMatriz = self.imagemMatriz.convert\_alpha()

self.image = self.imagemMatriz

self.altura = 31

self.largura = 5

self.angulo = angulo

self.speed = 15

self.x = x

self.y = y

self.rect = pygame.Rect(self.x+34, self.y+37, self.largura, self.altura)

def atualiza(self):

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

rot\_image = pygame.transform.rotate(self.imagemMatriz, self.angulo)

rot\_rect = orig\_rect.copy()

rot\_rect.center = rot\_image.get\_rect().center

self.image = rot\_image.subsurface(rot\_rect).copy()

self.rect = pygame.Rect(self.x+34, self.y+37, self.largura, self.altura)

def disparo(self):

real\_angulo=self.angulo+90

real\_angulo\*=math.pi/180

rady=-math.sin(real\_angulo)

radx=math.cos(real\_angulo)

rady = 0.5

if (rady<0):

self.y+=rady\*self.speed

if (rady>0):

self.y+=+rady\*self.speed

if (radx<0):

self.x+=radx\*self.speed

if (radx>0):

self.x+=+radx\*self.speed

'''

a rotina percorre o array dos tiros e percorre para cada tiro o array de asteroids

após isso compara se o retangulo de colisão de um objeto sobrepos o outro.

caso sim e o tamanho do asteroid for 78 (asteroid grande), o mesmo se divide em 4

como o tamanho 50, depois disso deleta o tiro

'''

def colisaoTiros(tiros, asteroids):

delTiro = -1

for i in range(len(tiros)):

for j in range(len(asteroids)):

if (tiros[i].rect.colliderect(asteroids[j].rect)):

asteroids[j].dano += 1

#if (asteroids[j].dano >= game.HPAsteroid):

# if asteroids[j].tamx == 78:

# for x in range(4):

# asteroids.append(Asteroid())

# asteroids[len(asteroids)-1].tamx = 50

# asteroids[len(asteroids)-1].tamy = 50

# asteroids[len(asteroids)-1].x = asteroids[j].x

# asteroids[len(asteroids)-1].y = asteroids[j].y

# asteroids[len(asteroids)-1].speed = 2.5

del asteroids[j]

delTiro = i

break

if delTiro > -1:

del tiros[delTiro]

delTiro = -1

game.atualizaScore(50)

break

def colisaoTirosIni(tiros, tirosImimigo):

delTiro = -1

for i in range(len(tiros)):

for j in range(len(tirosInimigo)):

if (tiros[i].rect.colliderect(tirosInimigo[j].rect)):

del tirosInimigo[j]

delTiro = i

break

if delTiro > -1:

del tiros[delTiro]

delTiro = -1

game.atualizaScore(25)

break

#Caso os tiros saiam da tela eles são deletados para n consumir tanta memoria

def removeTiros(tiros):

for tiro in range(len(tiros)):

if(tiros[tiro].y > tela.altura+tiros[tiro].altura

or tiros[tiro].y < 0 - tiros[tiro].altura

or tiros[tiro].x > tela.largura

or tiros[tiro].x < -74):

del tiros[tiro]

break

#Caso os tiros saiam da tela eles são deletados para n consumir tanta memoria

def removeTirosInim(tirosInimigo):

for tiroI in range(len(tirosInimigo)):

if(tirosInimigo[tiroI].y > tela.altura+tirosInimigo[tiroI].altura

or tirosInimigo[tiroI].y < 0 - tirosInimigo[tiroI].altura

or tirosInimigo[tiroI].x > tela.largura

or tirosInimigo[tiroI].x < -74):

del tirosInimigo[tiroI]

break

#Função main do jogo

def main(altura,largura,controle):

'''

Declaração de variaveis

criação da tela com a largura e altura baseadas no txt com os parametros

as variaveis globais são utilizadas em outros metodos

declara os arrays de tiros e de asteroids

estancia o Boss

declara os ticks para a criação de asteroids e dos tiros

carrega a imagem de background

define o clock para fazer o controle de fps do jogo

cria variaves para o controle

'''

gravaScore = False

BLACK = (0,0,0)

global tela

tela = Tela()

tela.largura = largura

tela.altura = altura

global screen

screen = pygame.display.set\_mode((tela.largura,tela.altura),0,32)

pygame.display.set\_caption(tela.nomeTela)

background = BLACK

global nave

nave = Nave()

global game

game = Game()

global tiros

tiros = []

global tirosInimigo

tirosInimigo = []

game.controle = controle

tamTiros = 0

tick\_shoot = 0

tick\_asteroid = 0

estFase = False

contAsteroids = 0

tamTiros = 0

clock = pygame.time.Clock()

asteroids = []

global speedrace

speedrace = 15

#background = pygame.image.load(tela.background).convert()

#background = pygame.transform.scale(background, (tela.largura,tela.altura) )

# se o status do jogo for ativo ou pausado

while game.status in(1,2):

#define FPS para ser 60

clock.tick(60)

pause = False

if not estFase:

if game.modo == 1:

if tick\_asteroid <= 0:

tick\_asteroid = 35

asteroids.append(Asteroid())

contAsteroids += 1

else:

if tick\_asteroid > 0:

tick\_asteroid -= 1

if contAsteroids == game.asteroids:

estFase = True

for event in pygame.event.get():

#se clicar em fechar ele troca o status e fecha o jogo

if event.type == pygame.QUIT:

#screen.fill(BLACK)

game.status = 4

quit()

tecla\_pressionada = pygame.key.get\_pressed()

#Verifica disparo dos inimigos

dispInimigo = len(asteroids)

if dispInimigo-1 >= 0:

for i in asteroids:

i.verificaDisparo()

#caso aperte espaço ou aperte o botao 3 cria um tiro

if tecla\_pressionada[K\_SPACE]:

if tick\_shoot <= 0:

tick\_shoot = 15

nave.shoot()

else:

if tick\_shoot > 0:

tick\_shoot -= 1

if tecla\_pressionada[K\_LEFT] and game.modo == 1:

nave.viraEsquerda()

if tecla\_pressionada[K\_RIGHT] and game.modo == 1:

nave.viraDireita()

if tecla\_pressionada[K\_UP] and game.modo == 1:

speedrace += 5

if tecla\_pressionada[K\_DOWN] and game.modo == 1:

speedrace -= 5

screen.fill(BLACK)

#desenha os tiros

tamTiros = len(tiros)

if tamTiros-1 >= 0:

for i in range(tamTiros):

tiros[i].atualiza()

tiros[i].disparo()

screen.blit(tiros[i].image, (tiros[i].x, tiros[i].y))

#desenha os tiros inimigo

tamTirosI = len(tirosInimigo)

if tamTirosI-1 >= 0:

for i in range(tamTirosI):

tirosInimigo[i].atualiza()

tirosInimigo[i].disparo()

if tirosInimigo[i].y > tela.altura:

nave.vidas = nave.vidas - 1

screen.blit(tirosInimigo[i].image, (tirosInimigo[i].x, tirosInimigo[i].y))

#atualiza a posição da nave e a desenha

nave.atualiza()

screen.blit(nave.image, (tela.largura/2, tela.altura - nave.altura))

#desenha os asteroids

if len(asteroids)-1 >= 0:

for i in range(len(asteroids)):

asteroids[i].moveAsteroids()

asteroids[i].atualizaAsteroids()

screen.blit(asteroids[i].image, (asteroids[i].x, asteroids[i].y))

#verifica a colição dos asteroids com os tiros

colisaoTiros(tiros,asteroids)

colisaoTirosIni(tiros,tirosInimigo)

#verifica se necessario deletar os tiros do inimigo e da nave

removeTiros(tiros)

removeTirosInim(tirosInimigo)

#HUD

scoreText = game.game\_font.render('Score: ' + str(game.score), 1, (255, 255, 255))

screen.blit(scoreText, (10, 5))

#vidasText = game.game\_font.render('Vidas: ' + str(nave.vidas), 1, (255, 255, 255))

#screen.blit(vidasText, (250, 5))

waveText = game.game\_font.render('Wave: ' + str(game.fase), 1, (255, 255, 255))

screen.blit(waveText, (400, 5))

asteroidsText = game.game\_font.render('Aviões: ' + str(len(asteroids)), 1, (255, 255, 255))

screen.blit(asteroidsText, (550, 5))

pygame.display.flip()

#if nave.vidas <= 0:

# quit()

#Caso acabem os asteroids e não seja uma batalha de boss cria novos asteroids

#acrescenta uma vida para o jogador

if(len(asteroids) == 0 and estFase):

game.fase += 1

game.asteroids += 2

nave.vidas += 1

estFase = False

contAsteroids = 0

destroyBoss = False

if game.status == 4:

pygame.quit

if \_\_name\_\_ == "\_\_main\_\_":

#lê arquivo e atribui os valores nas variaveis

Funcoes.lerArquivo()

main(600,800,'aaa')