# implementation of Spaceship - program template for RiceRocks

import simplegui

import math

import random

# globals for user interface

WIDTH = 800

HEIGHT = 600

score = 0

lives = 3

time = 0

level\_up = 20

started = False

class ImageInfo:

def \_\_init\_\_(self, center, size, radius = 0, lifespan = None, animated = False):

self.center = center

self.size = size

self.radius = radius

if lifespan:

self.lifespan = lifespan

else:

self.lifespan = float('inf')

self.animated = animated

def get\_center(self):

return self.center

def get\_size(self):

return self.size

def get\_radius(self):

return self.radius

def get\_lifespan(self):

return self.lifespan

def get\_animated(self):

return self.animated

# art assets created by Kim Lathrop, may be freely re-used in non-commercial projects, please credit Kim

# debris images - debris1\_brown.png, debris2\_brown.png, debris3\_brown.png, debris4\_brown.png

# debris1\_blue.png, debris2\_blue.png, debris3\_blue.png, debris4\_blue.png, debris\_blend.png

debris\_info = ImageInfo([320, 240], [640, 480])

debris\_image = simplegui.load\_image("http://commondatastorage.googleapis.com/codeskulptor-assets/lathrop/debris2\_blue.png")

# nebula images - nebula\_brown.png, nebula\_blue.png

nebula\_info = ImageInfo([400, 300], [800, 600])

nebula\_image = simplegui.load\_image("http://commondatastorage.googleapis.com/codeskulptor-assets/lathrop/nebula\_blue.f2014.png")

# splash image

splash\_info = ImageInfo([200, 150], [400, 300])

splash\_image = simplegui.load\_image("http://commondatastorage.googleapis.com/codeskulptor-assets/lathrop/splash.png")

# ship image

ship\_info = ImageInfo([45, 45], [90, 90], 35)

ship\_image = simplegui.load\_image("http://commondatastorage.googleapis.com/codeskulptor-assets/lathrop/double\_ship.png")

# missile image - shot1.png, shot2.png, shot3.png

missile\_info = ImageInfo([5,5], [10, 10], 3, 50)

missile\_image = simplegui.load\_image("http://commondatastorage.googleapis.com/codeskulptor-assets/lathrop/shot2.png")

# asteroid images - asteroid\_blue.png, asteroid\_brown.png, asteroid\_blend.png

asteroid\_info = ImageInfo([45, 45], [90, 90], 40)

asteroid\_image = simplegui.load\_image("http://commondatastorage.googleapis.com/codeskulptor-assets/lathrop/asteroid\_blue.png")

# animated explosion - explosion\_orange.png, explosion\_blue.png, explosion\_blue2.png, explosion\_alpha.png

explosion\_info = ImageInfo([64, 64], [128, 128], 17, 24, True)

explosion\_image = simplegui.load\_image("http://commondatastorage.googleapis.com/codeskulptor-assets/lathrop/explosion\_alpha.png")

# sound assets purchased from sounddogs.com, please do not redistribute

# .ogg versions of sounds are also available, just replace .mp3 by .ogg

soundtrack = simplegui.load\_sound("http://commondatastorage.googleapis.com/codeskulptor-assets/sounddogs/soundtrack.mp3")

missile\_sound = simplegui.load\_sound("http://commondatastorage.googleapis.com/codeskulptor-assets/sounddogs/missile.mp3")

missile\_sound.set\_volume(.5)

ship\_thrust\_sound = simplegui.load\_sound("http://commondatastorage.googleapis.com/codeskulptor-assets/sounddogs/thrust.mp3")

explosion\_sound = simplegui.load\_sound("http://commondatastorage.googleapis.com/codeskulptor-assets/sounddogs/explosion.mp3")

# alternative upbeat soundtrack by composer and former IIPP student Emiel Stopler

# please do not redistribute without permission from Emiel at http://www.filmcomposer.nl

#soundtrack = simplegui.load\_sound("https://storage.googleapis.com/codeskulptor-assets/ricerocks\_theme.mp3")

# helper functions to handle transformations

def angle\_to\_vector(ang):

return [math.cos(ang), math.sin(ang)]

def dist(p, q):

return math.sqrt((p[0] - q[0]) \*\* 2 + (p[1] - q[1]) \*\* 2)

# Ship class

class Ship:

def \_\_init\_\_(self, pos, vel, angle, image, info):

self.pos = [pos[0], pos[1]]

self.vel = [vel[0], vel[1]]

self.thrust = False

self.angle = angle

self.angle\_vel = 0

self.image = image

self.image\_center = info.get\_center()

self.image\_size = info.get\_size()

self.radius = info.get\_radius()

############################################################################

def get\_position(self):

return self.pos

def get\_radius(self):

return self.radius

############################################################################

def draw(self,canvas):

if self.thrust:

canvas.draw\_image(self.image, [self.image\_center[0] + self.image\_size[0], self.image\_center[1]] , self.image\_size,

self.pos, self.image\_size, self.angle)

else:

canvas.draw\_image(self.image, self.image\_center, self.image\_size,

self.pos, self.image\_size, self.angle)

# canvas.draw\_circle(self.pos, self.radius, 1, "White", "White")

def update(self):

# update angle

self.angle += self.angle\_vel

# update position

self.pos[0] = (self.pos[0] + self.vel[0]) % WIDTH

self.pos[1] = (self.pos[1] + self.vel[1]) % HEIGHT

# update velocity

if self.thrust:

acc = angle\_to\_vector(self.angle)

self.vel[0] += acc[0] \* .1

self.vel[1] += acc[1] \* .1

self.vel[0] \*= .99

self.vel[1] \*= .99

def set\_thrust(self, on):

self.thrust = on

if on:

ship\_thrust\_sound.rewind()

ship\_thrust\_sound.play()

else:

ship\_thrust\_sound.pause()

def increment\_angle\_vel(self):

self.angle\_vel += .05

def decrement\_angle\_vel(self):

self.angle\_vel -= .05

def shoot(self):

global a\_missile

forward = angle\_to\_vector(self.angle)

missile\_pos = [self.pos[0] + self.radius \* forward[0], self.pos[1] + self.radius \* forward[1]]

missile\_vel = [self.vel[0] + 6 \* forward[0], self.vel[1] + 6 \* forward[1]]

a\_missile = Sprite(missile\_pos, missile\_vel, self.angle, 0, missile\_image, missile\_info, missile\_sound)

############################################################################

missile\_sprite = a\_missile

missile\_group.add(missile\_sprite)

############################################################################

# Sprite class

class Sprite:

def \_\_init\_\_(self, pos, vel, ang, ang\_vel, image, info, sound = None):

self.pos = [pos[0],pos[1]]

self.vel = [vel[0],vel[1]]

self.angle = ang

self.angle\_vel = ang\_vel

self.image = image

self.image\_center = info.get\_center()

self.image\_size = info.get\_size()

self.radius = info.get\_radius()

self.lifespan = info.get\_lifespan()

self.animated = info.get\_animated()

self.age = 0

if sound:

sound.rewind()

sound.play()

############################################################################

def get\_position(self):

return self.pos

def get\_radius(self):

return self.radius

def draw(self, canvas):

global time

if self.animated == True:

current\_rock\_index = (time % self.image\_size[0]) // 1

current\_rock\_center = [self.image\_center[0] + current\_rock\_index \* self.image\_size[0], self.image\_center[1]]

canvas.draw\_image(self.image, current\_rock\_center, self.image\_size,

self.pos, self.image\_size, self.angle)

time += 1

else:

canvas.draw\_image(self.image, self.image\_center, self.image\_size,

self.pos, self.image\_size, self.angle)

############################################################################

def update(self):

# update angle

self.angle += self.angle\_vel

# update position

self.pos[0] = (self.pos[0] + self.vel[0]) % WIDTH

self.pos[1] = (self.pos[1] + self.vel[1]) % HEIGHT

############################################################################

self.age += 1

if self.age >= self.lifespan:

return True

else:

return False

############################################################################

############################################################################

#other object will always be the ship

def collide(self, other\_object):

if dist(self.pos, other\_object.get\_position()) <= (self.radius + other\_object.get\_radius()):

return True

else:

return False

############################################################################

# key handlers to control ship

def keydown(key):

if key == simplegui.KEY\_MAP['left']:

my\_ship.decrement\_angle\_vel()

elif key == simplegui.KEY\_MAP['right']:

my\_ship.increment\_angle\_vel()

elif key == simplegui.KEY\_MAP['up']:

my\_ship.set\_thrust(True)

elif key == simplegui.KEY\_MAP['space']:

my\_ship.shoot()

def keyup(key):

if key == simplegui.KEY\_MAP['left']:

my\_ship.increment\_angle\_vel()

elif key == simplegui.KEY\_MAP['right']:

my\_ship.decrement\_angle\_vel()

elif key == simplegui.KEY\_MAP['up']:

my\_ship.set\_thrust(False)

# mouseclick handlers that reset UI and conditions whether splash image is drawn

def click(pos):

global started

center = [WIDTH / 2, HEIGHT / 2]

size = splash\_info.get\_size()

inwidth = (center[0] - size[0] / 2) < pos[0] < (center[0] + size[0] / 2)

inheight = (center[1] - size[1] / 2) < pos[1] < (center[1] + size[1] / 2)

if (not started) and inwidth and inheight:

started = True

############################################################################

soundtrack.play()

timer.start()

############################################################################

def draw(canvas):

global time, started, lives, score

# animiate background

time += 1

wtime = (time / 4) % WIDTH

center = debris\_info.get\_center()

size = debris\_info.get\_size()

canvas.draw\_image(nebula\_image, nebula\_info.get\_center(), nebula\_info.get\_size(), [WIDTH / 2, HEIGHT / 2], [WIDTH, HEIGHT])

canvas.draw\_image(debris\_image, center, size, (wtime - WIDTH / 2, HEIGHT / 2), (WIDTH, HEIGHT))

canvas.draw\_image(debris\_image, center, size, (wtime + WIDTH / 2, HEIGHT / 2), (WIDTH, HEIGHT))

# draw UI

canvas.draw\_text("Lives", [50, 50], 22, "White")

canvas.draw\_text("Score", [680, 50], 22, "White")

canvas.draw\_text(str(lives), [50, 80], 22, "White")

canvas.draw\_text(str(score), [680, 80], 22, "White")

# draw and update ship

my\_ship.draw(canvas)

my\_ship.update()

# draw splash screen if not started

if not started:

canvas.draw\_image(splash\_image, splash\_info.get\_center(),

splash\_info.get\_size(), [WIDTH / 2, HEIGHT / 2],

splash\_info.get\_size())

############################################################################

# group

process\_sprite\_group(rock\_group, canvas)

process\_sprite\_group(missile\_group, canvas)

process\_sprite\_group(explosion\_group, canvas)

if group\_collide(rock\_group, my\_ship):

lives -= 1

if group\_group\_collide(missile\_group, rock\_group):

score += 10

if lives <= 0:

started = False

timer.stop()

for r in rock\_group:

rock\_group.discard(r)

lives = 3

score = 0

my\_ship.pos = [WIDTH / 2, HEIGHT / 2]

my\_ship.vel = [0, 0]

soundtrack.rewind()

############################################################################

############################################################################

# timer handler that spawns a rock

def rock\_spawner():

global a\_rock, rock\_group, my\_ship, score, level\_up

rock\_pos = [random.randrange(0, WIDTH), random.randrange(0, HEIGHT)]

#Experiment with varying the velocity of rocks based on the score

if score >= level\_up:

rock\_vel = [random.random() \* .6 + score\*0.01 , random.random() \* .6 + score\*0.01]

rock\_avel = random.random() \* .2

else:

rock\_vel = [random.random() \* .6 - .3, random.random() \* .6 - .3]

rock\_avel = random.random() \* .2 - .1

a\_rock = Sprite(rock\_pos, rock\_vel, 0, rock\_avel, asteroid\_image, asteroid\_info)

rock\_sprite = a\_rock

if len(rock\_group) < 12 and dist(rock\_sprite.pos, my\_ship.pos) >= (rock\_sprite.radius + my\_ship.radius) + 50 :

rock\_group.add(rock\_sprite)

# take a set and a canvas and call the update and draw methods for each sprite in the group

def process\_sprite\_group(sprite\_group, canvas):

for s in sprite\_group:

s.draw(canvas)

if s.update() == True:

sprite\_group.remove(s)

# check for collisions between other\_object and elements of the group

# other object will always be your ship

def group\_collide(sprite\_group, other\_object):

global explosion\_group

for s in sprite\_group:

if s.collide(other\_object) == True:

sprite\_group.remove(s)

#explosion animation

explosion = Sprite(s.get\_position(), [-1,1], 0, 0, explosion\_image, explosion\_info, explosion\_sound)

explosion\_sprite = explosion

explosion\_group.add(explosion\_sprite)

return True

# destroy rocks when they are hit by a missile

def group\_group\_collide(group1, group2):

for m in group1:

if group\_collide(group2, m) == True:

group1.discard(m)

return True

############################################################################

# initialize stuff

frame = simplegui.create\_frame("Asteroids", WIDTH, HEIGHT)

# initialize ship and two sprites

my\_ship = Ship([WIDTH / 2, HEIGHT / 2], [0, 0], 0, ship\_image, ship\_info)

a\_rock = Sprite([WIDTH / 3, HEIGHT / 3], [1, 1], 0, .1, asteroid\_image, asteroid\_info)

a\_missile = Sprite([2 \* WIDTH / 3, 2 \* HEIGHT / 3], [-1,1], 0, 0, missile\_image, missile\_info, missile\_sound)

explosion = Sprite([WIDTH / 2, HEIGHT / 2], [-1,1], 0, 0, explosion\_image, explosion\_info, explosion\_sound)

# register handlers

frame.set\_keyup\_handler(keyup)

frame.set\_keydown\_handler(keydown)

frame.set\_mouseclick\_handler(click)

frame.set\_draw\_handler(draw)

timer = simplegui.create\_timer(1000.0, rock\_spawner)

############################################################################

soundtrack.play()

soundtrack.set\_volume(0.5)

# set

rock\_group = set([])

missile\_group = set([])

explosion\_group = set([])

# label

frame.add\_label('Control the spaceship to destroy asteroids before they strike your spaceship.')

frame.add\_label('')

frame.add\_label('controls spaceship:')

frame.add\_label('rotate right: "clockwise"')

frame.add\_label('rotate left: "counterclockwise"')

frame.add\_label('thrust forward: "up"')

frame.add\_label('fire shots: "space"')

frame.add\_label('')

############################################################################

# get things rolling

frame.start()