ENGR 133 LC4-05

Pole Position

Written By:

Alec Pannunzio

Team Members:

Jonathan, jdufresn@purdue.edu

Sam, graha205@purdue.edu

James, long365@purdue.edu

# Introduction:

I chose to write a pole-position game for a few reasons. Firstly, games are fun, and I have always loved wondering how video games were made and trying to make them for myself. Secondly, one of my interests in terms of a career path is to work on autonomous cars as a computer engineer. One of the main ways current autonomous cars determine what is the road and what is not is through various image processing techniques. The CPU for my pole-position game also uses image processing to drive itself autonomously.

Function Descriptions / Descriptions of Different Project Modules and Algorithms

# Graphics Engine (Pygame):

I wrote this project on the pygame graphics engine to establish a base to work on. Pygame allows me to transform images and display them to the screen in a timely manner.

#### The deformation problem:

When I first started working with pygame I found that there is a problem with pygame’s scale and rotation transformations. When an image is scaled down, it is shrunk down to fit inside a rectangular bounding box of the target scaling dimensions. This works great until the image is scaled *and* rotated *and* you want the picture to rotate about its center. The reasons that this problem originates is a bit long winded for this report but essentially what ended up happening is as the image rotated its corners stuck out from the bounding box, which meant that pygame further shrunk the image past the desired scale. What I did to solve this problem is manually calculate how far the corners will stick out and compensate by scaling the image to be that much bigger. The implementation of this is in Sprite.updateDisplayImage(self)

#### The Coordinate System problem:

Another issue I ran into with pygame is the way that it managed its coordinate system, although it stems from the computer graphics convention of the positive y direction being towards the bottom of the screen rather that the top of the screen, consistent with most math and physics axis. This means that sine and cosine will dictate that positive rotation is *clockwise* rather than counterclockwise, which is the physics and math convention. Despite this, pygame made the design decision to make positive rotation still *counterclockwise*, meaning that pygame rotation and sine and cosine calculations disagree. I fixed this issue by popping a negative in front of the rotation of my game objects before using pygame’s rotation method, but it sure was confusing before I discovered the discrepancy and got everything straightened out.

# Game Engine:

Since I was trying to program a game, I figured a good place to start would be to write a game engine. This, (surprisingly enough) is in the folder “game\_engine”. The game engine class hierarchy is as follows:

A picture containing text, black, night sky

Description automatically generated

To work with the game engine, a game developer would create a child classes of GameObject or any of its children (most likely Sprites and RelativeSprites) and an instance of the ObjectDraw class. They would then use ObjectDraw’s add() method to add the child classes of GameObject to the ObjectDraw instance. They then would create a loop at the end of their code like this:

while(not objectDraw.done):

    objectDraw.run(); # run the game engine stuff

Which will run all the game engine processes. This is implemented in PolePositionRunner.py

Anything game-specific processes that the game designer wants done will likely be put in the update method of child classes which follow the following template:

# updates the object values. is call repetively by the update loop

    def update(self):

        super(*<classname>,*self).update();

# do class-specific stuff here

#### First-person mode:

I wanted to do first-person mode right. Not by just rotating a picture around the center point and calling it a day but by actually implementing a system of cameras, where Sprites could be shown from the perspective of another Object2D. I did this with the RelativeSprite class. If a RelativeSprite does not have a camera assigned it will be shown at the exact position and rotation that its *xPosition, yPosition,* and *rotation* class members reflect (Exactly as if it was a normal Sprite instance). However, if the setCamera() method is called with another Object2D, then the RelativeSprite will be displayed onscreen from the perspective of that Object2D. This is done such that if a RelativeSprite has itself as its camera, it will be shown unrotated and at the center of the screen. The bulk of this algorithm is in RelativeSprite.update() and RelativeSprite.paint().

# Pole Position:

Pole position is essentially all the .py files outside of other folders.

polePositionRunner.py

Racetrack.py

Racecar.py

Playercar.py

Computercar.py

A picture containing dark, night sky

Description automatically generated

## Computercar Driving:

First. I passed the track image thorough my team’s edge-detection program from our team project. The Computercar then uses my raytracing function (raytracing.raytrace()) in conjunction with Racetrack.isTrackEdge() function to figure out where it is on the track. Racetrack’s isTrackEdge looks at the edge-detection output and if the point it is given is a bright pixel on the edges map it will return True. Then, the CPU algorithm compares the lengths of rays straight sideways and at 45-degree angles forward from sideways to come up with a value representing where it is on the track. This value is fed into a PD calculation as the error which determines the corrective steering output.

Additional descriptions of functions

Every user defined function has a description commented on the line above the function declaration in the code summarizing it. Additionally, every class has a description at the top of the file that class is declared in underneath the imports, and detailed descriptions of its class members and their types are listed directly below that.

User manual

To get pole position running on your computer, you must first install the [Python 3.9 IDLE](https://www.python.org/downloads/) on your computer. Make sure that you also get the Python functions for the command line, since you will need them to get Pygame. *Note: Pole Position will not run on anaconda without modification.* Then, you must open your command line. (On windows, press the windows key, type “cmd” and hit enter) Type the following commands in your command line:

pip install numpy

pip install pygame

pip install matplotlib

If this doesn’t work try running the command line as an administrator, and check that you have the correct Python version.



Graphical user interface, application

Description automatically generated



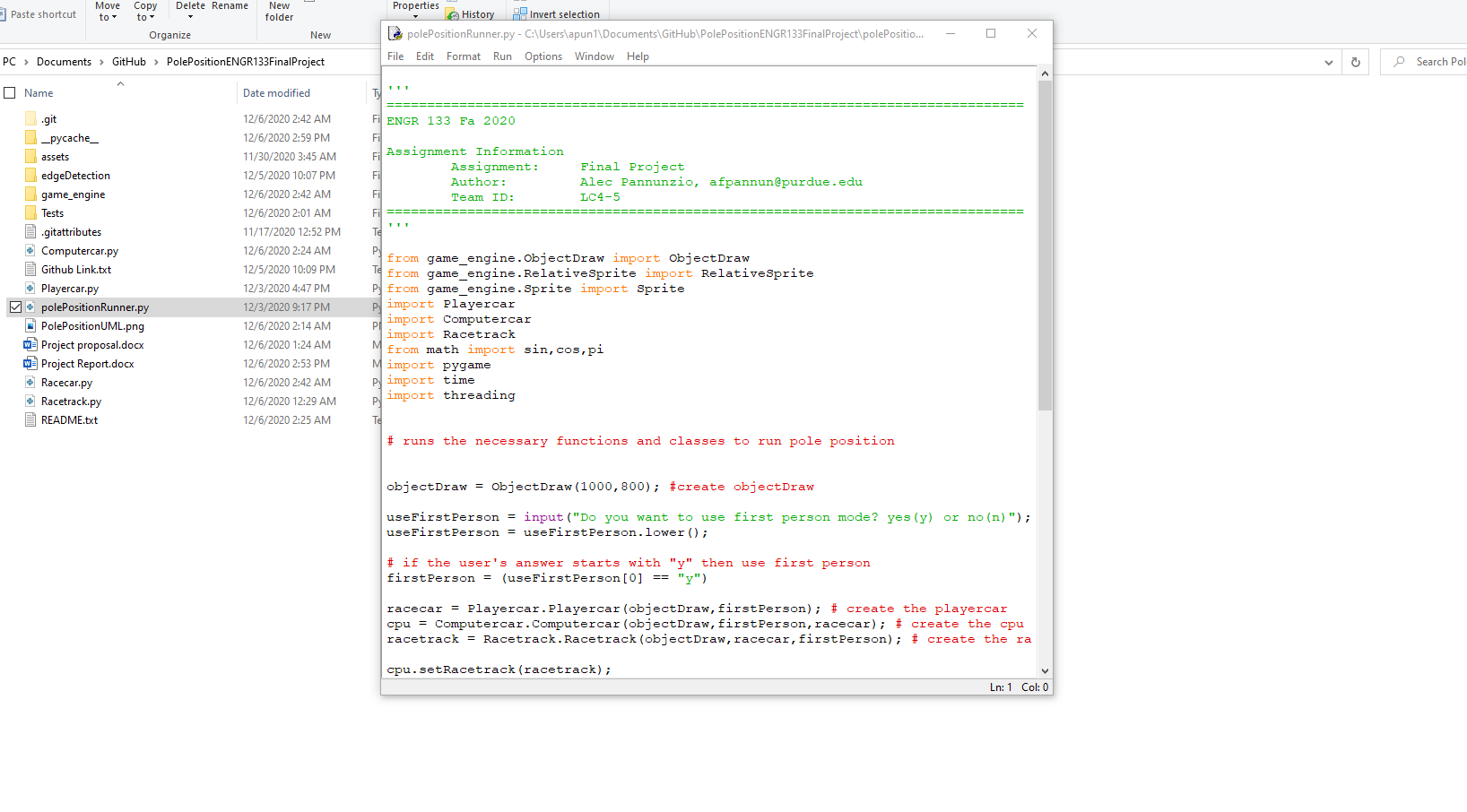
If you have additional problems installing pygame, consult [this guide](https://www.pygame.org/wiki/GettingStarted).

Now that you have installed all the libraries, you can run the program!

Now go into the project files and find polePositionRunner. Should be in the top level of the file structure. Right click on it and select “Edit with IDLE”. Alternatively, you could try double-clicking it, but this does not always run the correct version of Python.

If the “Edit with IDLE” option doesn’t work, try opening Python IDLE 3.9 fist and either opening it directly from the IDLE or try right-clicking again.

It should look like this when you are all done.

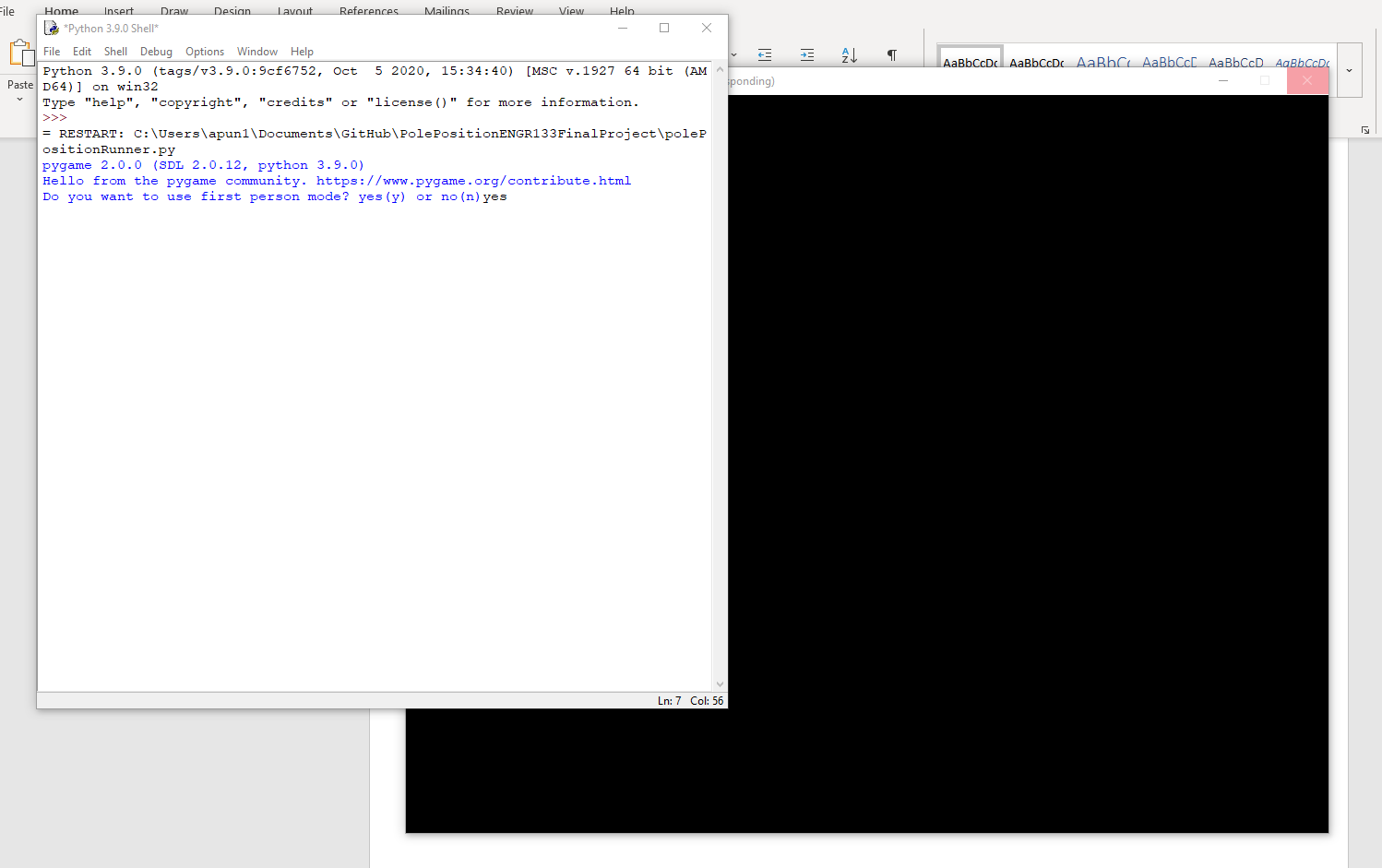


Once your screen looks like that, you just have to press F5 to run the program! (Or alternatively go to Run->open module)

Note: The files must be in the correct folder structure to work properly. Download the project as it is supposed to be [here](https://github.com/samiam567/PolePositionENGR133FinalProject). (Github repo)

Once you run the program, the python shell will pop up along with a pygame window. On the shell you will be prompted, “Do you want to use first person mode? yes(y) or no(n)”. I would *highly* recommend trying both, but it doesn’t really matter which one you chose first. Type either y for first person mode or n for third person mode. Alternatively, if you type *anything* that starts with y it will run first person mode, andanything that doesn’t start with y will run third person mode.

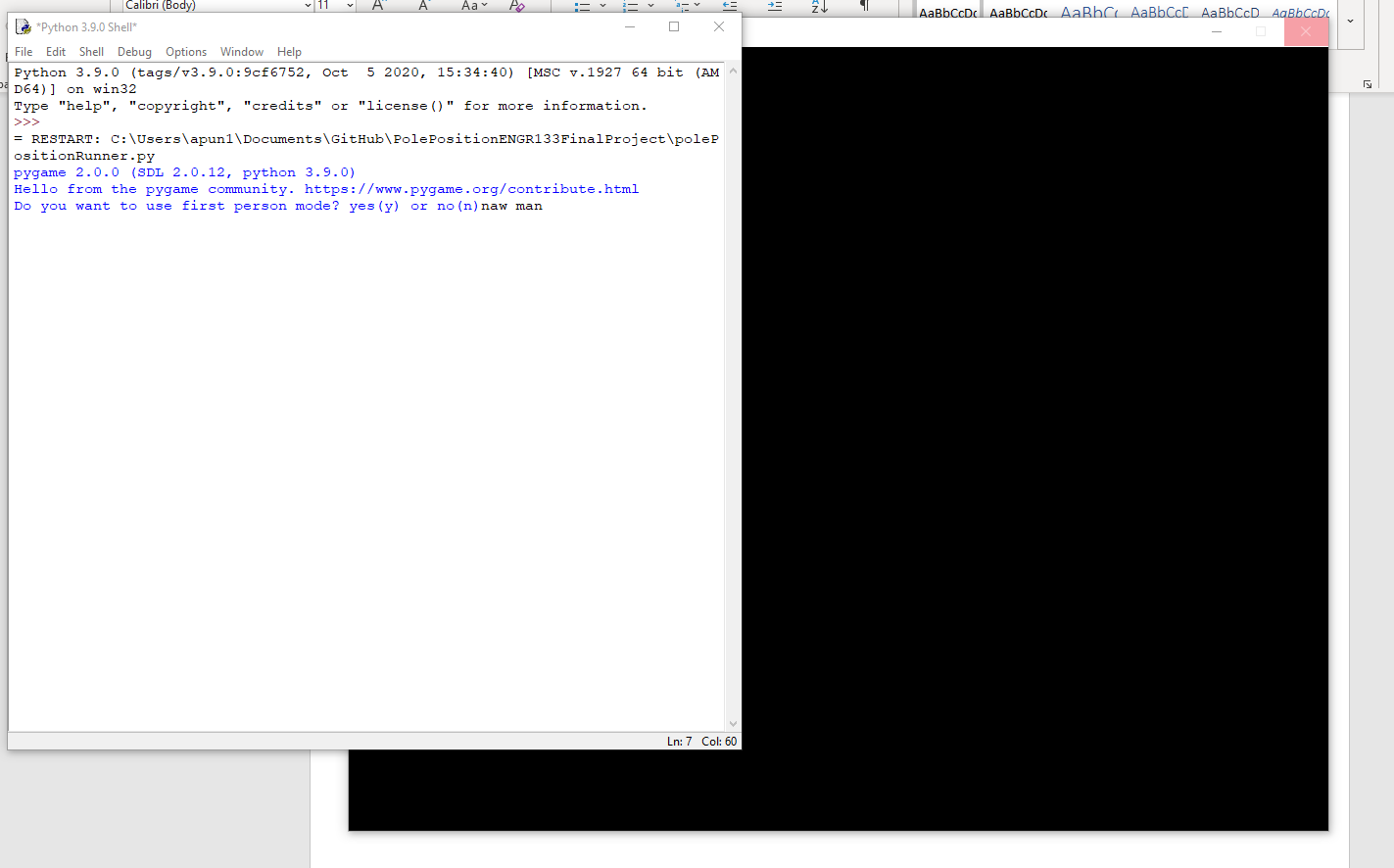
Typing yes:



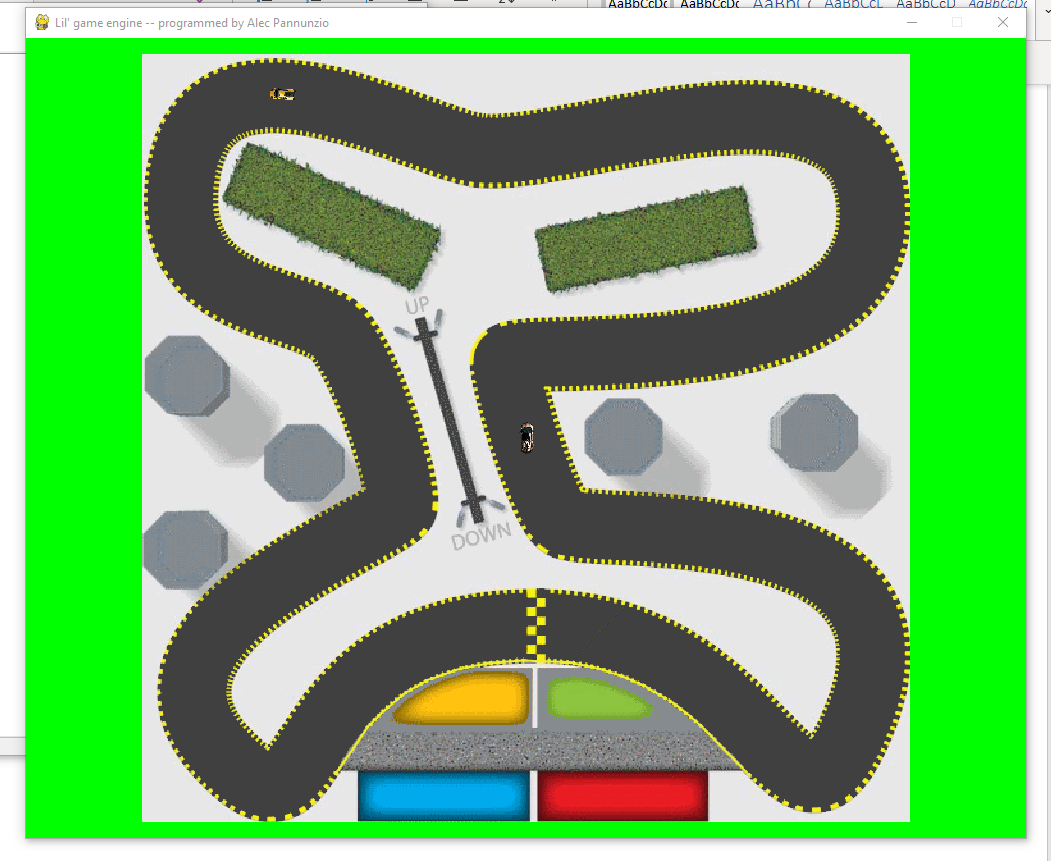
Will get you here \/



Typing no:



Will get you here:



At this point, press w to accelerate, s to slow down, a to turn left, and d to turn right. The shell will notify you if you go off the track. Have fun!

I made the CPU in first-person mode beatable, but if you can beat the CPU in third-person mode I would be very impressed, then I would promptly make the CPU drive faster and beat you 😊.

Code

## Game Engine:

#### GameEngineToolbox:

'''

===============================================================================

ENGR 133 Fa 2020

Assignment Information

    Assignment:     Final Project

    Author:         Alec Pannunzio, afpannun@purdue.edu

    Team ID:        LC4-5

===============================================================================

'''

# a toolbox full of useful functions for the game engine

#checks that the passed variable is of the passed required type or types. Returns True if it is, false otherwise

def checkType(variable, requiredTypes, errorMessage = "Incorrect type", raiseError = True):

    rightType = False;

    if type(requiredTypes) is tuple:

        for requiredType in requiredTypes:

            if (type(variable) is requiredType):

                rightType = True;

        if (raiseError):

            if not rightType:

                raise TypeError(errorMessage);

        return rightType;

    else:

        if not (type(variable) is requiredTypes):

            if (raiseError):

                raise TypeError(errorMessage);

            return False;

        else:

            return True;

# will make all objects bounce off the borders of the screen

def border\_bounce(objectDraw):

    for current\_object in objectDraw.objects:

        if (current\_object.getPosition()[0] > objectDraw.screenSizeX-current\_object.xSize/2):

            current\_object.setSpeed((-1 \* abs(current\_object.getSpeed()[0]),current\_object.getSpeed()[1]));

        if (current\_object.getPosition()[1] < current\_object.ySize/2):

            current\_object.setSpeed((current\_object.getSpeed()[0],abs(current\_object.getSpeed()[1])));

        if (current\_object.getPosition()[0] < current\_object.xSize/2):

            current\_object.setSpeed((abs(current\_object.getSpeed()[0]),current\_object.getSpeed()[1]));

        if (current\_object.getPosition()[1] > objectDraw.screenSizeY-current\_object.ySize/2):

            current\_object.setSpeed((current\_object.getSpeed()[0],-1 \* abs(current\_object.getSpeed()[1])));

'''

===============================================================================

ACADEMIC INTEGRITY STATEMENT

    I have not used source code obtained from any other unauthorized

    source, either modified or unmodified. Neither have I provided

    access to my code to another. The project I am submitting

    is my own original work.

===============================================================================

'''

#### GameObject:

'''

===============================================================================

ENGR 133 Fa 2020

Assignment Information

    Assignment:     Final Project

    Author:         Alec Pannunzio, afpannun@purdue.edu

    Team ID:        LC4-5

===============================================================================

'''

# an object that can be used in the game engine. has very general properties

'''

class members

name - string - the name of the GameObject

'''

from game\_engine.GameEngineToolbox import checkType;

class GameObject():

    #initialize the GameObject

    def \_\_init\_\_(self,name):

        self.name = str(name); #store the name as one of our class members

    # returns the gameObject's name

    def getName(self):

        return self.name;

    # this is called by the engine thread and child classes will override this

    def update(self):

        pass;

    # this is called by the engine thread and child classes will override this

    def paint(self):

        pass;

# tests this class

def testGameObject():

    bob = GameObject("bob");

    print(bob.getName());

    badboi = GameObject(69);

    print(badboi.getName());

'''

===============================================================================

ACADEMIC INTEGRITY STATEMENT

    I have not used source code obtained from any other unauthorized

    source, either modified or unmodified. Neither have I provided

    access to my code to another. The project I am submitting

    is my own original work.

===============================================================================

'''

#### Object2D:

'''

===============================================================================

ENGR 133 Fa 2020

Assignment Information

    Assignment:     Final Project

    Author:         Alec Pannunzio, afpannun@purdue.edu

    Team ID:        LC4-5

===============================================================================

'''

from game\_engine.GameObject import GameObject

from game\_engine.GameEngineToolbox import checkType

# Object2D is a template for an object that can be transformed around the screen

'''

class members:

xPosition - float - the position of the center of this object on the x axis

yPosition - float - the position of the center of this object the y axis

xSpeed - float - the speed of the object in the x direction

ySpeed - float - the speed of the object in the y direction

xAcceleration - float - the acceleration of the object in the x direction

yAcceleration - float - the acceleration of the object in the y direction

rotation - float - the rotation of this object (in radians)

angularVelocity - float - the speed at which the object is rotating

xSize - int - the size of this object in the x dimension

ySize - int - the size of this object in the y dimension

'''

class Object2D(GameObject):

    # initialize the object

    def \_\_init\_\_(self,name,xPosition,yPosition,xSize,ySize):

        super(Object2D,self).\_\_init\_\_(name);

        #check types

        checkType(xPosition,(int,float),"xPosition must be an number");

        checkType(yPosition,(int,float),"yPosition must be an number");

        checkType(xSize,(int,float),"xSize must be an number");

        checkType(ySize,(int,float),"ySize must be an number");

        # set the passed parameters to their respective class members

        self.xPosition = xPosition;

        self.yPosition = yPosition;

        self.xSize = xSize;

        self.ySize = ySize;

        # initialize the rest of the class members to zero

        self.xSpeed = 0;

        self.ySpeed = 0;

        self.xAcceleration = 0;

        self.yAcceleration = 0;

        self.rotation = 0.0;

        self.angularVelocity = 0;

    # updates the object values. is call repetively by the update loop

    def update(self):

        super(Object2D,self).update();

        self.xPosition += self.xSpeed;

        self.yPosition += self.ySpeed;

        self.xSpeed += self.xAcceleration;

        self.ySpeed += self.yAcceleration;

        self.rotation += self.angularVelocity;

    # sets the position of the object

    def setPosition(self,xPos, yPos):

        checkType(xPos,(int,float),"xPosition must be an int");

        checkType(yPos,(int,float),"yPosition must be an int");

        self.xPosition = xPos;

        self.yPosition = yPos;

    # sets the position of the object

    def setPosition(self, position):

        checkType(position,tuple,"position must be a tuple");

        self.xPosition = position[0];

        self.yPosition = position[1];

    # sets the speed of the object

    def setSpeed(self,xSpeed,ySpeed):

        checkType(xSpeed,(int,float),"xSpeed must be a number");

        checkType(ySpeed,(int,float),"ySpeed must be a number");

        self.xSpeed = xSpeed;

        self.ySpeed = ySpeed;

    # sets the speed of the object

    def setSpeed(self,speed):

        checkType(speed,tuple,"speed must be an tuple");

        self.xSpeed = speed[0];

        self.ySpeed = speed[1];

    # sets the acceleration of the object

    def setAcceleration(self,xAccel,yAccel):

        checkType(xAccel,(int,float),"xAcceleration must be a number");

        checkType(yAccel,(int,float),"yAcceleration must be a number");

        self.xAcceleration = xAccel;

        self.yAcceleration = yAccel;

    # sets the acceleration of the object

    def setAcceleration(self,acceleration):

        checkType(acceleration,tuple,"acceleration must be an tuple");

        self.xAcceleration = acceleration[0];

        self.yAcceleration = acceleration[1];

    #sets the rotation of the object

    def setRotation(self,rot):

        checkType(rot,(int,float),"rotation must be a number");

        self.rotation = rot;

    # sets the angular velocity of the object

    def setAngularVelocity(self,angV):

        checkType(angV,(int,float),"angularVelocity must be a number");

        self.angularVelocity = angV;

    # returns a list containing the [xPosition, yPosition]

    def getPosition(self):

        return (self.xPosition,self.yPosition);

    # returns a list containing the [xSpeed, ySpeed]

    def getSpeed(self):

        return (self.xSpeed, self.ySpeed);

    # returns a list containing the [xAcceleration, yAcceleration]

    def getAcceleration(self):

        return (self.xAcceleration, self.yAcceleration);

    # returns the rotation

    def getRotation(self):

        return self.rotation;

    # returns the angular velocity

    def getAngularVelocity(self):

        return self.angularVelocity;

# tests this class

def testObject2D():

    bob = Object2D("bob",0,0,10,10);

    print(bob.getName());

    bob.setSpeed(1,2);

    bob.setAcceleration(2,1);

    bob.update();

    bob.update();

    print(bob.getPosition());

    print(bob.getSpeed());

'''

===============================================================================

ACADEMIC INTEGRITY STATEMENT

    I have not used source code obtained from any other unauthorized

    source, either modified or unmodified. Neither have I provided

    access to my code to another. The project I am submitting

    is my own original work.

===============================================================================

'''

#### ObjectDraw:

'''

===============================================================================

ENGR 133 Fa 2020

Assignment Information

    Assignment:     Final Project

    Author:         Alec Pannunzio, afpannun@purdue.edu

    Team ID:        LC4-5

===============================================================================

'''

import pygame

import sys

from threading import Thread

from game\_engine.Sprite import Sprite;

from game\_engine.GameEngineToolbox import checkType;

from game\_engine.GameObject import GameObject;

# Controller for the game engine. Updates and draws the objects on a Pygame window

'''

Class members:

screenSizeX - int - x size of the screen

screenSizeY - int - y size of the screeb

WHITE - tuple - represents the color white

BLACK - tuple - represents the color black

screen - pygame.surface - the screen we are using to put objects on

clock - Clock - the pygame clock. We will use this to set delays

done - boolean - whether the engine is done running yet. Setting this to false will stop the engine

objects - list[GameObject] (manually enforced) - a list of object for the engine to

tickSpeed - int - analogous to frames per second. (1/wait time inbetween frames)

keysPressed - list[boolean] - the keys that are currently pressed down as fetched by pygame.key.get\_pressed()

backgroundColor - tuple - the background color for the screen. Painted before all other objects

'''

class ObjectDraw():

    #initializes the objectdraw

    def \_\_init\_\_(self,screenSizeX, screenSizeY):

        #make sure types check out and assign screen size class members

        checkType(screenSizeX,int,"screenSizeX must be an int");

        checkType(screenSizeY,int,"screenSizeY must be an int");

        self.screenSizeX = screenSizeX;

        self.screenSizeY = screenSizeY;

        #define colors

        self.WHITE = (255,255,255)

        self.BLACK = (0,0,0)

        #initialize pygame

        pygame.init();

        self.screen = pygame.display.set\_mode([screenSizeX, screenSizeY]) # get the screen

        pygame.display.set\_caption("Lil' game engine -- programmed by Alec Pannunzio"); #set the title of our display

        self.clock = pygame.time.Clock(); # get the clock

        #set up variables

        self.objects = []; # create the objects list

        self.done = True;

        self.tickSpeed = 60;

        self.keysPressed = [];

        self.backgroundColor = self.WHITE;

    # sets tickSpeed

    def setTickSpeed(self,newTick):

        checkType(newTick,int,"tickSpeed must be an integer");

        self.tickSpeed = newTick;

    #sets the background color

    def setBackgroundColor(self,newColor):

        checkType(newColor,tuple,"backgroundColor must be a tuple");

        self.backgroundColor = newColor;

    # will start the game engine

    def start(self):

        if (self.done): #make sure we aren't already running

            self.done = False;

        else:

            print("engine already running");

    # will pause the game engine

    def pause(self):

        self.done = True;

    #will stop the game engine

    def stop(self):

        print("stopping");

        self.done = True;

        pygame.quit();

    #should only be called by start(), runs the game engine loop

    def run(self):

        if (True):

            #set keysPressed

            self.keysPressed = pygame.key.get\_pressed();

            # handle pygame events

            for event in pygame.event.get():

                if event.type == pygame.QUIT:

                    print("shutting down");

                    #shut 'er down

                    done = True;

                    pygame.display.quit();

                    pygame.quit();

                    sys.exit();

                    return;

                elif event.type == pygame.MOUSEBUTTONDOWN: # when the mouse is clicked

                    pass;

            #update the objects

            for current\_object in self.objects:

                current\_object.update();

            # draw background

            self.screen.fill(self.backgroundColor)

            #paint the objects

            for current\_object in self.objects:

                current\_object.paint(self.screen);

            pygame.display.flip(); #push the updates to the display

            self.clock.tick(self.tickSpeed); # delay for a bit inbetween frames

    # will add the gameObject to the game engine to be run

    def add(self,gameObject):

        assert issubclass(gameObject.\_\_class\_\_, GameObject); # make sure the object we are adding is a child of GameObject

        self.objects.append(gameObject);

    #returns the keys that the user has pressed

    def getKeysPressed(self):

        return self.keysPressed;

'''

===============================================================================

ACADEMIC INTEGRITY STATEMENT

    I have not used source code obtained from any other unauthorized

    source, either modified or unmodified. Neither have I provided

    access to my code to another. The project I am submitting

    is my own original work.

===============================================================================

'''

#### Raytrace:

'''

===============================================================================

ENGR 133 Fa 2020

Assignment Information

    Assignment:     Final Project

    Author:         Alec Pannunzio, afpannun@purdue.edu

    Team ID:        LC4-5

===============================================================================

'''

from game\_engine.GameEngineToolbox import checkType

from math import pi,sin,cos

'''

sends a ray starting from the given position and at the given angle and returns the distance when the passed function is true or the max length is exeeded

params:

startpos - tuple[float] - the position to start the ray at

angle - float - the angle (in degrees)

length - float - the maximum length to probe

function - function - stop when this function returns functionOutput

functionOutput - boolean - stop when the function output equals this

increment - float - how much to increment the probing length each time

'''

def raytrace(startpos,angle,length,function,functionOutput,increment = 1.3):

    checkType(startpos,(tuple),"starting position must be a tuple");

    checkType(startpos[0],(float,int),"start pos must contain numbers");

    checkType(angle,(float,int),"angle must be a number");

    checkType(length,(int,float),"ray length must be a number");

   # checkType(function,(type(raytrace)),"function must be of type function");

    checkType(increment,(float,int),"increment must be a number");

    angle = pi\*angle/180; # convert angle to radians

    cLength = 0; # the current probe-length for the ray

    # set the position of the ray to the startposition

    x = startpos[0];

    y = startpos[1];

    # while we have not exceeded the length and the function did not return true

    while ( (not (function(x,y) == functionOutput) ) and (cLength < length) ):

        cLength += increment; # increment the length of the ray

        # calculate the pos of the endpoint of the ray

        x = startpos[0] + cLength \* cos(angle);

        y = startpos[1] + cLength \* sin(angle);

    return cLength

'''

===============================================================================

ACADEMIC INTEGRITY STATEMENT

    I have not used source code obtained from any other unauthorized

    source, either modified or unmodified. Neither have I provided

    access to my code to another. The project I am submitting

    is my own original work.

===============================================================================

'''

## Sprite:

'''

===============================================================================

ENGR 133 Fa 2020

Assignment Information

    Assignment:     Final Project

    Author:         Alec Pannunzio, afpannun@purdue.edu

    Team ID:        LC4-5

===============================================================================

'''

from game\_engine.Object2D import Object2D

from game\_engine.GameEngineToolbox import checkType;

from math import sin, cos, pi, atan, tan

import pygame

# an image that can be moved and rotated

'''

class members:

imgSource - string - the source of the image

img - pygame.Surface - the image object

displayImg - pygame.Surface - the transformed image object ready to be displayed

showSizeX - int - the size the image is transformed to in the x dimension. (when the image rotates the corners would stick out so we have to allow a bigger box for the picture to fit in in the transformation)

showSizeY - int - the size the image is transformed to in the y dimension. (when the image rotates the corners would stick out so we have to allow a bigger box for the picture to fit in in the transformation)

imgScale - float - how many times bigger this surface is than the original image

'''

class Sprite(Object2D):

    def \_\_init\_\_(self,name,xPosition,yPosition,scaling,imgSource):

        checkType(scaling,(int,float),"the scaling factor must be a int or float");

        #calculate the size of the image

        self.img = pygame.image.load(imgSource).convert(); # load the image from the imgSource

        self.xSize = scaling \* self.img.get\_width();

        self.ySize = scaling \* self.img.get\_height();

        super(Sprite,self).\_\_init\_\_(name,xPosition,yPosition,self.xSize,self.ySize);

        self.img.set\_colorkey((0,0,0)); # set the colorkey of the image to black

        self.displayImg = self.img; #initialize displayImg

        # initialize some class members

        self.showSizeX = self.xSize;

        self.showSizeY = self.ySize;

        self.imgScale = scaling;

    # updates the sprite and readies it for rendering

    def updateDisplayImage(self):

        # rotate the image to the correct rotation

        self.displayImg = pygame.transform.rotate(self.img,-self.rotation);

        '''

        scale the image

        (compensate for the corners sticking out from the bounding box by scaling the image that much larger)

        '''

        # calculate how far away from the center the corners would be

        cornerAngle = atan(self.ySize/self.xSize);

        cornerDist = ((self.ySize/2)\*\*2 + (self.xSize/2)\*\*2)\*\*0.5; #the length from the center of the rectangle to the corner. Serves as the "hypotenuse"

        # X dimension

        showSizeX1Multi = abs(cos(cornerAngle + pi\*(self.rotation)/180)); #corner @ 45 degrees if unrotated

        showSizeX2Multi = abs(cos(-cornerAngle + pi\*(self.rotation)/180)); #corner @ -45 degrees if unrotated

        #use the larger of the two

        if (showSizeX1Multi > showSizeX2Multi):

            showSizeXMulti = showSizeX1Multi;

        else:

             showSizeXMulti = showSizeX2Multi;

        # Y dimension

        showSizeY1Multi = abs(sin(cornerAngle + pi\*(self.rotation)/180)); #corner @ 45 degrees if unrotated

        showSizeY2Multi = abs(sin(-cornerAngle + pi\*(self.rotation)/180)); #corner @ -45 degrees if unrotated

        #use the larger of the two

        if (showSizeY1Multi > showSizeY2Multi):

            showSizeYMulti = showSizeY1Multi;

        else:

            showSizeYMulti = showSizeY2Multi;

        #calculate the size we should scale the picture to

        self.showSizeX = int(2\*cornerDist\*showSizeXMulti);

        self.showSizeY = int(2\*cornerDist\*showSizeYMulti);

        self.displayImg = pygame.transform.scale(self.displayImg, (self.showSizeX,self.showSizeY));

    #updates this Sprite

    def update(self):

        super(Sprite,self).update(); # call the update of the parent class

        self.updateDisplayImage();

    #rotates the picture, but not the actual sprite object

    def rotatePicture(self,angle):

        self.img = pygame.transform.rotate(self.img,angle);

        self.xSize = self.imgScale \* self.img.get\_width();

        self.ySize = self.imgScale \* self.img.get\_height();

    def paint(self,screen):

        # paint the Sprite, adjusting so that the xPosition,yPosition are the coordinates of the center

        screen.blit(self.displayImg,[self.xPosition-self.showSizeX/2, self.yPosition-self.showSizeY/2]);

'''

===============================================================================

ACADEMIC INTEGRITY STATEMENT

    I have not used source code obtained from any other unauthorized

    source, either modified or unmodified. Neither have I provided

    access to my code to another. The project I am submitting

    is my own original work.

===============================================================================

'''

## RelativeSprite

'''

===============================================================================

ENGR 133 Fa 2020

Assignment Information

    Assignment:     Final Project

    Author:         Alec Pannunzio, afpannun@purdue.edu

    Team ID:        LC4-5

===============================================================================

'''

from game\_engine.Object2D import Object2D

from game\_engine.ObjectDraw import ObjectDraw

from game\_engine.GameEngineToolbox import checkType

from game\_engine.Sprite import Sprite

from math import atan2, sqrt, sin, cos, pi

# a more advanced sprite that can be painted in the perspective of another object. Basically a simple way of doing cameras

# NOTE: this object will act as a normal Sprite unless you set the camera with setCamera().

# NOTE: if it is a first person game it is customary to set all of the gameobjects' cameras to the RelativeSprite representing the character, INCLUDING setting the camera of the character to itself.

'''

class members:

hasCamera - bool - whether the RelativeSprite has a camera assigned to it. If this is false this object will be painted the same as a normal Sprite.

camera - Object2D - the object that we will paint relative to

zeroXPosition - float - offSets the sprites x position by this amount

zeroYPosition - float - offSets the sprites y position by this amount

zeroRotation - float - offsets the sprites rotation by this amount

displayXPosition - float - the x position that we will display (relative to the camera)

displayYPosition - float - the y position that we will display (relative to the camera)

displayRotation - float - the rotation that we will display (relative to the camera)

'''

class RelativeSprite(Sprite):

    def \_\_init\_\_(self,name,xPosition,yPosition,scaling,imgSource,objectDraw):

        super(RelativeSprite,self).\_\_init\_\_(name,xPosition,yPosition,scaling,imgSource);

        checkType(objectDraw,ObjectDraw,"objectDraw must be an ObjectDraw");

        self.hasCamera = False;

        self.objectDraw = objectDraw;

        self.camera = None;

        self.zeroXPosition = self.objectDraw.screenSizeX/2;

        self.zeroYPosition = self.objectDraw.screenSizeY/2;

        self.zeroRotation = 0;

        self.displayXPosition = xPosition;

        self.displayYPosition = yPosition;

        self.displayRotation = 0;

    # sets the camera of this object to the passed Object2D

    def setCamera(self,camera):

        assert issubclass(camera.\_\_class\_\_, Object2D); # make sure the object we are adding is a child of Object2D

        self.camera = camera;

        self.hasCamera = True;

    # removes the camera from the object so it paints like a normal Sprite

    def removeCamera(self):

        self.camera = None;

        self.hasCamera = False;

    # calls the update method of superclasses and updates the displayimage to be in the perspective of the camera

    def update(self):

        if not self.hasCamera:

            super(RelativeSprite,self).update(); # update like a normal Sprite

            self.displayXPosition = self.xPosition;

            self.displayYPosition = self.yPosition;

        else:

            super(Sprite,self).update(); # call the update method of Object2D, NOT Sprite

            #save current position

            prevRotation = self.rotation;

            self.displayXPosition = self.xPosition;

            self.displayYPosition = self.yPosition;

            cameraPosition = self.camera.getPosition();

            #translate relative to the camera

            self.displayXPosition -= cameraPosition[0];

            self.displayYPosition -= cameraPosition[1];

            '''

            rotate around camera

            '''

            #convert to polar

            angle = atan2(self.displayYPosition,self.displayXPosition); # the angle of this object relative to the camera

            radius = sqrt(self.displayXPosition\*\*2 + self.displayYPosition\*\*2) # thie distance this object is away from the camera

            # rotate the object around the camera <the camera's rotation>

            angle -= pi\*self.camera.getRotation()/180;

            # convert back to rectangular and assign to displayPosition

            self.displayXPosition = radius \* cos(angle);

            self.displayYPosition = radius \* sin(angle);

            self.rotation -= self.camera.getRotation(); # add internal rotation to match camera

            #add zero offsets

            self.rotation += self.zeroRotation;

            self.displayXPosition += self.zeroXPosition;

            self.displayYPosition += self.zeroYPosition;

            #set displayRotation

            self.displayRotation = self.rotation;

            '''

            update the display image with the relative values

            '''

            super(RelativeSprite,self).updateDisplayImage(); # update the display image with the relative values

            '''

            reset rotation back to where it was before

            '''

            self.rotation = prevRotation;

            # we don't have to reset position since we used displayPosition rather than directly changing the object's position

    #paint relative to the camera

    def paint(self,screen):

        if self.hasCamera:

            # if we have a camera paint using the relative values

            screen.blit(self.displayImg,[self.displayXPosition-self.showSizeX/2, self.displayYPosition-self.showSizeY/2]);

        else:

            # otherwise just use Sprite's paint method

            super(RelativeSprite,self).paint(screen);

    # set the zero position of the relativeSprite

    def setZeroPosition(self,zeroX,zeroY):

        checkType(zeroX,(int,float),"zero position must be a number");

        checkType(zeroY,(int,float),"zero position must be a number");

        self.zeroXPosition = zeroX;

        self.zeroYPosiiton = zeroY;

    #set the zero rotation of the relativeSprite

    def setZeroRotation(self,zeroRot):

        checkType(zeroRot,(int,float),"zeroRotation must be a number");

        self.zeroRotation = zeroRot;

'''

===============================================================================

ACADEMIC INTEGRITY STATEMENT

    I have not used source code obtained from any other unauthorized

    source, either modified or unmodified. Neither have I provided

    access to my code to another. The project I am submitting

    is my own original work.

===============================================================================

'''