Python DeCal

Week 10

WEDNESDAY

https://forms.gle/M3H9mE0aHTwWd85y9

Announcements

- Good Job with the presentations last week!

- We want to make sure you're on track so your newest homework assignment will be to write about what you've done so far (Due Next Wednesday)! **You should start coding this week if you haven't already**

- DO NOT SAVE YOUR PROJECT FOR THE WEEK IT'S DUE, YOU WILL NOT FINISH IF YOU DO THIS

Breakout Rooms

- Little different than usual, really just QnA about projects
- While waiting, discuss the Yilun's lecture on solving ODEs numerically. Why did we talk about that, how is it useful?

Simulations!

- This is a more fun type of lecture to see some cool shit you can do with python!

EXAMPLES

Solar System

What yours could look like if you are doing this for your project

Key Principles

- Each Time step you need to increment all the relevant values that change with time
- Could be: position, velocity, acceleration, phase, other values that change with time (maybe mass or luminosity?)
- Need to have a way of solving for these values at each new time step
- Need to continue incrementing time steps until sufficient time is passed

(While Loop?)

The Main Loop

- The main loop is the loop everything runs inside of, if the loop isn't running your simulation isn't running

  Five Key Points in a Main Loop:

  - Get relevant values (**x**, **v**, **a**, **t**, etc...)
  - Calculate Changes From Physics
  - Apply Changes to old values (**v** + **dv**) or (**x** + **dx**) and (**t** + **dt**)
  - Draw Results
  - Repeat

Let's Make a Game

Step 1: Make your objects

I like to do this in the form of objects and classes if there is going to be lots of them.

```
class Vector:
    def __init__(self, x, y):
        self.x = x
        self.y = y
    def len(self):
        return math.sqrt(self.x*self.x + self.y*self.y)
    def __add__(self, other):
        return Vector(self.x + other.x, self.y + other.y)
    def __sub__(self, other):
        return Vector(self.x - other.x, self.y - other.y)
    def __mul__(self, other):
        return Vector(self.x * other, self.y * other)
    def __rmul__(self, other):
        return Vector(self.x * other, self.y * other)
    def __truediv__(self, other):
        return Vector(self.x / other, self.y / other)
    def angle(self):
        return math.atan2(self.y, self.x)
    def norm(self):
        if self.x == 0 and self.y == 0:
            return Vector(0, 0)
        return self / self.len()
    def dot(self, other):
        return self.x*other.x + self.y*other.y

class Paddle:
    def __init__(self,x,y):
        self.position = Vector(x, y - 30)
        self.velocity = Vector(0, 0)
        self.width = 20
        self.height = 100

class Ball:
    def __init__(self,x,y):
        self.radius = 10
        self.color = white
        self.position = Vector(x,y)
        self.velocity = Vector(0,0)

ball = Ball(400,250)
R_Paddle = Paddle(750, 250)
L_Paddle = Paddle(50, 250)
Paddles = [L_Paddle, R_Paddle]
```

Let's Make a Game

Step 2: Initialize Everything

There are different ways of doing this, I found making a main function is the best way

In this example, I used a library called pygame to handle all the screen drawing, but you can use whatever tool you want (matplotlib, vpython, etc....)

```
import pygame
from pygame.locals import *
import random, math, sys

pygame.init()
#initialize and create some variables
width, height = 800, 500
Surface = pygame.display.set_mode((800,500))
pygame.display.set_caption("Pong")
black = (0,0,0)
white = (255, 255, 255)
red = (255, 0, 0)
blue = (0,0,200)
```

```
def main():
    score_p = 0
    score_c = 0
    speed = ball.velocity.len()
    while True:
        GetInput()
        Move()
        Collision()
        score()
        Draw()


if __name__ == '__main__': main()
```

Let’s Make a Game

Step 3: Get input (optional)

If you wanted to make a playable game, then you would need to find a way of getting input from the user. This can be done in pygame as well.

For a simulation, this is not at all needed, so skip this slide unless interested

```
def GetInput():
    keystate = pygame.key.get_pressed()
    for event in pygame.event.get():
        if event.type == pygame.QUIT or keystate[pygame.K_ESCAPE]:
            pygame.quit(); sys.exit()
        elif event.type == pygame.KEYDOWN:
            if event.key == pygame.K_UP:
                R_Paddle.velocity.y = -10
            elif event.key == pygame.K_DOWN:
                R_Paddle.velocity.y = 10
            elif event.key == pygame.K_SPACE:
                ball.velocity.x = random.randrange(-15,15)
                ball.velocity.y = random.randrange(-15,15)
        elif event.type == pygame.KEYUP:
            if event.key == pygame.K_UP:
                R_Paddle.velocity.y = 0
            elif event.key == pygame.K_DOWN:
                R_Paddle.velocity.y = 0
```

Let’s Make a Game

Step 4: Physics functions!

Write helper functions that handle the load of using physics to calculate what happens in each frame

This example has a function to handle collisions and a function to handle movement

```
def Collision():
    Circle = ball
    if Circle.position.y < Circle.radius and Circle.velocity.y < 0:
        Circle.velocity.y *= -1
    if Circle.position.y > height-Circle.radius and Circle.velocity.y > 0:
        Circle.velocity.y *= -1
    if Circle.position.x < L_Paddle.position.x + L_Paddle.width + ball.radius/2 and Circle.velocity.x < 0:
        if Circle.position.y > L_Paddle.position.y and Circle.position.y < L_Paddle.position.y + L_Paddle.height:
            Circle.velocity.x *= -1
    if Circle.position.x > R_Paddle.position.x - R_Paddle.width + ball.radius/2 and Circle.velocity.x > 0:
        if Circle.position.y > R_Paddle.position.y and Circle.position.y < R_Paddle.position.y + R_Paddle.height:
            Circle.velocity.x *= -1


def Move():
    for pad in Paddles:
        pad.position = pad.position + pad.velocity
    ball.position = ball.position + ball.velocity

    L_Paddle.velocity.y = ball.velocity.y * 0.95
```

Let’s Make a Game

Step 5: Have a way of keeping score

In your simulation, you might want to track the number of times a specific condition is met.

Example: How many times does a pendulum reach a certain height

Example: How many times do two asteroids collide?

```
def score():
    global score_p
    global score_c
    score_p = 0
    score_c = 0
    if ball.position.x < L_Paddle.position.x:
        score_p += 1
        ball.position.x = 400
        ball.position.y = 250
        speed = 0
    elif ball.position.x > R_Paddle.position.x:
        score_c += 1
        ball.position.x = 400
        ball.position.y = 250
        speed = 0
```

Let’s Make a Game

Step 6: Drawing the simulation

This can be vastly different depending on what library you use to draw. I used PyGame but you can use matplotlib or vpython, etc...

```
def Draw():
    clock = pygame.time.Clock()
    Surface.fill(black)
    pygame.draw.rect(Surface, red, [R_Paddle.position.x,R_Paddle.position.y,R_Paddle.width, R_Paddle.height])
    pygame.draw.rect(Surface, blue, [L_Paddle.position.x,L_Paddle.position.y,L_Paddle.width, L_Paddle.height])
    pygame.draw.circle(Surface,ball.color,(int(ball.position.x),int(ball.position.y)),ball.radius)
    font = pygame.font.Font('batmfa__.ttf', 18)
    text = font.render('Computer: {0:1.0f}'.format(score_c), True, blue,black)
    textRect = text.get_rect()
    textRect.center = (210,25)
    Surface.blit(text, textRect)
    font2 = pygame.font.Font('batmfa__.ttf', 18)
    text2 = font.render('Player: {0:1.0f}'.format(score_p), True, red,black)
    textRect2 = text2.get_rect()
    textRect2.center = (590,25)
    Surface.blit(text2, textRect2)
    pygame.display.flip()
```

Result!

Friday

https://forms.gle/tCFHeynrt6YYtuJy9

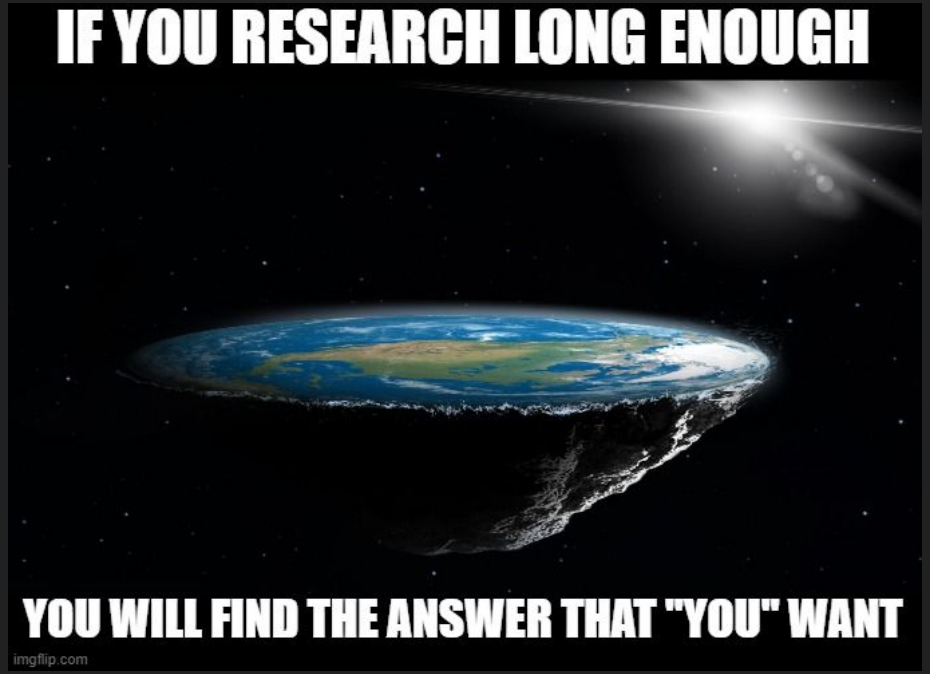
Announcements

- Don’t forget the interim report - DUE NEXT WEDNESDAY
- Keep working on your final project lolol

- Don’t wait till OHs for help… Send us an email if you need. We can arrange meetings etc.
  - “Yilun Does Not Sleep”
- The earlier you have a problem solved the better.

Several checkboxes for research / projects…

- CODING
  - Well, we all know about this
- LITERATURE
  - Sorry but we all have to read
  - Where do we find papers to read…
  - How to find the ones I want
- DATA
  - welp …
  - I want to use a table from some paper
  - What if I don’t have a telescope



Literature

- arXiv.org
  - Updating everyday
  - Preprints of papers
  - Many different topics

- Astrophysics Data System (ADS)
  - https://ui.adsabs.harvard.edu/classic-form
  - Customise your search
  - Generating bibliography
  - All astronomers use it!

Data Acquisition

- **VizieR**
  - https://vizier.u-strasbg.fr/viz-bin/VizieR
  - Catalogues, tables from papers, everything!
- **SIMBAD**
  - http://simbad.u-strasbg.fr/simbad/
  - Database for basic information about a target
- **Gaia archive**
  - https://gea.esac.esa.int/archive/
  - Astrometry, photometry
  - Galactic astronomy
- Get your own data lolol
- You can use astropy to query the data
  - `astroquery`
  - https://astroquery.readthedocs.io/en/latest/index.html

gaia