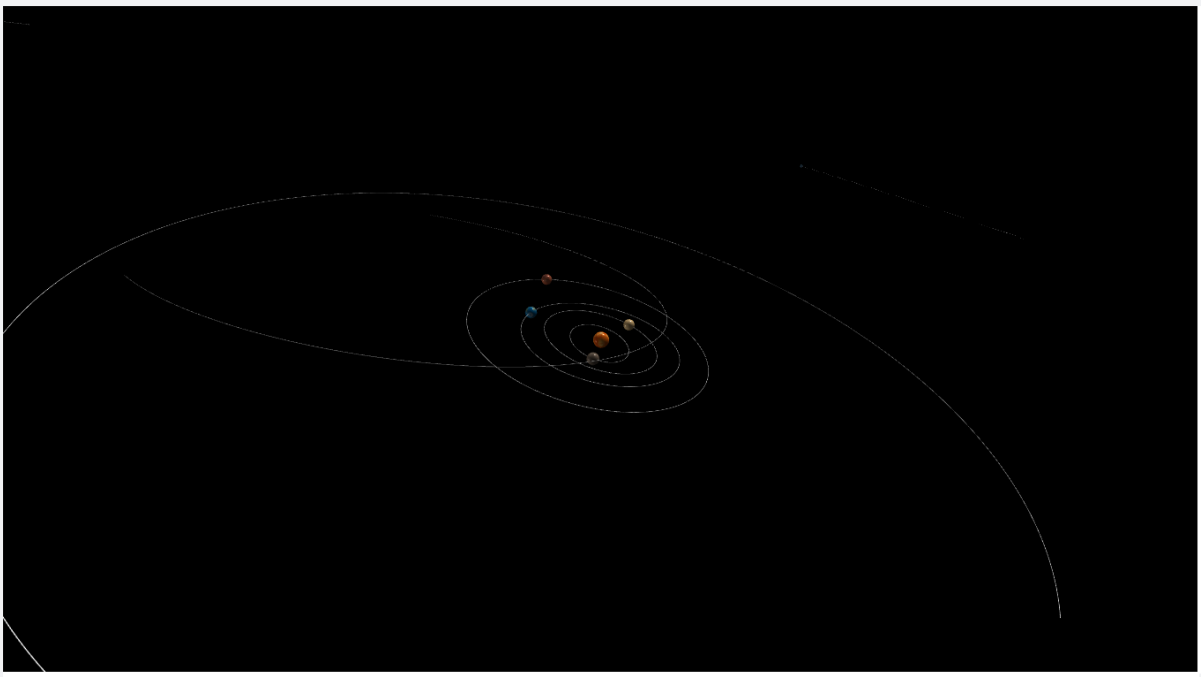
Manual - SistemasPlanetarios.py



Desenvolvido por:

João Pedro Ferreira do Nascimento Lucas	nºUSP: 11802291
João Pedro Santos Lourenço	nºUSP: 13782522
Julia Souto Leoni	nºUSP: 13386270
Marcus Vinicius Novaes Flori	nºUSP: 13836921
Pedro Ruiz Pereira Lopes	nºUSP: 13725611
Raul de Assis Santos	nºUSP: 13748892

Disciplina:
Computação Científica em Python (LOM 3260)

Docente:
Luiz Tadeu Fernandes Eleno

1. Introdução

O programa SistemasPlanetarios.py é um sistema de simulação e análise da posição de astros do espaço sideral com base na passagem do tempo e criado em linguagem de programação python. Pelo código, são inseridos os astros de interesse do público alvo e, caso estejam nos dados da biblioteca ephem.py, a simulação surge imediatamente, caso não estejam, será necessário configurá-los com seus dados de órbita.

2. Metodologia

2.1 Variáveis usadas

As principais variáveis utilizadas no código são aquelas referentes às órbitas dos objetos estelares, tais quais, inclinação, latitude, argumento de periélio, semi-eixo maior, anomalia, excentricidade, período de periélio e velocidade orbital, para os casos de objetos elípticos.

A própria “ephem.py” contém dados de órbita de inúmeros corpos que podemos utilizar, no caso, os mais importantes e próximos da terra. Caso os dados de órbita não estivessem no banco de dados da biblioteca, seria necessário “criar” os objetos no código, como no caso do sistema de satélites artificiais da Terra, em que alguns tiveram de ser criados a partir de fontes de dados externas.

2.2 Texturas

A partir da documentação do VPython, foi possível configurar e aplicar texturas ao código para deixá-lo mais bonito e atrativo aos usuários. Aplicar texturas novas pode ser feito indicando um link ou a partir de imagens. Optou-se pelo segundo método já que o código dependeria menos de condições não controladas pelo usuário.

2.3 Satélites artificiais

Foram escolhidos o satélite brasileiro Amazônia 1, o satélite sinobrasileiro CBERS-4, a estação espacial internacional e o telescópio Hubble. Para a sua configuração

3. O código e seu funcionamento

3.1 Sistema Solar

Para o caso do sistema Solar, recebemos o primeiro código, o qual simulava os planetas do nosso sistema e foi base para a criação dos outros planetas, um ponto importante

é que, para a simulação do sistema solar, foi necessário nos atentarmos para as referências da Terra.

Isso ocorre, pois todos os dados aos quais a biblioteca tem acesso tomam inicialmente como base a posição da terra, dessa forma, num sistema em que ela é um dos objetos orbitantes, tal qual o nosso sistema solar, para atribuir seus dados é na verdade necessário fazer a referência como ao sol.

3.1.1 Criando a Função do cometa Halley

Função das características do cometa Halley. Foi necessário fornecer a inclinação, a longitude, o argumento do periélio, o semi-eixo maior, a anomalia média, a excentricidade, o período do periélio e o seu nome. Os dados usados foram extraídos da Wikipédia.

```python
import ephem

# cometa Halley - dados da Wikipedia
# https://en.wikipedia.org/wiki/Halley%27s_Comet

def create_halley():
    halley = ephem.EllipticalBody()
    halley._inc = 161.96
    halley._Om = 59.396
    halley._om = 112.05
    halley._a = 17.866
    halley._M = 0.07323
    halley._e = 0.96658
    halley_epoch_M = '2061-08-04'
    halley_epoch = '2061-08-04'
    halley.name = 'Halley'
    return halley

```

3.1.2 Pré-código

Aqui foram importadas as bibliotecas usadas e o arquivo com a função dos dados do halley. Visava-se a inclusão de outro cometa também, mas os dados foram insuficientes.

```python
import numpy as np
import datetime as dt
import vpython as vp
import ephem
import cometas
```

3.1.3 Lista de objetos

Cria uma lista com os planetas a serem implementados no código. N representa o tamanho da lista e a data de início é a de agora.

```
# creating the list of objects to be followed, including planets and Halley's comet
objects = [ephem.Mercury(), ephem.Venus(), ephem.Sun(), ephem.Mars(), ephem.Jupiter(), ephem.Saturn(), ephem.Uranus(), ephem.Neptune(),
cometas.create_halley()]

N = len(objects)  # no. of objects
now = dt.datetime.now()
```

3.1.4 Função *locate(obj, date, earth=False)*

A função calcula as coordenadas cartesianas a partir dos dados fornecidos.O sistema de referência do Ephem é a Terra, deste modo, ao configurar o Sol é necessário um cuidado especial: se o objeto for o Sol, o valor de "r" será a distância a Terra, caso contrário, será a do Sol.

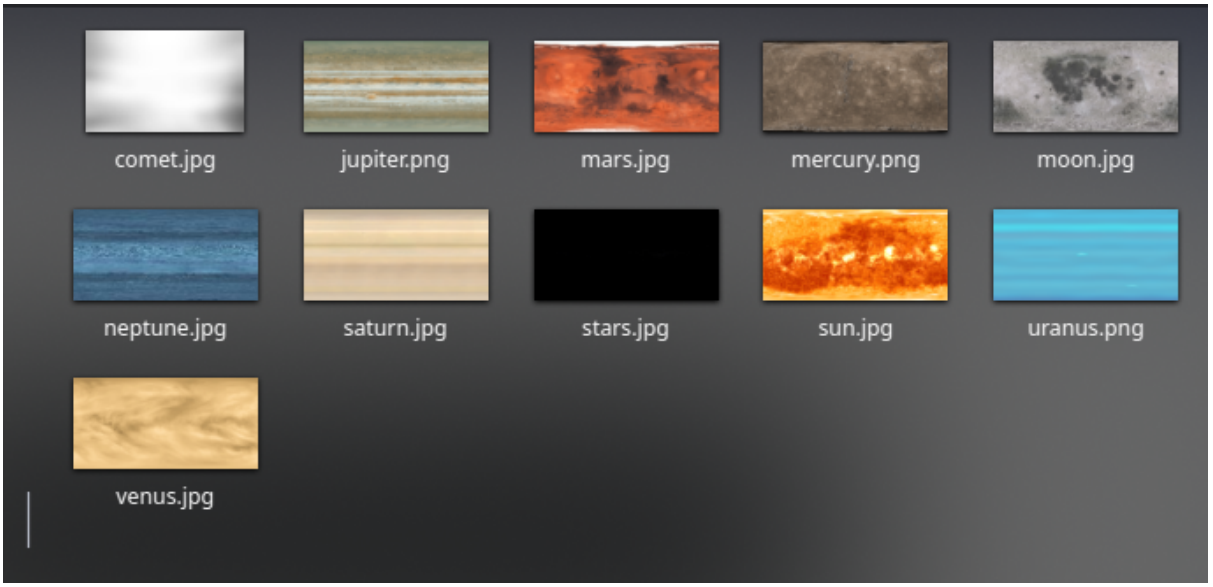
```
def locate(obj, date, earth=False):
    '''
        Determines the position of the object at date
        Converts from ecliptic (spherical) to cartesian coordinates
        The xy plane will be the plane of the ecliptic, with the x axis pointing to the vernal equinox
    '''
    obj.compute(date)  # using ephem .compute() method to get object coordinates
    th, ph = obj.hlat, obj.hlon

    # earth must be treated separatelly
    if obj.name == 'Sun':  # i.e., the Earth
        r = obj.earth_distance
    else:
        r = obj.sun_distance

    # converting to cartesian coordinates
    x = r * np.cos(ph) * np.cos(th)
    y = r * np.sin(ph) * np.cos(th)
    z = r * np.sin(th)
    return x, y, z
```

3.1.5 Textura dos planetas

A textura dos planetas foi baixada em uma pasta. Elas serão integradas ao código por meio da função.



3.1.6 Função *pic_texture()*

Foi criada uma função para aplicar as texturas aos corpos celestes escolhidos. No caso da Terra, por definição, já existia uma textura definida no VPython, já as outras foram as da pasta.

```
# ****************************************
# texture generator
# Returns (with yield) elements from colors. After len(textures) calls, wraps and starts again from the beginning
textures = ['images/mercury.png', 'images/venus.jpg', vp.textures.earth, 'images/mars.jpg', 'images/jupiter.png', 'images/saturn.jpg', 'images/
uranus.png', 'images/neptune.jpg', 'images/comet']
def pick_texture():
    n = 0
    while True:
        yield textures[n]
        n = (n + 1) % len(textures)
texture = pick_texture()
# ****************************************
```

3.1.7 Criando a cena com o Sol ao Centro

A cena inicial tem dimensões 1500x700 o que não será um problema já que ela é ajustável graficamente pelo usuário. O fundo é preto. A configuração do Sol tem coordenadas (0,0,0), seu raio é um pouco maior que os demais e a sua textura está presente na cena.

```
# creating the scene with vpython
vp.scene.width = 1500
vp.scene.height = 700
vp.scene.background = vp.color.black

# sun at the center
sun = vp.sphere(pos=vp.vector(0, 0, 0), radius=0.1, texture='images/sun.jpg')
```

3.1.8 Criando os objetos

São retidos os últimos 10 anos. Isso ocorre pois optamos por não ver a órbita completa de alguns objetos estelares para que o código não fique lento e não armazene

informações desnecessárias. É também por essa razão que os traços de órbita dos planetas mais afastados da Terra não ficam completos, em muitos dos casos, suas órbitas levam mais de 10 anos para completarem o movimento de translação-ao redor do sol.

```
# creating animated objects
balls = []  # position of objects
tracks = []  # trace of the object's orbit
for (p, i) in zip(objects, range(N)):
    x, y, z = locate(p, now)
    pos = vp.vector(x, y, z)
    balls.append(vp.sphere(pos=pos, radius=0.075, texture=next(texture)))
    tr = vp.curve(pos)
    tr.radius = .0025
    tr.retain = 10 * 365  # will retain only the last tr.retain positions
    tracks.append(tr)
```

3.1.9 Animando os objetos

A taxa de atualização é de 200 porque é necessária uma grande escala de tempo para perceber a mudança de determinados astros. A escala da variação de tempo é dada em dias.

```
# animation function
n = 1
while True:
    vp.rate(200)
    date = now + dt.timedelta(days=n)  # increasing date day by day
    for (p, b, t) in zip(objects, balls, tracks):  # relocating objects
        x, y, z = locate(p, date)
        b.pos = vp.vector(x, y, z)
        t.append(b.pos)
    n += 1
```

3.2 Terra, Lua e satélites artificiais

Para a Lua, a biblioteca Ephem continha os dados de efemérides, podendo tratá-la como foi feito para os planetas do sistema solar, com tratamento heliocêntrico, utilizando hlat, hlong e earthdistance.

Para os satélites artificiais, criados no programa por meio de TLE, foi preciso de um tratamento diferente dos dados, utilizando as coordenadas angulares sublat, sublong e elevation.

3.2.1 Configuração dos dados dos satélites artificiais

Para criação dos satélites como objetos lidos pelo “ephem”, foram obtidos os dados TLE (Two Line Element Set) de cada satélite (com as fontes devidamente citadas no código)

que foram lidos pela função readtle() do ephem, criando os objetos, do tipo "ephem.EarthSatellite".

```
import ephem

# fonte TLE ISS / https://www.n2yo.com/satellite/?s=25544
linha1 = 'ISS'
linha2 = '1 25544U 98067A   22349.56974730  .00009336  00000-0  17159-3 0  9998'
linha3 = '2 25544  51.6425 158.0459 0003999 163.5159 307.5458 15.49959742373311'
iss = ephem.readtle(linha1, linha2, linha3)

# fonte TLE Amazonia 1 / https://www.n2yo.com/satellite/?s=47699#results
linha1 = 'Amazonia1'
linha2 = '1 47699U 21015A   22348.68610750 -.00000044  00000-0  00000-0 0  9995'
linha3 = '2 47699  98.4470  63.4053 0001488 117.9208 242.2132 14.40819803 94253'
amazonia1 = ephem.readtle(linha1, linha2, linha3)

# fonte TLE CBERS-4 / https://www.n2yo.com/satellite/?s=40336
linha1 = 'CBERS-4'
linha2 = '1 40336U 14079A   22348.53045017  .00000165  00000-0  72424-4 0  9996'
linha3 = '2 40336  98.4670  59.2014 0000734  77.3546 282.7730 14.35456752420242'
cbers4 = ephem.readtle(linha1, linha2, linha3)

# fonte TLE Hubble / https://www.n2yo.com/satellite/?s=20580
linha1 = 'HST'
linha2 = '1 20580U 90037B   22349.27408213  .00003512  00000-0  19203-3 0  9990'
linha3 = '2 20580  28.4712  49.9906 0002546 181.0832 221.6335 15.11408957593851'
hst = ephem.readtle(linha1, linha2, linha3)
```

3.2.2 Pré código

Foram importadas as bibliotecas usadas e o arquivo com as informações dos satélites artificiais desejados.

```
import numpy as np
import datetime as dt
import vpython as vp
import ephem
import satelites
```

3.2.3 Terra ao centro e a criação dos objetos

A Terra é configurada ao centro. O seu raio foi convertido de km para unidades astronômicas. Foi aplicada a textura já presente no VPython. Após isso, foi criada uma lista com a Lua, o satélite Amazônia 1, a ISS, o satélite CBERS-4 e o Hubble.

```
# Terra no centro
earth = vp.sphere(pos=vp.vector(0, 0, 0), radius= ephem.earth_radius/1.496e+11, texture= vp.textures.earth)

# creating the list of objects to be followed

objects = [ephem.Moon(), satelites.amazonia1, satelites.iss, satelites.cbers4, satelites.hst]

N = len(objects)
now = dt.datetime.now()
```

3.2.4 Função *locate(obj, date)*

A função locate(obj, date) retorna a posição x, y, z em determinada data de determinado objeto. Para a Lua, foi possível utilizar a mesma abordagem utilizada para os planetas do sistema solar, com parâmetros hlat, hlon e earth_distance. Para os satélites artificiais, foi necessário usar os parâmetros ".sublat", ".sublong" e ".elevation", respectivamente latitude, longitude (posição do satélite em termos de latitude e longitude geográficas da superfície da Terra) e elevação do satélite em relação a superfície do planeta. Somando a elevação do satélite com o raio da Terra, foi obtido a distância do satélite em relação a origem da cena do VPython.

```
'''
    Determina posição do satélite em relação a data dada
    Toma a posição da projeção do satélite na Terra em função de longitude e latitude na superfície do planeta
    A distância do satélite até o ponto (0,0,0) do espaço tridimensional do vpython é dado pelo raio da Terra
    somado a elevação do satélite em relação a superfície do planeta, também em UA.
'''

def locate(obj, date):
    obj.compute(date)  # using ephem.compute() method to get object coordinates
    if obj.name != 'Moon':
        th, ph = obj.sublat, obj.sublong
        r = (obj.elevation + ephem.earth_radius)/1.496e+11
    else:
        th, ph = obj.hlat, obj.hlon
        r = obj.earth_distance
    # converting to cartesian coordinates
    x = r * np.cos(ph) * np.cos(th)
    y = r * np.sin(ph) * np.cos(th)
    z = r * np.sin(th)
    return x, y, z
```

3.2.5 Cores e texturas

Foi aplicada textura à Lua. O satélite Amazônia 1, a ISS, o satélite CBERS-4 e o Hubble estão nas cores verde, branco, vermelho e azul - respectivamente.

A função utilizada para aplicar as texturas foi a mesma utilizada previamente.

```
# texture generator
# Returns (with yield) elements from textures. After len(textures) calls, wraps and starts again from the beginning
textures = ['images/moon.jpg', vp.color.green, vp.color.white, vp.color.red, vp.color.blue]
def pick_texture():
    n = 0
    while True:
        yield textures[n]
        n = (n + 1) % len(textures)
texture = pick_texture()
```

3.2.6 Criando a cena

Desta vez a resolução é de 1500x700. O fundo se mantém preto.

```
# creating the scene with vpython
vp.scene.width = 1500
vp.scene.height = 700
vp.scene.background = vp.color.black
```

3.2.7 Criando os objetos

São criados os objetos. A função vai conferir qual é o satélite: caso seja a Lua, o seu tamanho será de $\frac{1}{2}$UA, caso contrário, o seu raio será de $\frac{1}{10}$UA. Da Lua serão salvas as últimas 500 posições. Já dos satélites artificiais serão salvas as últimas 60.

```
# creating animated objects
balls = []  # position of objects
tracks = []  # trace of the object's orbit
for (p, i) in zip(objects, range(N)):
    x, y, z = locate(p, now)
    pos = vp.vector(x, y, z)
    print(p.name)
    if p.name == 'Moon':
        balls.append(vp.sphere(pos=pos, radius=(ephem.earth_radius / 1.496e+11)/2, texture=next(texture)))
    else:
        balls.append(vp.sphere(pos=pos, radius=(ephem.earth_radius / 1.496e+11)/10, color=next(texture)))
    tr = vp.curve(pos)
    tr.radius = (ephem.earth_radius/1.496e+11)/100
    if p.name == 'Moon':
        tr.retain = 500  # will retain only the last tr.retain positions
    else:
        tr.retain = 60
    tracks.append(tr)
```

3.2.8 Animação

A taxa de atualização é de 100. A escala da variação de tempo é dada em minutos.

```
# animation function
n = 0
while True:
    vp.rate(100)
    date = now + dt.timedelta(minutes=n)  # increasing date minute by minute
    for (p, b, t) in zip(objects, balls, tracks):  # relocating objects
        x, y, z = locate(p, date)
        b.pos = vp.vector(x, y, z)
        t.append(b.pos)
    n += 1
```

3.3 Júpiter e as 4 Galileanas

No caso do sistema planetário de Júpiter, muitas de suas luas não se encontram no banco de dados da biblioteca ‘ephem’ e, também pela quantidade de luas, optamos por não simular todas. Portanto, escolhemos a simulação das 4 mais famosas luas, aquelas que foram observadas por Galileu Galilei nos idos dos anos de 1700, eram elas: Io, Europa, Ganimedes e Calisto.

No código do sistema de Júpiter elas são representadas pelas cores verde, branco, vermelho, azul e ciano.

3.3.1 Pré código

Foram importadas as bibliotecas datetime, VPython e ephem.

```
import datetime as dt
import vpython as vp
import ephem
```

3.3.2 Júpiter ao centro e lista de objetos

Júpiter é colocado no centro e seu raio em km é convertido para UA. Foi aplicada a textura do planeta. Após isso, foram criadas as quatro Luas: Io, Ganymede, Callisto e Europa.

```
jraio = 69911000/1.496e+11 # raio de jupiter em unidades astronomicas

Jupiter = vp.sphere(pos=vp.vector(0, 0, 0), radius= jraio, texture= 'images/jupiter.png')
objects = [ephem.Io(), ephem.Ganymede(), ephem.Callisto(), ephem.Europa()]

N = len(objects)  # no. of objects
now = dt.datetime.now()
```

3.3.3 Objetos e cores

As cores de Io, Ganymede, Callisto e Europa foram verde, branco, vermelho e azul.

```
def locate(obj, date):
    '''
        returns the x, y, z coordinates relative to Jupiter in terms of planet radius
    '''
    obj.compute(date)  # using ephem .compute() method to get object coordinates
    x, y, z = obj.x*jraio, obj.y*jraio, obj.z*jraio
    return x, y, z

# ****************************************
# texture generator
# Returns (with yield) elements from colors. After len(textures) calls, wraps and starts again from the beginning
colors = [vp.color.green, vp.color.white, vp.color.red, vp.color.blue, vp.color.cyan]
def pick_color():
    n = 0
    while True:
        yield colors[n]
        n = (n + 1) % len(colors)
texture = pick_color()
# ****************************************
```

3.3.4 Criando a cena

A cena possui dimensões 500x500 e o fundo é preto.

```
#criando a cena em vpython
vp.scene.width = vp.scene.height = 500
vp.scene.background = vp.color.black
```

3.3.5 Criando os objetos animados

De maneira análoga às outras simulações, foi definido um raio arbitrário para as esferas que representam as luas do planeta, definido como ⅕ raio de Júpiter, para melhor visualização na simulação.

```
#creating animated objects
balls = []  # position of objects
tracks = []  # trace of the object's orbit
for (p, i) in zip(objects, range(N)):
    x, y, z = locate(p, now)
    pos = vp.vector(x, y, z)
    balls.append(vp.sphere(pos=pos, radius=jraio/5, color=next(texture)))
    tr = vp.curve(pos)
    tr.radius = jraio/30
    tr.retain = 40  # will retain only the last tr.retain positions
    tracks.append(tr)
```

3.3.6 Animação

A taxa de atualização é de 100. A taxa de variação de tempo é dada em horas.

```
# animation function
n = 1
while True:
    vp.rate(100)
    date = now + dt.timedelta(hours=n)  # increasing date hour by hour
    for (p, b, t) in zip(objects, balls, tracks):  # relocating objects
        x, y, z = locate(p, date)
        b.pos = vp.vector(x, y, z)
        t.append(b.pos)
    n += 1
```

4. Bibliotecas registradas

As bibliotecas registradas utilizadas no código são: VPython, Ephem, Numpy e Datetime.

```
import numpy as np
import datetime as dt
import vpython as vp
import ephem
import cometas
```

5. Referências

https://github.com/luizeleno/LOM3260

https://pt.wikipedia.org/wiki/Amazonia_1

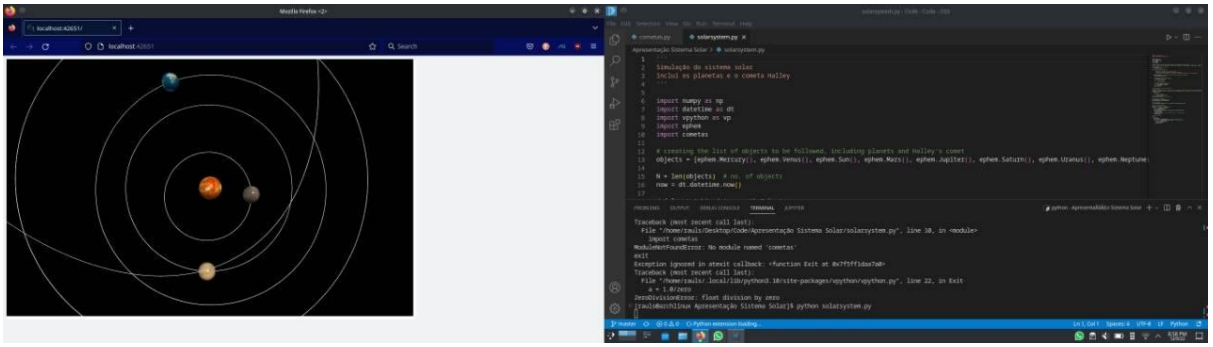
https://pt.wikipedia.org/wiki/CBERS-4

https://rhodesmill.org/pyephem/quick.html

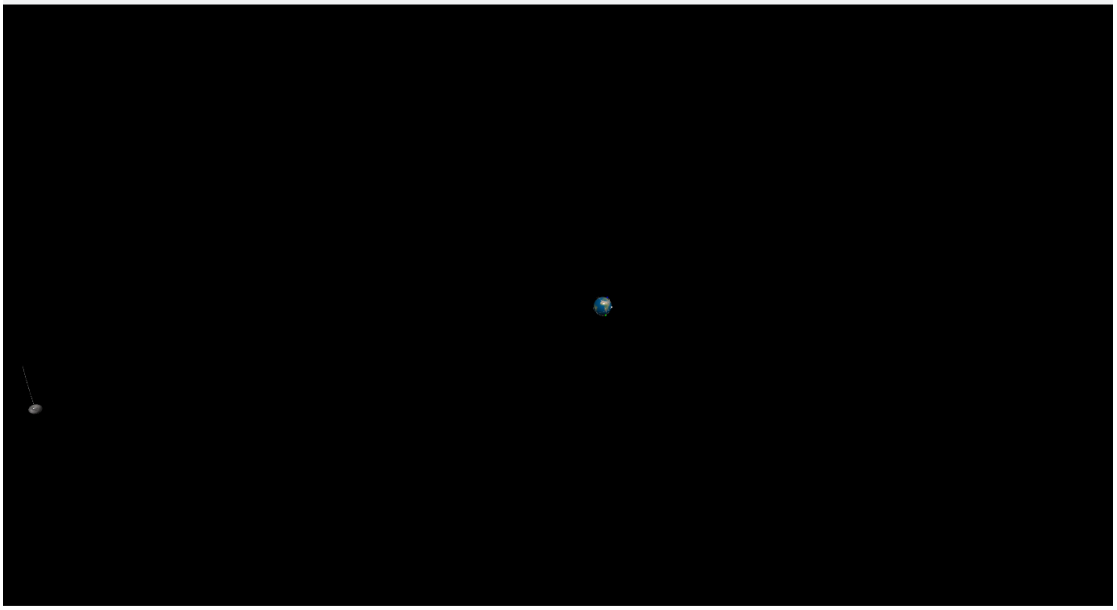
https://www.glowscript.org/docs/VPythonDocs/textures.html

APÊNDICE A - Visualização do código funcional

Visualização do Sistema Solar e o cometa Halley ao lado do código



Visualização da Terra, a Lua e os satélites artificiais



Visualização de Júpiter e suas luas