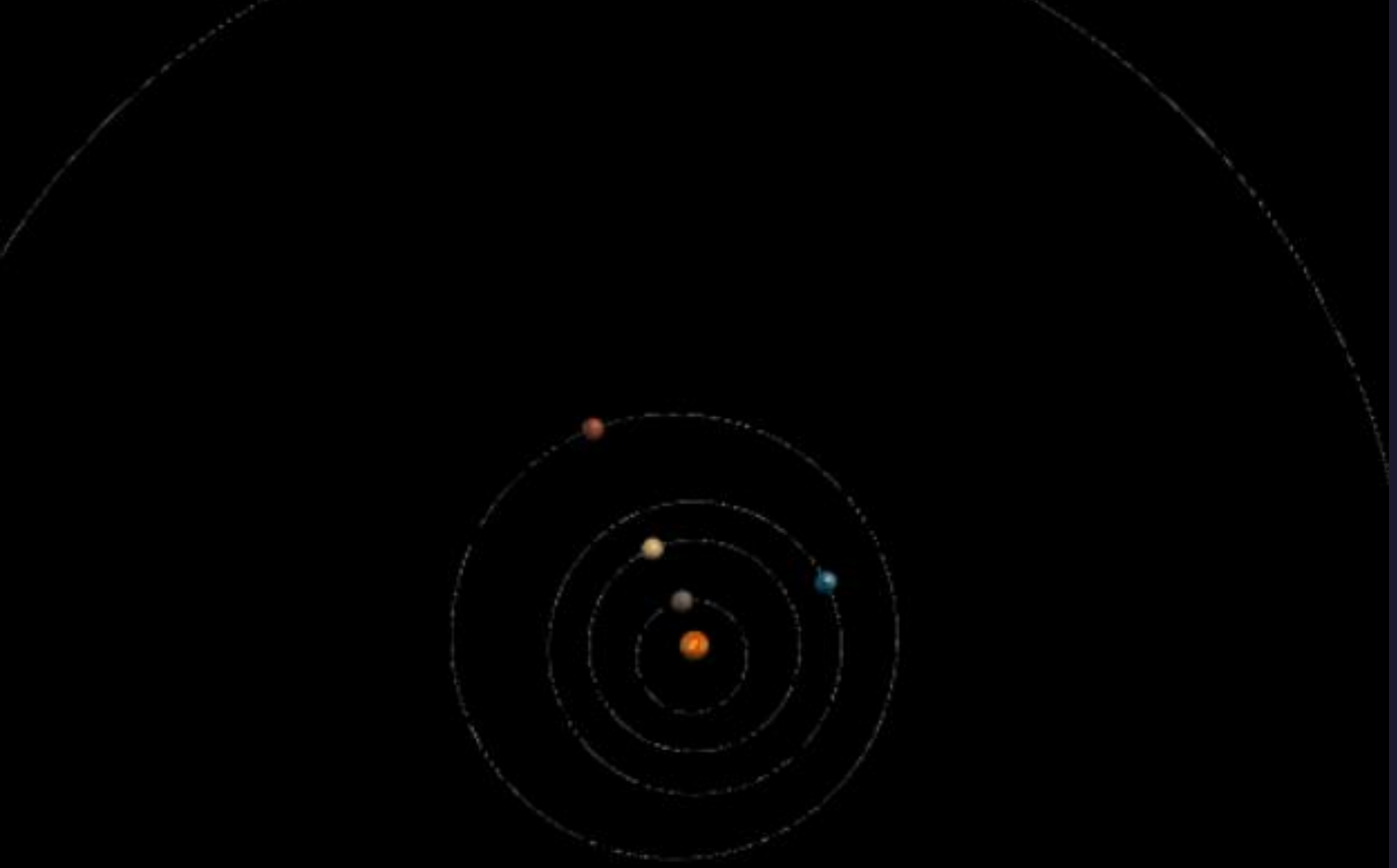
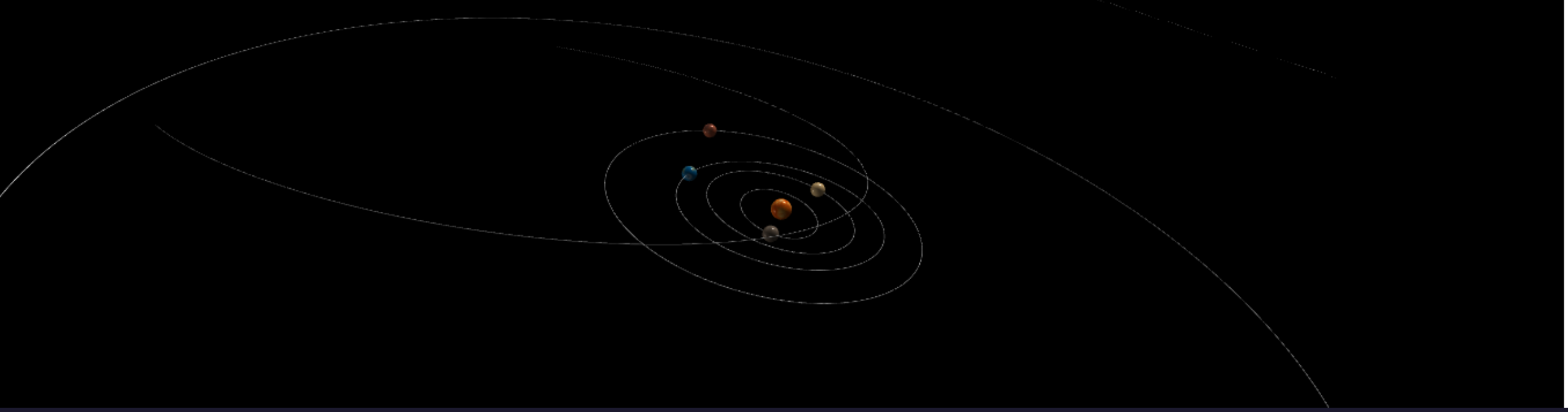


Simulação de Sistemas Planetários



Estrutura do Sistema

O programa SistemasPlanetarios.py é um sistema que simula a posição dos astros, dentro de uma certa proporção, em função do tempo, sendo todo escrito em linguagem de programação Python. Consiste essencialmente nos plantes do Sistema Solar e no Cometa Halley.

Atividade do código

O código transforma as coordenadas espaciais dos astros em coordenadas cartesianas a partir de uma função e também ordena os planetas e satélites através de uma lista.

Com isso, o programa simula a órbita dos planetas no sistema tendo como referência a hora e data atual, a biblioteca ephem,py utiliza um banco de dados que passa por muitas análises para informar a posição real dos astros(com poucos desvios), é uma biblioteca que contém a maioria dos dados astronômicos individuais dos corpos do sistema.

Informações de órbita

```
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
```

As variáveis utilizadas foram basicamente, inclinação, latitude, argumento de periélio, semi-eixo maior, anomalia, excentricidade, período de periélio e velocidade orbital.
Sem esses dados não é possível realizar a simulação da órbita de corpos celestes que não estão presentes na biblioteca, eles servem para adicionar novos objetos no sistema.
Para acrescentar objetos que não estão no banco de dados da biblioteca, é necessário obter algumas dessas respectivas variáveis.

Comando import das bibliotecas

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

A partir deste códigos de import acessamos todas as informações disponibilizadas na biblioteca que descrevem o movimento elíptico de cada planeta no programa.

Conversão para as coordenadas cartesianas

```python
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
```

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, pois, no caso, sua referencia seria a própria Terra.

Basicamente, temos uma coordenada angular e convertemos ela para posições x, y, z como mostrado no código.

Adicionando textura aos planetas

A textura dos planetas foi baixada em uma pasta. Foi criada uma função, pic_texture(), 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 acessadas pela pasta.

Definindo o objeto central

```
# 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')
```

A cena inicial tem dimensões ajutadas graficamente pelo usuário. O fundo é preto.
A configuração, em questão, do Sol, tem coordenadas (0,0,0) de sua posição, seu raio é um pouco maior que os demais e a sua textura está presente na cena.

Configurações dos dados dos satélites artificiais

```
import ephem

linha1 = 'ISS'
linha2 = '1 25544U 98067A   22348.81396991  .00006034  00000-0  11385-3 0  9999'
linha3 = '2 25544  51.6409 161.7936 0004026 159.8255  51.3354 15.49939714373193'
iss = ephem.readtle(linha1, linha2, linha3)

linha1 = 'Amazonia1'
linha2 = '1 47699U 21015A   22348.47777215 -.00000044  00000-0  00000+0 0  9991'
linha3 = '2 47699  98.4471  63.1997 0001495 118.7340 241.4003 14.40819795 94227'
amazonia1 = ephem.readtle(linha1, linha2, linha3)

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)

linha1 = 'HST'
linha2 = '1 20580U 90037B   22348.56474272  .00003257  00000-0  17736-3 0  9994'
linha3 = '2 20580  28.4712  54.6968 0002586 173.9231 321.5688 15.11403489593484'
hst = ephem.readtle(linha1, linha2, linha3)
```

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.

Animando os objetos

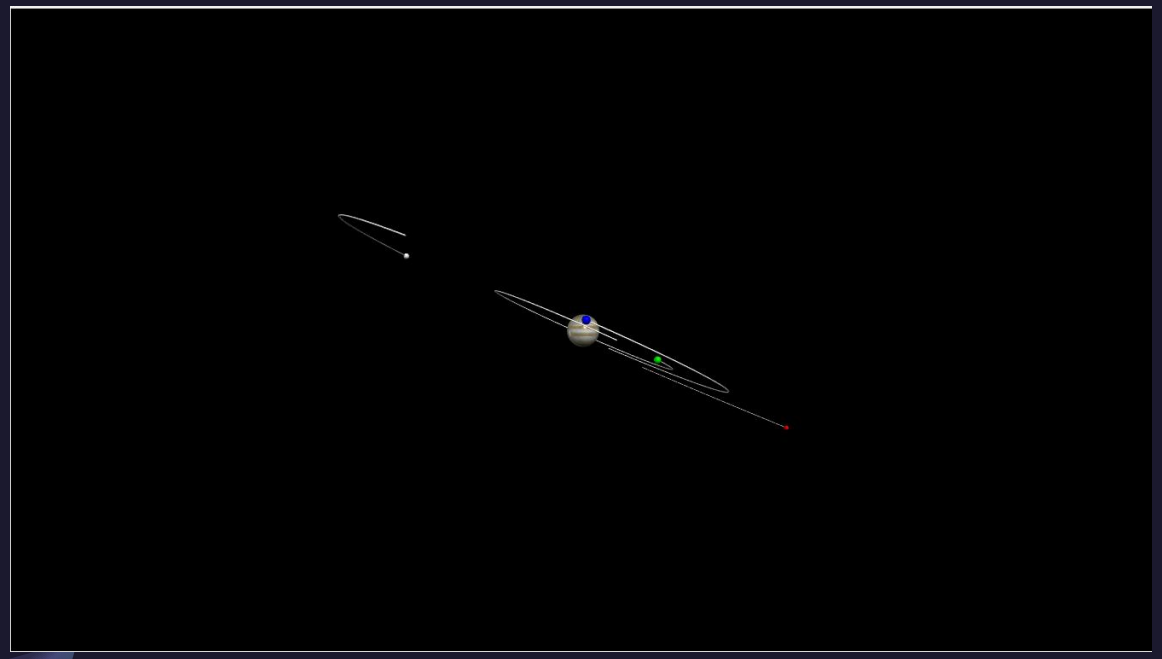
```python
# animation function
n = 0
while True:
    vp.rate(100)
    date = now + dt.timedelta(minutes=n)  # increasing date minute by minute
    for (p, b, t) in zip(objects, balls, tracks):  # relocating objects
        x, y, z = locate(p, date)
        b.pos = vp.vector(x, y, z)
        t.append(b.pos)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

Para animar o código inicialmente utilizaríamos a biblioteca matplot.lib, depois de alguns testes, optamos pela utilização apenas da biblioteca Vpython que já satisfazia nossas necessidades.

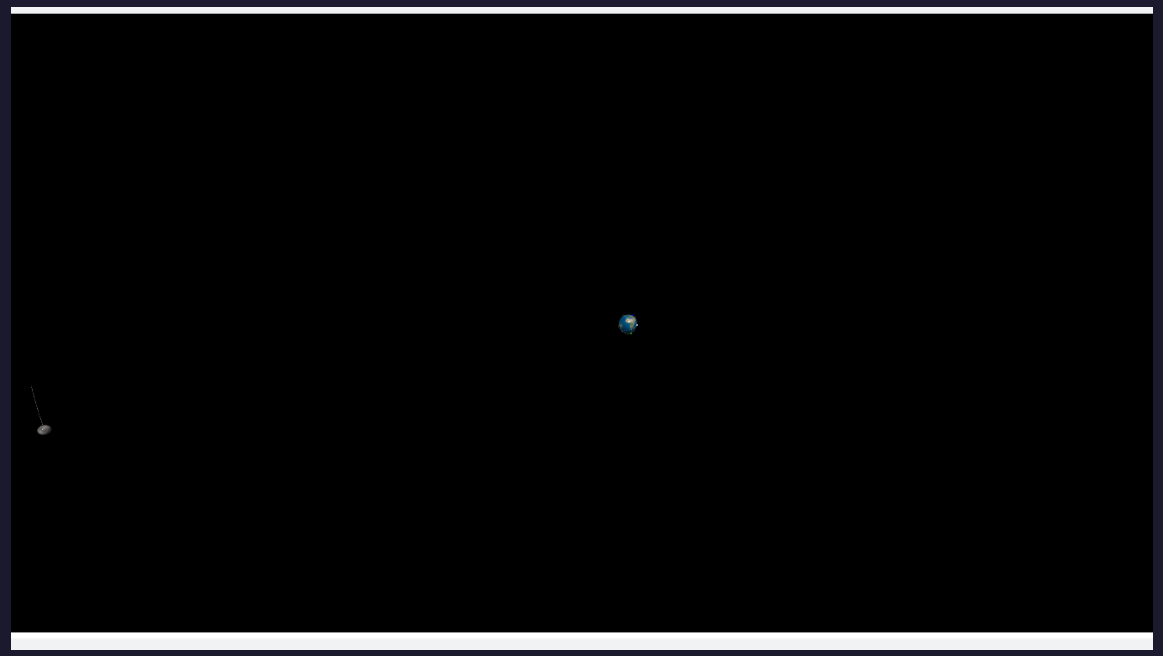
```python
#creating animated objects
balls = []  # position of objects
tracks = []  # trace of the object's orbit
for (p, i) in zip(objects, range(N)):
    x, y, z = locate(p, now)
    pos = vp.vector(x, y, z)
    balls.append(vp.sphere(pos=pos, radius=jraio/5, color=next(texture)))
    tr = vp.curve(pos)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

Resultados finais

Sistemas de Jupiter

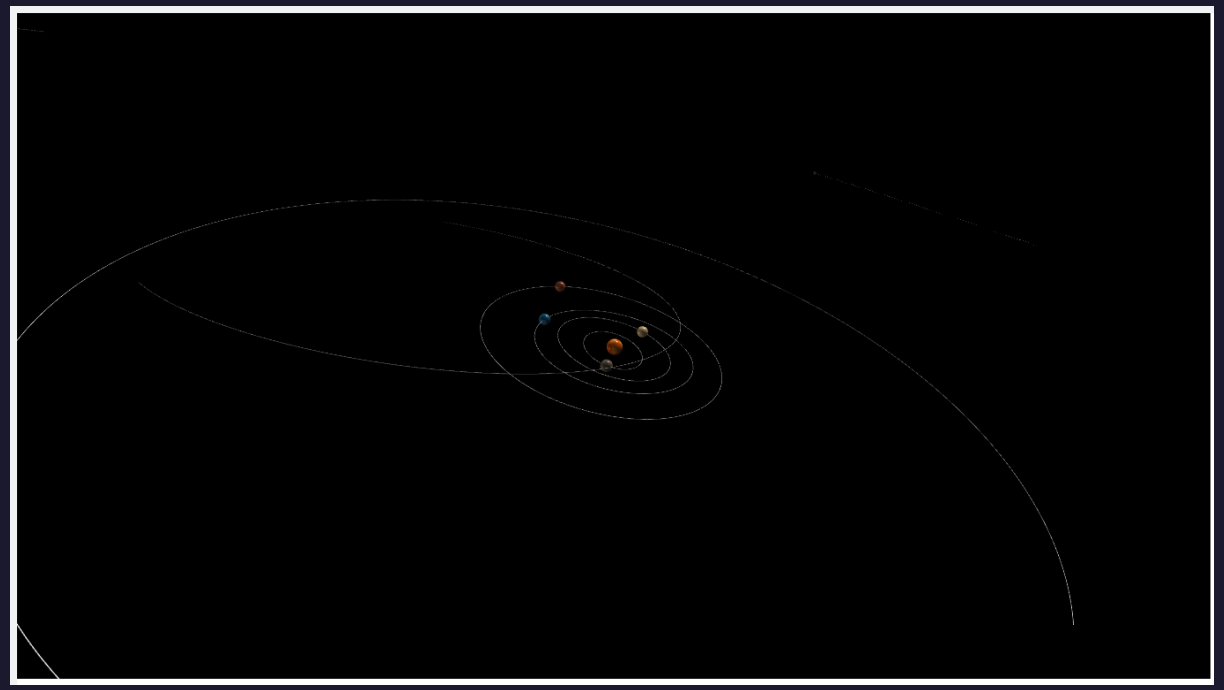


Sistema Terra-Lua e Satélites artificiais



Resultados finais

Simulação Sistema Solar



Obrigado pela atenção!!!