Where is the space station?

Cesar Cisneros; Jeff Klenzing

Abstract

This program is an interactive map that allows any user to locate the orbit of the space station at a given city and location. The user has the ability to choose from a drop-down list the date, that is, a set month, day and set year, and the city. Both the date and the city appear as drop-down tables, creating a variety of combinations to pick from. These two variables (time and city) are collected to show the closest position of the space station through an orbit on the map, plotting the orbit and city, marked as a star. The orbits appear as waves, some only depicting portions of the orbit due to certain gaps in the data.

Introduction

My GUI is based on a frame widget that allowed me to incorporate both the map and the functions of the program within this frame. The dates were picked at random along with the cities, covering at least one city per continent. The city and date will appear as buttons, therein containing a list of their respective values. depicting a particular date as default along with the city of the user's choice. The cities were entered as geographic coordinates and plotted with a star in order to be easily identified.

```
def load_new_data(date):
    '''
    get new date from dropdown box
    load new data using pysat for selected date
    '''

    year = int(date[6:])
    month = int(date[0:2])
    day = int(date[3:5])

    pdate = pysat.datetime(year,month,day)
#    start = ps.datetime(2017,1,01)
#    stop = ps.datetime(2017,1,14)
    iss.load(date=pdate)

    print(date)
```

Above, the function *load_new_data* was used to load the data using pysat and converted into indices to meet the desired date format in order to be read and plotted accordingly later.

Methods

The program focused on a particular set of dates to perform its function. Given the size of the archive, it seemed only logical to include the data of two weeks to perform the desired functionality. I used Pysat (Python Satellite Data Analysis Toolkit) as a means of accessing the historical data for the ISS object. I used Basemap to create the background map, which depicts landforms and main bodies of water.

In order to calculate the closest distance, that is the closest longitude and latitude, from the satellite object, I used the great-circle distance. This distance allowed me to find the shortest distance between the latitude and longitude of a sphere.

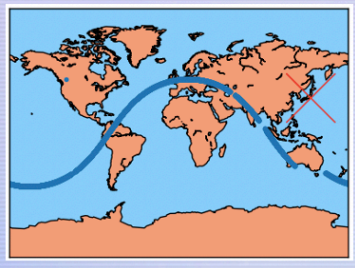
$$\Delta\sigma = \arccos\left(\sin\phi_1 \cdot \sin\phi_2 + \cos\phi_1 \cdot \cos\phi_2 \cdot \cos(\Delta\lambda)\right).$$

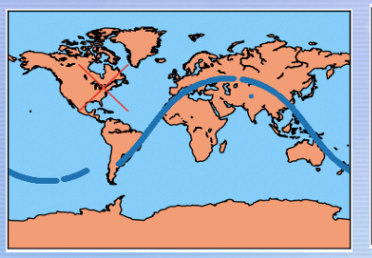
Essentially, this is the distance the great-circle distance measures when finding the distance between two different points.

Above, I found the sine value of the Latitude of the given city and converted the output values to radians. In addition, I also found the cosine of Longitude as well as the cosine of the Latitude. All these values were converted into radians. For the values obtained through this distance, I used a function (idxmin) to determine the minimum distance and return this value, which would later determine the physical position of the orbit on the map. Minimize the function in order to find the closest approach to a given city).

Results

To the upper right, the GUI is plotting the orbit of the city of Tokyo, Japan from 01/09/2017. On the bottom left, the GUI is plotting the orbit of the city of Toronto, Canada from 01/05/2017. On to the bottom right, the GUI is plotting the orbit of the city of Sydney, Australia from 01/06/2017. All the orbits appear blue given they were taken during different time frames; however, if they were to be plotted within the same time lapse, the GUI would plot each orbit with a different color to differentiate between them.





Discussion

Despite the program's basic functions in identifying orbits, the program can be further developed into a far more complex algorithm to meet different goals. For instance, one can incorporate every city in the world in order to identify trends between one or more orbits in contrast to other cities.

Furthermore, adding additional interfaces, that is more conditions for the program to be identified beyond just a city and date, could result in more accurate predictions about future orbital locations. One example could be the addition of multiple orbits at once for one given location and date or, rather than plotting past orbits, plot future orbits for a given date and compare those

Contact

[Cesar Cisneros]
[Heliophysics]
[301-395-5154]

References

1. https://cdn.britannica.com/06/104206-050-96A27B10.gif
2. https://www.movable-type.co.uk/scripts/latlong.html