AEC 2022
Team Shediac

Greg Kean
Evan Fitzgerald
Justen Di Ruscio
Paul Walsh

The Problem

- Our team has been tasked with creating a software tool that calculates the distance and transmission latency between two planets orbiting the Sun.

- The software must also warn users if a specified point in time coincides with a solar conjunction, and notify them of the date of the next planetary opposition

Constraints

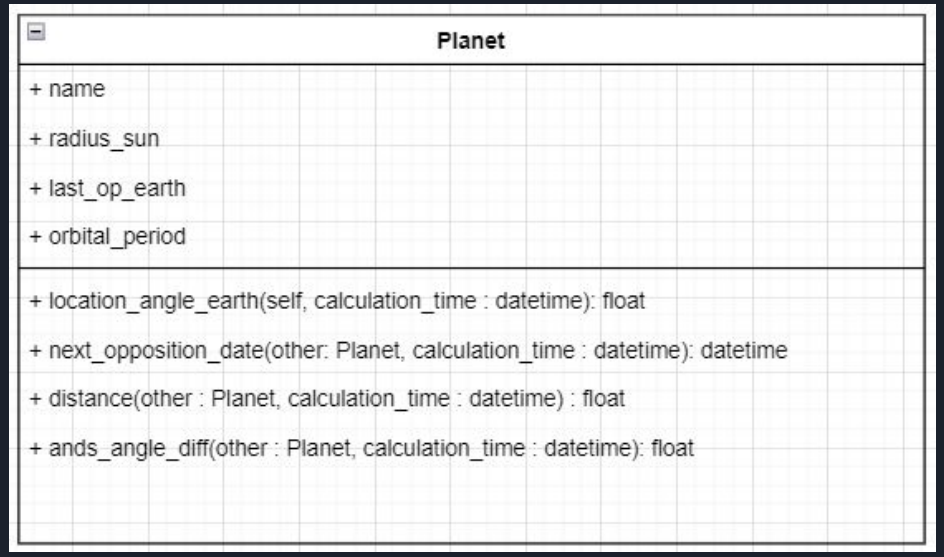
- **Input -** Any number of planets with:
  - Planet Name
  - Orbital Period
  - Orbital Radius
  - Date/time of Last Opposition with Earth

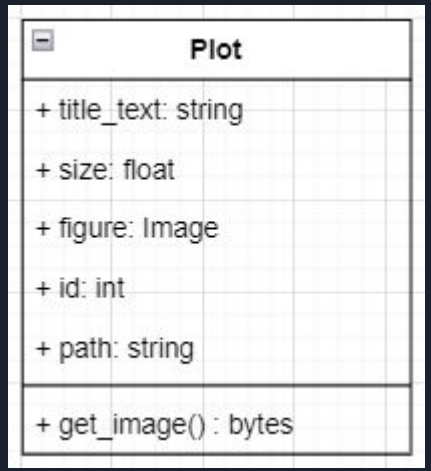
- **Output**
  - Distance separating the specified planets
  - Latency associated with transmitting a signal between the planets
  - Whether the specified date falls within the moratorium on transmitting commands due to solar conjunction
  - Date of the next planetary opposition
  - A graph illustrating daily signal latency over the next 24 months from the date specified
  - A visualization illustrating the positions of both planets in their orbits on the date specified

Approach

- Reduce complexity through smart design
- Work in parallel as much as feasible
- Pomodoro method
- Use tools for fast, simple application development
  - Python3
  - GitHub
- Use tried and tested frameworks and libraries
  - PySimpleGUI
  - NumPy
  - pandas
  - Plotly

Detailed Solution





Detailed Solution Cont’d

Utility Functions:

Driver:

- main()

GUI:

- planet_positions ():
- create_min_layout(): list
- create_result _layout(image, dist_between, morat, next+opp, date): list

Calculations:

- is_moratorium (planet_angle_diff: float) : boolean
- kin_duration(displacment : float, velocity :float) : float
- angular _velocity (period: timedelta): float
- Communication_latency (planet1: Planet, planet2: Planet, calculation_time: datetime)

Deliverables

Visualization:

- Pysimplegui console application

Backend:

- Python Backend
- Object Oriented Software
- Simple documentation and type hints

Strength Analysis

- Portable
- Scalable
- Separation of concerns
- DRY (don’t repeat yourself) principle
- Abstract implementation for future enhancements and maintenance
- Adherence to Proper UX Design
  - Consistent color tones
  - Clarity was priority
  - One primary function

7 Principles of ECL

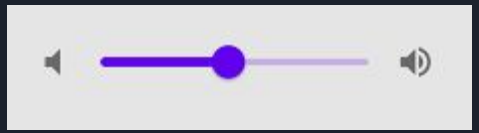
1. Seek Purpose
   a. Creating a solution to solve a problem
2. Take Responsibility
   a. Each member had guidelines of what they were expected to do and expected to manage time properly
3. Expand Involvement
   a. Team members were each given tasks, but collaborated with other members when able
4. Widen Approaches
   a. Many algorithms were considered before coding began
5. Advance Understanding
   b. All team members worked on technologies that were new to them
6. Realize Diversity
   c. Previous experiences and strengths were considered when assigning tasks
7. Deliberate Values
   d. Team discussion held on whether having a fast or robust solution was more important

Opportunities

- Distance and latency between two planets follow a periodic pattern. Computation could thus be simplified through finding and using the associated equation.

Enhancements Beyond Given Problem

- Slider UI element, allowing user to quickly view calculations over a range of times
- Extend calculation to any number of planets



Summary

- Clear, Simple PySimpleGUI front-end
  - Fulfilling many UX principles
- Powerful and Modular Python Back-end
  - With respect for Software Engineering Principles
- Solution designed to avoid complexities introduced by constraints
- Displayed bonus using Informative animation.

Questions?