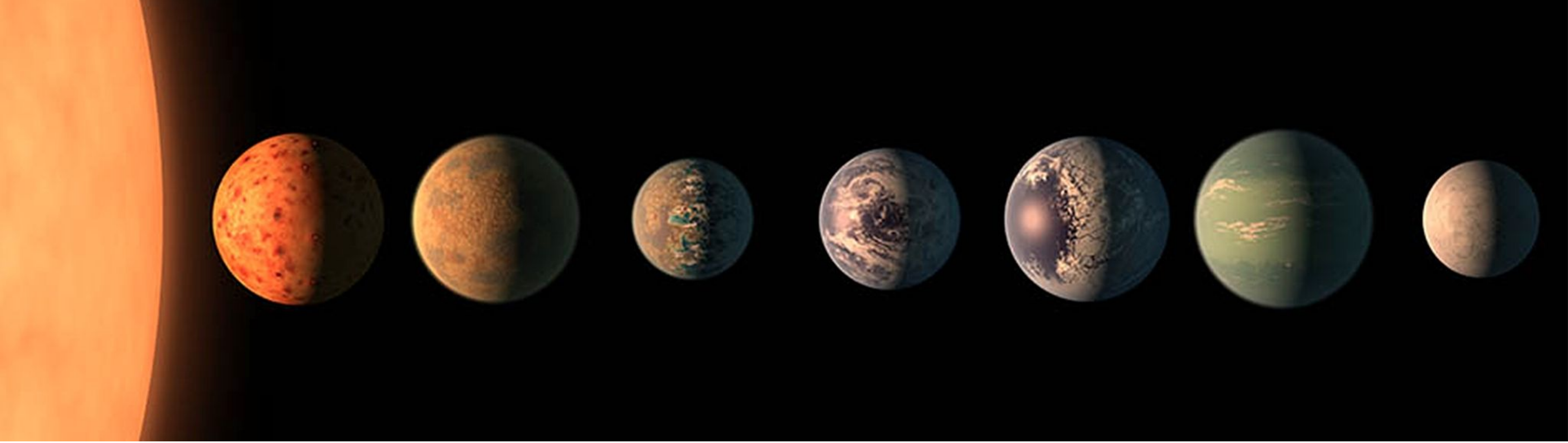
Simulating Solar Systems

Claire Chen, Josh Bonivert, Eve Mango, Sofia Daubert | Python DeCal



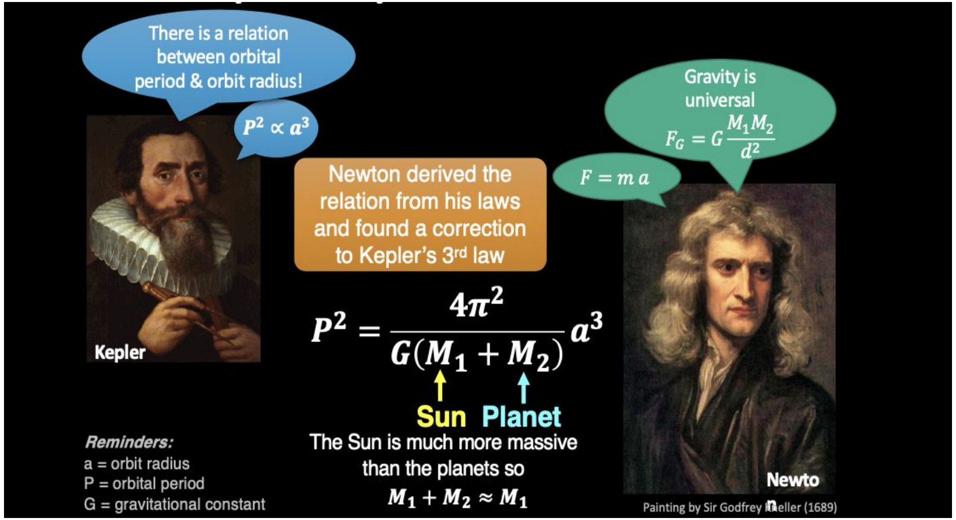
1.
Planets & orbits

- **Kepler's 3 Laws describe planetary motion**

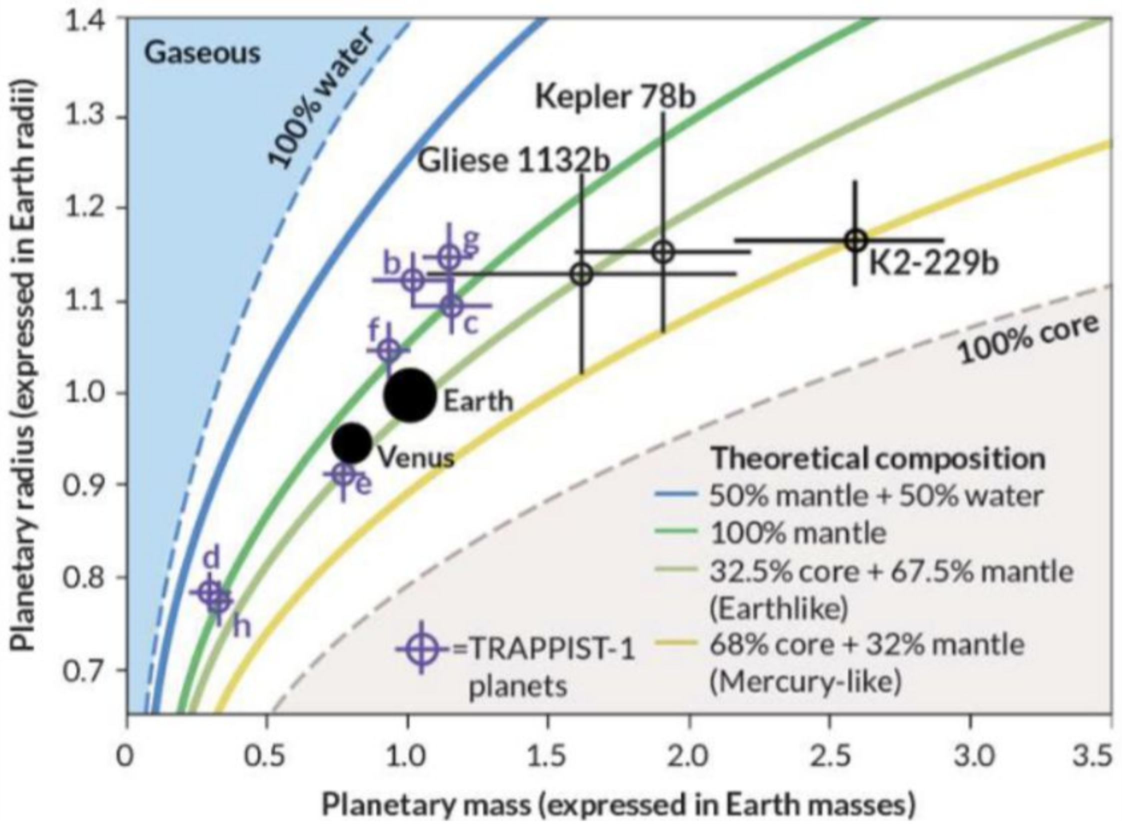
  a) Planet orbits in an ellipse w/ Sun at 1 focus
  b) Line connecting planet to Sun sweeps out equal areas in equal time intervals
  c) Square of orbital period is proportional to cube of planet's semimajor axis

Newton's Laws Explain Kepler's Observations

$$F = G\frac{m_1 m_2}{r^2}$$

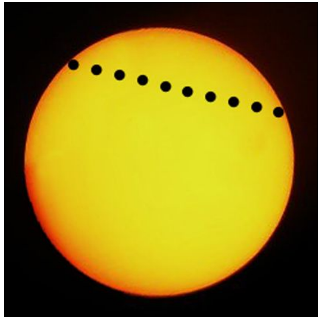


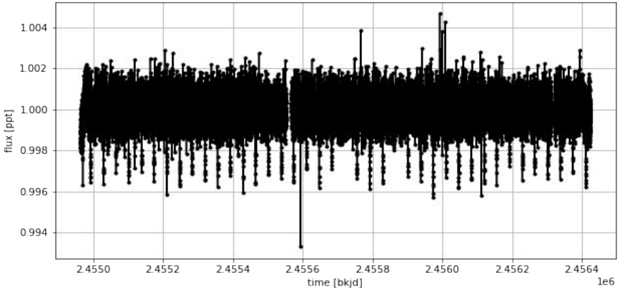
Different Types of Planets

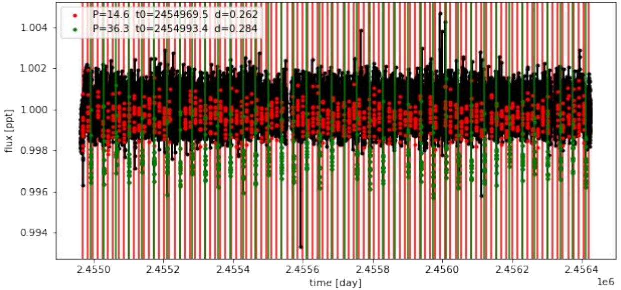


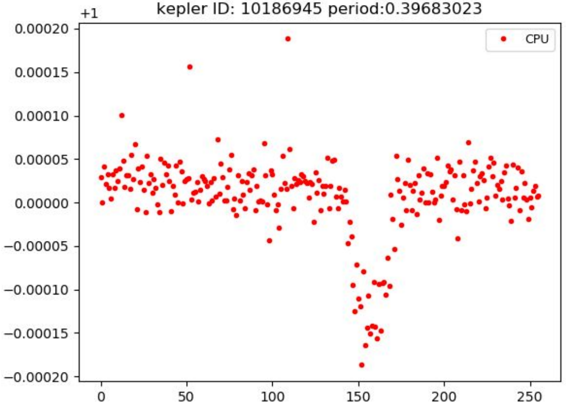
Transits

- Lightcurve: apparent magnitude (observed brightness) of star over time
- Periodic dips in observed brightness can be from transiting objects
- Detecting exoplanets through transit photometry
- Transit depth = planet:star radius ratio, squared



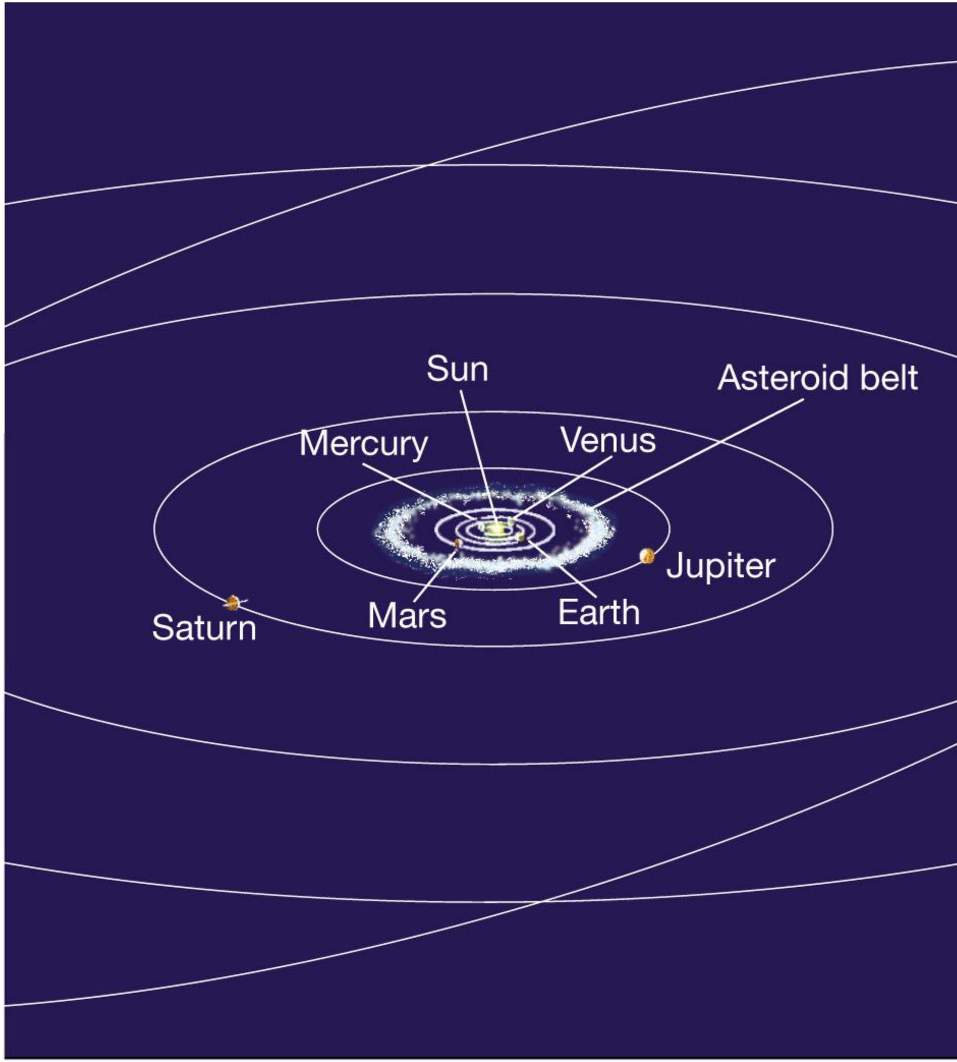


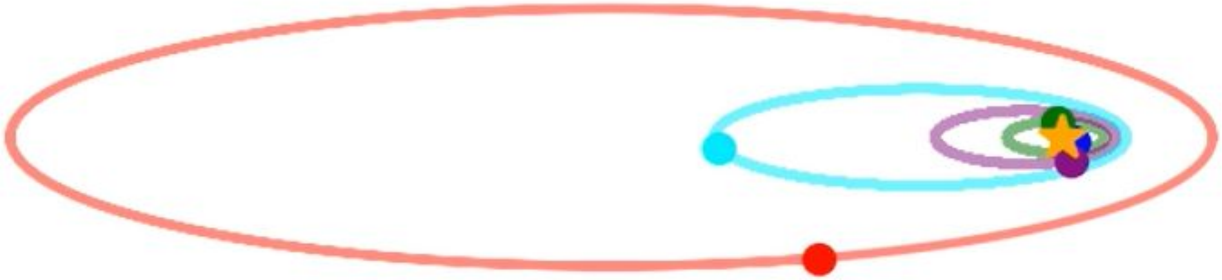




Objectives

- Simulate a solar system with multiple planets orbiting over time
- Represent transits of those planets
- Let users make different planets & create their own systems to simulate
- Make it look dope





2.
Putting it all together

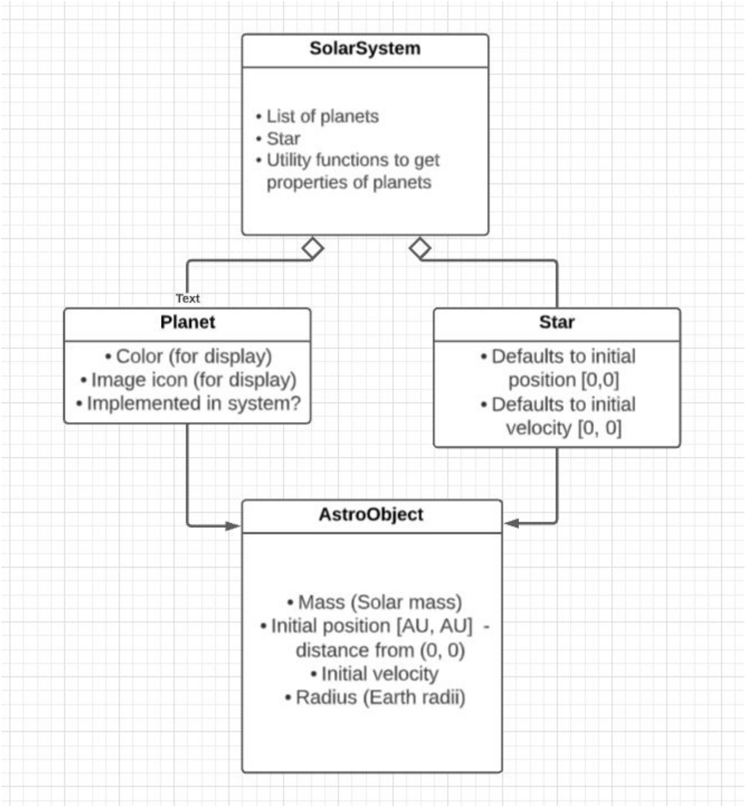
Setting Up the System

- Object-oriented programming!
- Classes for Planet, Star, Solar System
- Functions for physics (gravitation)
- Planet & Star are subclasses of AstroObject to minimize repetition

$$F = G\frac{m_1 m_2}{r^2}$$

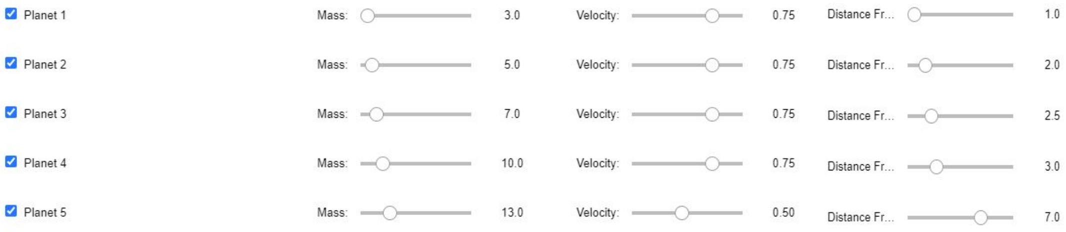
$$m_1\frac{d^2\mathbf{r}_1}{dt^2} = G\frac{m_1 m_2}{r^3}\mathbf{r}, \quad m_2\frac{d^2\mathbf{r}_2}{dt^2} = -G\frac{m_1 m_2}{r^3}\mathbf{r}$$

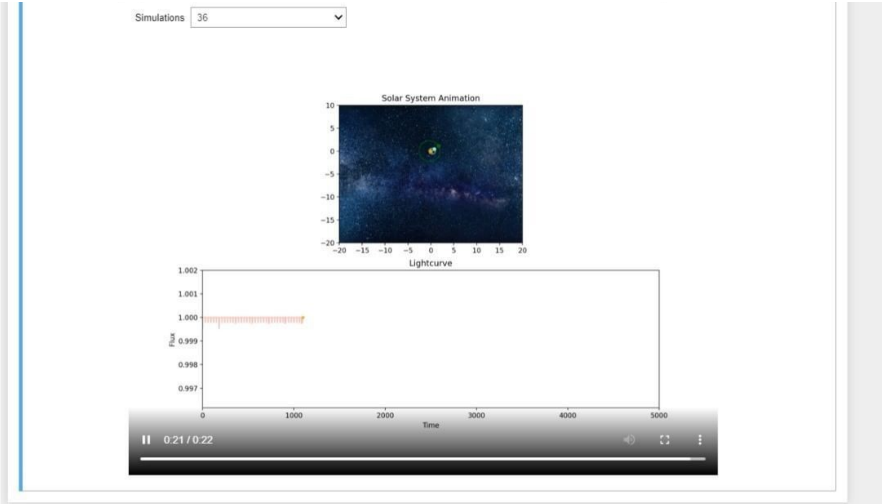
$$\frac{d^2\mathbf{r}_1}{dt^2} = G\frac{m_2}{r^3}\mathbf{r}, \quad \frac{d^2\mathbf{r}_2}{dt^2} = -G\frac{m_1}{r^3}\mathbf{r}$$



User Inputs & GUI

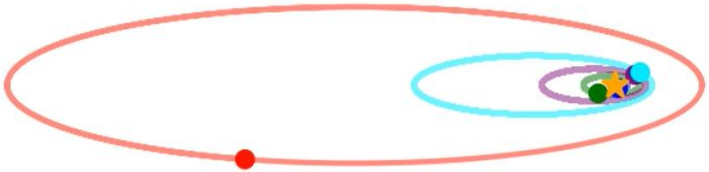
- Jupyter Widgets
- User sets mass, velocity, and distance of planet from sun
- Slider bars and checkboxes for ease of use
- Dropdown to playback generated videos within notebook

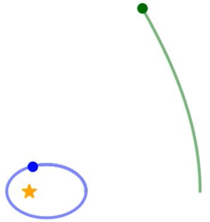


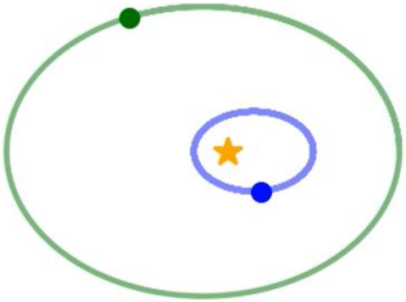


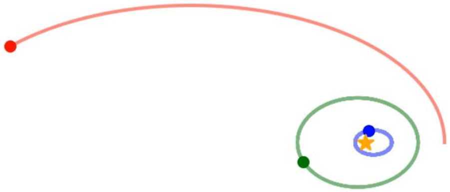
Animating Orbits

- Iterate through planets in system to access their solutions to gravitation diff. eq.
- Plot orbit path using list of calculated positions (solutions)
- Challenge: ensuring enough time & space to fully display all



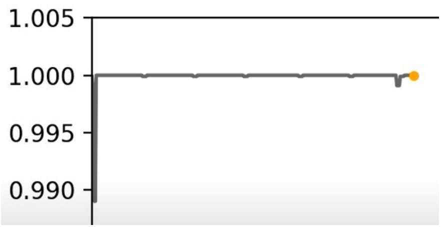


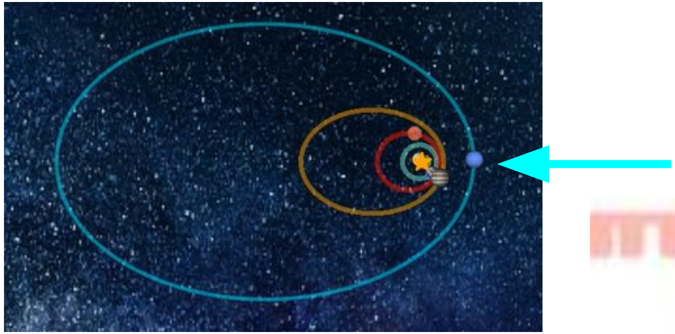




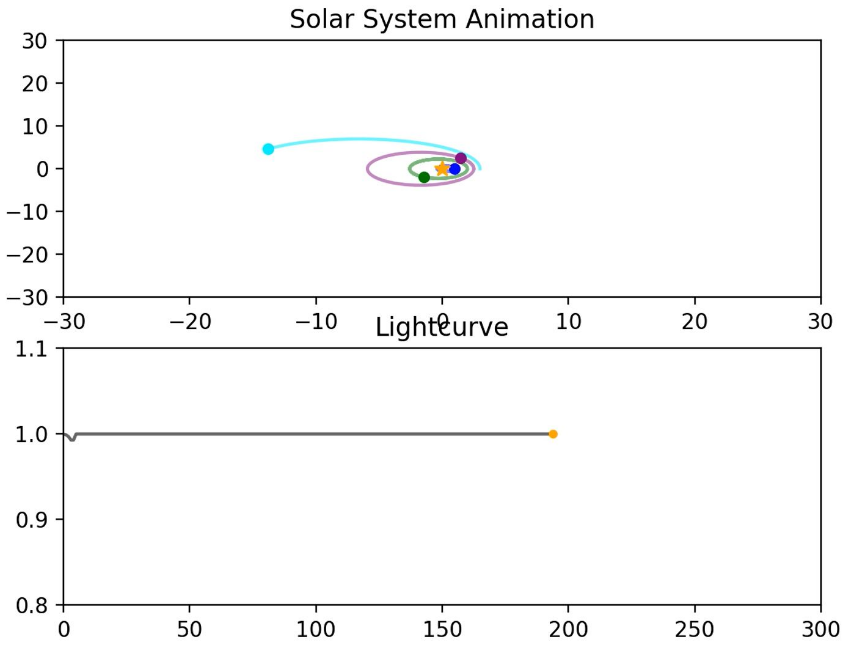
Transits

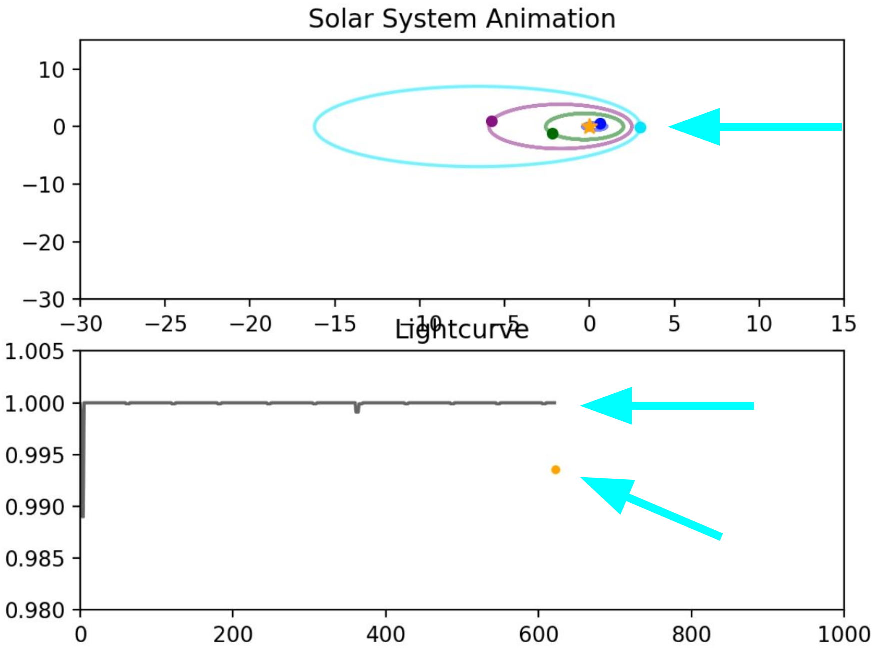
- Live-updating plot of apparent brightness -- lightcurve
- Define transit = when planet moves back to where it started (+/- a bit)
- If no planets are back to starting pos, flux = 1.0 (base)
- If a planet is back at starting pos, flux = 1.0 - (planet:star radius ratio)
- Iteration through all planets so multiple transits at same time can stack
- Challenge: updating in real time, not skipping any, & making them visible





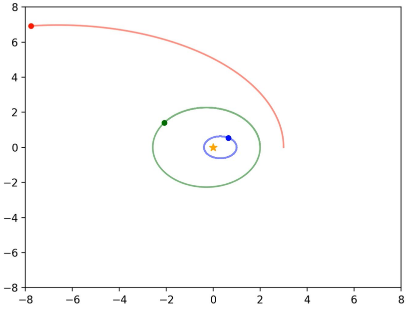
Transit Bugs

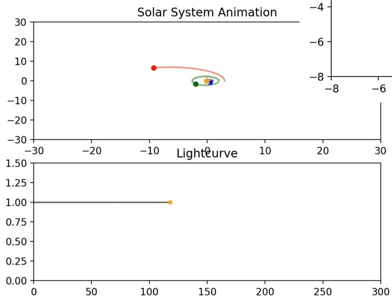


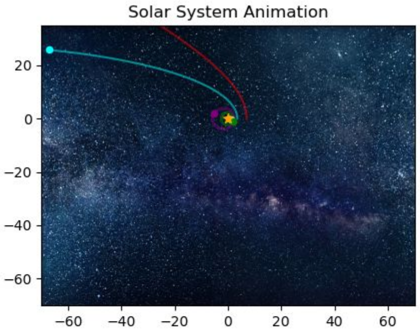


Testing & Debugging

- Not enough time to complete all orbits?
  - Very rough estimation of time needed: multiplier x distance of farthest planet
- Transits too small to be seen?
  - Auto-set y-range on plot to go down to 10 x largest transit depth
- Whole orbit can't fit in frame?
  - Axes limits automatically set based on distance of farthest planet
- Some people just want to see the world(s) burn… or freeze
  - Slider limits what masses/distances/velocities can be

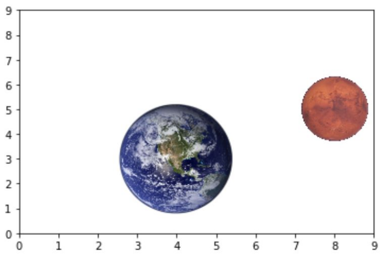


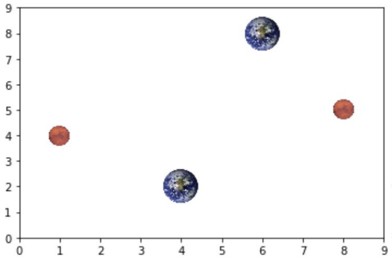


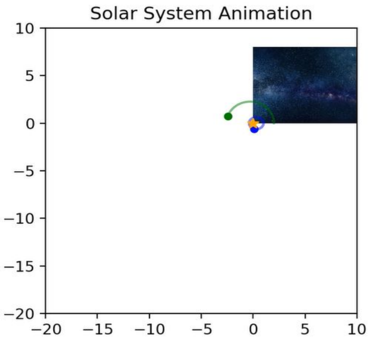


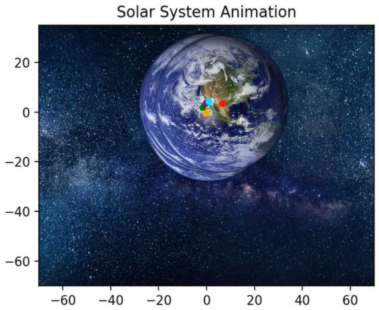
Step 5: Graphics & Display

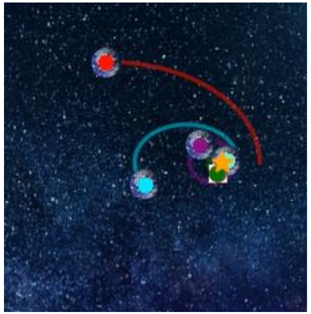
- Matplotlib; annotation box
- Background to enhance space effect
- Skins on planets: custom image icon for each
- getImage() reads image and places Annotation Box at planet's location (solution/point) for each frame

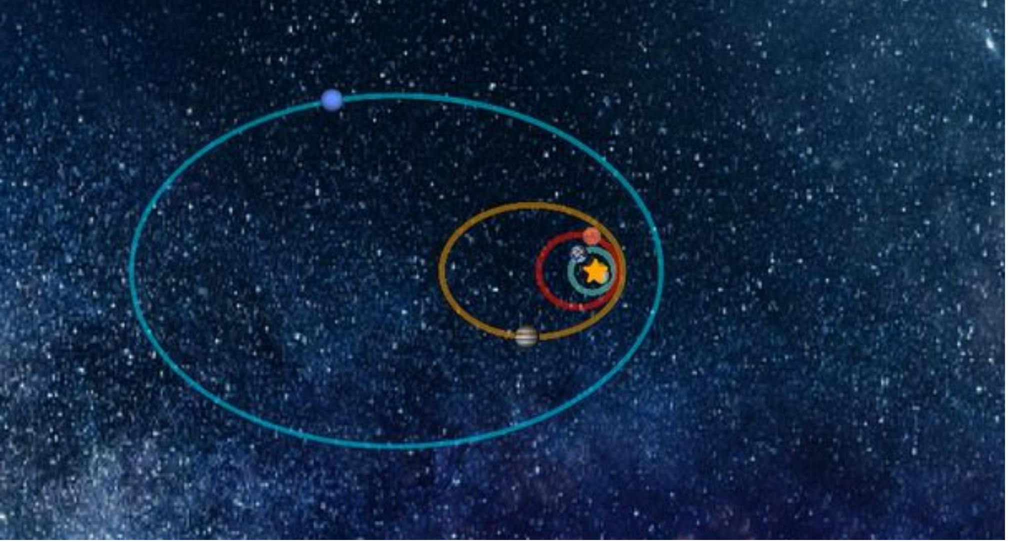


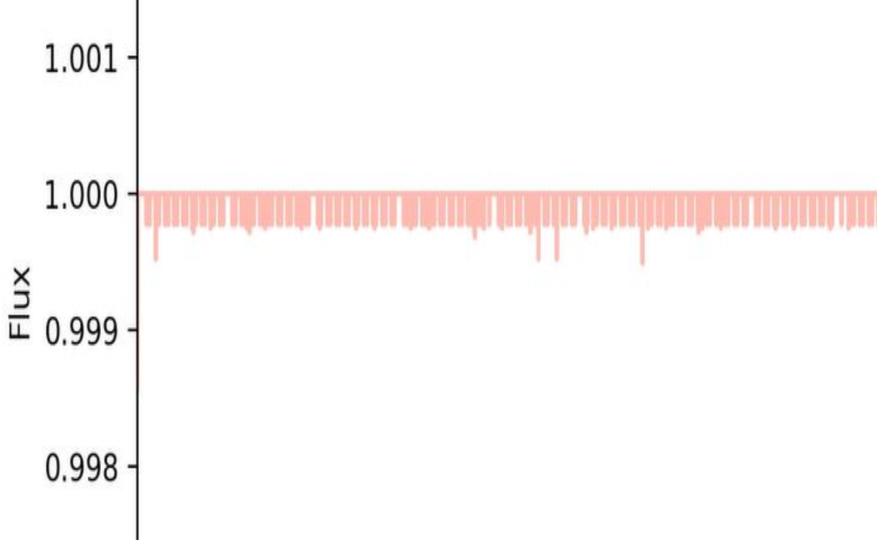




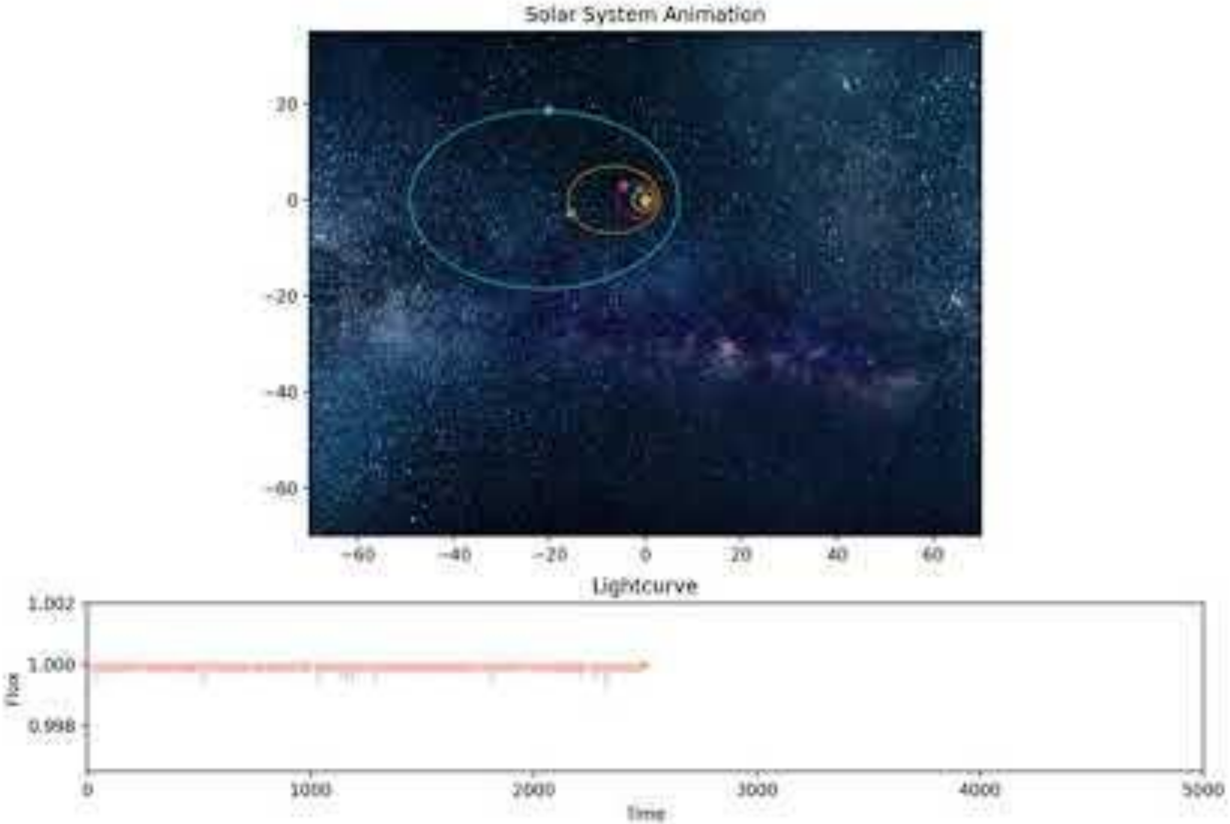




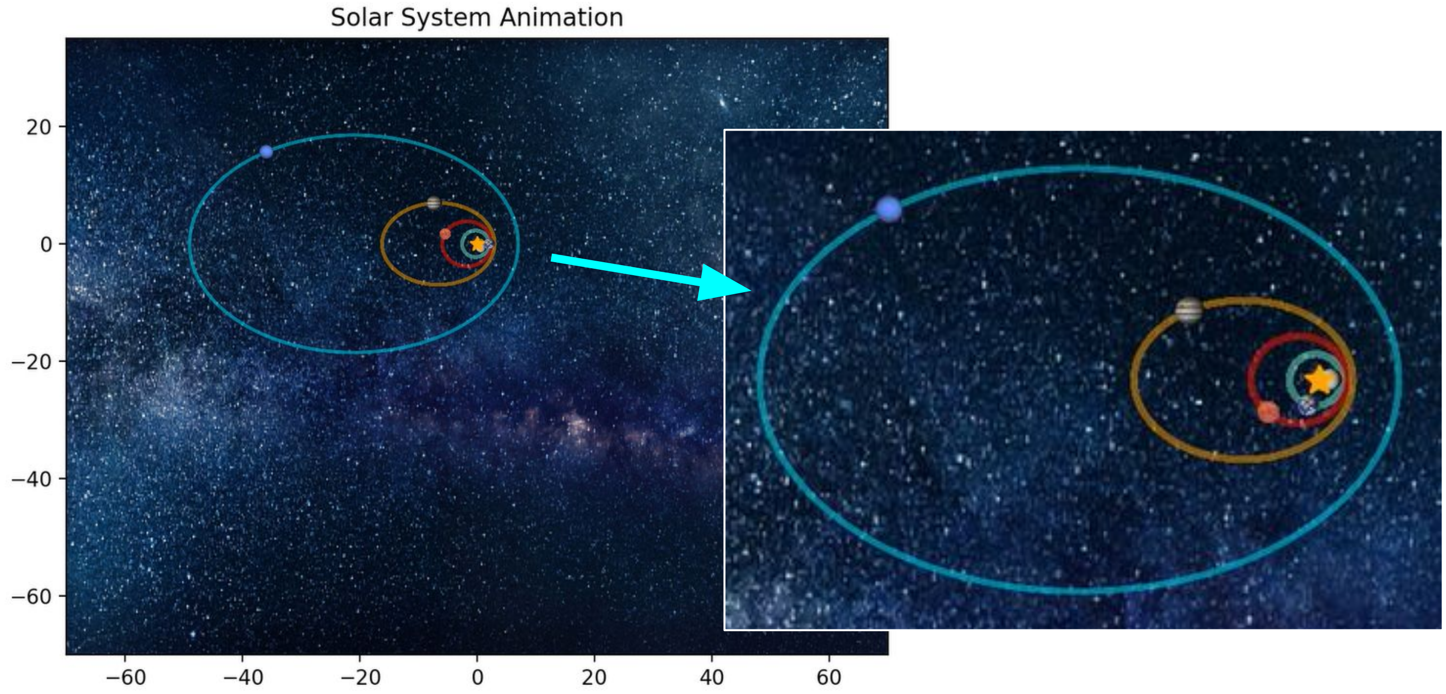


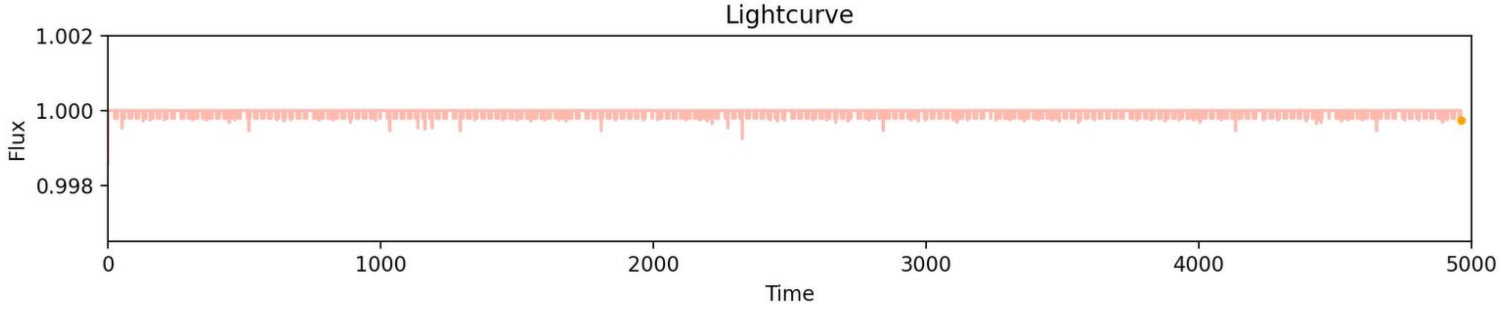


3.
The Final Product



Solar System Animation
20
0
-20
-40
-60
-60
-40
-20
0
20
40
60
Lightcurve
1.002
1.000
0.998
Flux
0
1000
2000
3000
4000
5000
Time


Solar System Animation
20
0
−20
−40
−60
−60
−40
−20
0
20
40
60


Lightcurve
1.002
1.000
0.998
Flux
0
1000
2000
3000
4000
5000
Time

References & Citations

- Tools used:
  - Astropy, Scipy, Numpy, Matplotlib, FFMPEG, Widgets
  - https://ipywidgets.readthedocs.io/en/latest/
  - https://matplotlib.org/stable/gallery/text_labels_and_annotations/demo_annotation_box.html

Helpful links:

  - https://petercbsmith.github.io/marker-tutorial.html
  - https://www.tutorialspoint.com/how-to-use-a-custom-png-image-marker-in-a-plot-matplotlib
  - https://math24.net/newtons-law-universal-gravitation.html


Questions?