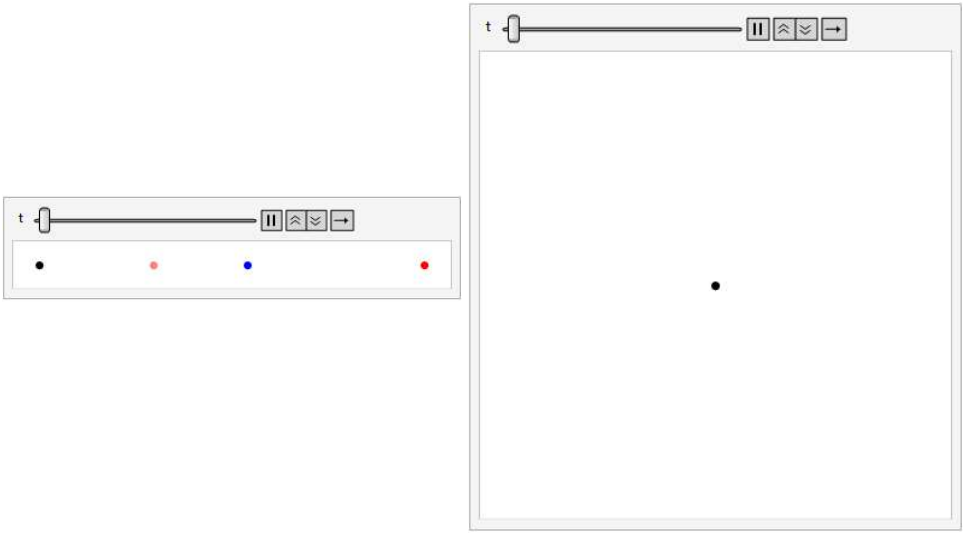
McKay Harms and Taylor Fleming
Solar System Modeling
Physics 230 Section 1
https://github.com/mcsheep13/orbitsmodel

Our goal with this project was to create an accurate model of the Solar System complete with functional orbiting planets moving at the correct speeds relative to each other. The plan was to use parametric plotting in two and three dimensions to account for axes and inclination in the solar plane, and that strategy proved to be effective. We defined functions, created plots, and used the Show command to effectively combine them into animations, then used point graphics to model the planets themselves. Although we could model the planets without the guidance of the circular orbits, it proved to be chaotic and hard for the eye to follow without the help of the ellipses, especially in three dimensions. Using the same basic strategies still allowed plenty of opportunities to use our unique coding styles and the inner and outer planet models, though similar in appearance, were coded very differently. We also used multiple Mathematica files to ease the computing burden on our computers and avoid conflicts in the Git repository.

We divided the work between us simply by splitting the planets into inner and outer planets. As we began to set up our code to be able to graph and animate them, we found other places for workload division. McKay, before we started coding our graphs, created a Mathematica notebook that included all the data necessary for our project so that we could simply call the variables from that notebook instead of coding the astronomical data every time we needed it. When we began graphing our data and looking into animating points to travel the paths of the orbits, Taylor figured out how to code a point to travel along a path and wrote a method that we used and adapted to the rest of the animations.

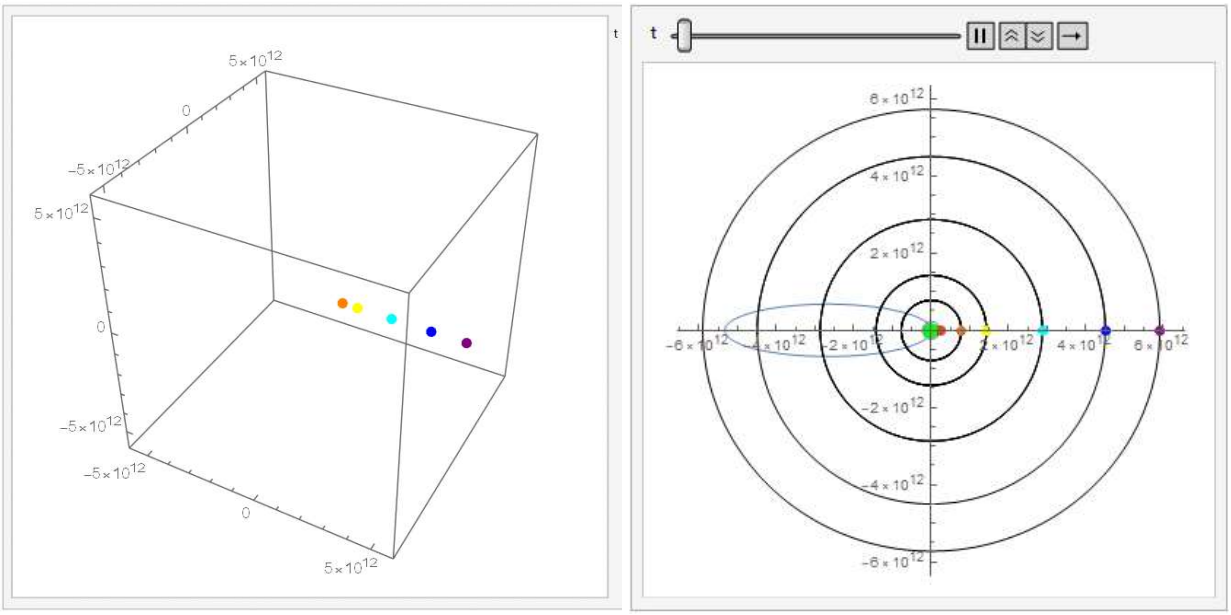


The first challenge we encountered wasn't with our code at all—it was with Git. Several commits happening one after the other corrupted our Mathematica variables file and led to some frantic research and coding as we tried to recover about an hour's worth of programming variable names and database accesses. Fortunately, Mathematica has an entire library called "AuthorTools" that has several commands helpful for recovering files in an unreadable format. It took some research and several failed attempts, but we were able to recover and export the

variable file with only a few minor losses and were much more cautious in our Git commits from that point on. As seen above, our animations didn't always go to plan. The importance of PlotRange was emphasized as, first, we see an orbiting Mercury without a fixed plot range: a point in space. With no fixed box to rotate in, the point simply stays motionless and the axes move around it. With multiple planets, the results weren't much better, and we were careful to include accurate plot ranges from that point on.

The only major modification we made to the project was our goal of combining all 3D animations into one cohesive plot. With the differences in code, it proved to be impossible without major modifications and we settled for a combined 2D plot instead. We had also at first had hopes of using ParametricPlot and LogLogPlot together to be able to account for vast orbital differences and make the system seem more cohesive. Upon further research, however, we realized that the two commands were not closely related nor easily combined.

We learned many things about Mathematica in this project that we otherwise might not have discovered through our in-class labs. Upon trying to adapt methods to help our graphs merge together, we discovered that attempting to make a parametric plot with log-log bounds was not built into Mathematica, unlike many other graphing options. In addition to this, we more fully experienced the freedom a programmer has to code similar plots and animations in ways distinct from others. Attempting to merge our 3D plots, we found that while they were similar in appearance, the choices made in coding were incompatible without heavy modification. When we exported our animations as GIFs, we discovered the limitations of animations in Mathematica as many of them ended up being choppier than we hoped, even leading to an apparent backward orbit of Jupiter. Despite these troubles, we learned some valuable lessons and came to more fully understand the amazing tool that Mathematica is.



A couple of points that were noted in our peer reviews asked about our methods to avoid the slow processing of data from Mathematica's AstronomicalData database and the differences in our methods for 3D modeling. While the file did run slowly whenever we had to pull numbers from Wolfram Alpha, the ease of using a separate variable file essentially eliminated any such difficulties. Since we only had to run the file once per coding session, the slowness was largely

unimportant and we could run and test our actual coding files with relative speed and ease. Beyond that, the concern itself is relatively insignificant since Mathematica, once a database has been loaded into a file, executes any other pulls from that same database in a matter of seconds. Our differences in 3D modeling came with the more extreme inclination on which Pluto orbits. It didn't have an effect on the 2D plots as inclination was neglected in those plots, but it messed with the plot range and t value for the parametric plot. The inner planets were unaffected by these changes, but Taylor adjusted timing for his plots and ranged his plots around Pluto's inclined orbit, making it incompatible with McKay's 3D plots for the inner planets. The animation timing to align all the orbits at the start and finish of each loop was also noted in our peer reviews. While we did try to ensure that all the planets orbited at least once per cycle by building our timing around the orbital period of the outermost planet of each of our plots, we didn't go so far as to line them all up with one another each cycle. The ratios of the orbital timing don't fit very well, so we opted for a simpler approach.