Escaping the Solar System:
Solar System & Jovian Gravitational Assist Simulator.

Tayyab Jafar **&** Kevan Rayner

10052661 **&** 10055256

April 1st, 2015

PHYS104/6

MATLAB DESIGN PROJECT 2015

Background:

One famous problem in physics and maths is the n-body problem. The n-body problem *is the problem of predicting the individual motion of a group of celestial objects interacting with each other gravitationally*. In our case, we reduce this down to the three-body problem. The three-body problem applies to the prediction of future motion of three bodies interacting with each other, *from an initial set of data that specifies the positions, masses and velocities of the three bodies*.

Newton's Law of Gravitation states that there is an attraction, a force, between every point mass in the universe that acts along in a direct line between the two masses and is expressed as, where m_1 and m_2 represent the mass of the two objects, r is the distance between the two bodies and G is the gravitational constant.

$$\vec{\boldsymbol{F}}_{12} = -G\frac{m_1 m_2}{r_{12}^3}\vec{\boldsymbol{r}}_{12}, \qquad G = 6.67 \times 10^{-11}\, N\frac{m^2}{kg^2}$$

We can express the in vector form as followed:

$$\vec{\boldsymbol{r}}_{12} = \begin{pmatrix} x_2 - x_1 \\ y_2 - y_1 \\ z_2 - z_1 \end{pmatrix}$$

From this we can determine the resultant magnitude:

$$|\vec{\boldsymbol{r}}_{12}| = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2}$$

From this we can split up the components of our force between the two masses:

$$\begin{pmatrix} \boldsymbol{F}x_{12} \\ \boldsymbol{F}y_{12} \\ \boldsymbol{F}z_{12} \end{pmatrix} = -G\frac{m_1 m_2}{((x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2)^{\frac{3}{2}}}\begin{pmatrix} x_2 - x_1 \\ y_2 - y_1 \\ z_2 - z_1 \end{pmatrix}$$

Newton's 2nd law states that for a net force, $\vec{F}$ applied to an object of mass, m, will cause the object to accelerate with an acceleration, $\vec{a}$*:*

$$\vec{F} = m\vec{a}$$

In order to solve our 3-body problem, we first start with a precise value for the positions and velocities of each body being considered in the system along with their masses. From Newton's Law of Gravitation, we can determine the acceleration of these bodies:

$$\vec{\boldsymbol{F}}_{12} = -G\frac{m_1 m_2}{|r_{12}^3|}\hat{\boldsymbol{r}}_{12}$$

$$m_2\vec{\boldsymbol{a}}_2 = -G\frac{m_1 m_2}{|r_2 - r_1|^3}(r_2 - r_1)$$

$$\begin{pmatrix} \vec{\boldsymbol{a}}_{2_x} \\ \vec{\boldsymbol{a}}_{2_y} \\ \vec{\boldsymbol{a}}_{2_z} \end{pmatrix} = \begin{pmatrix} \left[-G\frac{m_1}{((x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2)^{\frac{3}{2}}}(x_2 - x_1)\right] \\ \left[-G\frac{m_1}{((x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2)^{\frac{3}{2}}}(y_2 - y_1)\right] \\ \left[-G\frac{m_1}{((x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2)^{\frac{3}{2}}}(z_2 - z_1)\right] \end{pmatrix}$$

This may start to look very difficult, but this can be simplified down with the following relationships:

$$\begin{pmatrix} F_{x_{12}} \\ F_{y_{12}} \\ F_{z_{12}} \end{pmatrix} = \begin{pmatrix} m_1\frac{d^2x_1}{dt^2} \\ m_1\frac{d^2y_1}{dt^2} \\ m_1\frac{d^2y_1}{dt^2} \end{pmatrix}$$

This is the force for each component as shown previously. However, since MATLAB can only solve first order differential equations, we must convert our system into first order linear systems, resulting in six equations:

$$a_x = \frac{dv_x}{dt}, v_x = \frac{dx}{dt}$$

$$\begin{pmatrix} F_{x_{12}} \\ F_{y_{12}} \\ F_{z_{12}} \end{pmatrix} = \begin{pmatrix} m_1\frac{dvx_1}{dt} \\ m_1\frac{dvy_1}{dt} \\ m_1\frac{dvz_1}{dt} \end{pmatrix}, \quad \begin{pmatrix} vx_1 \\ vy_1 \\ vz_1 \end{pmatrix} = \begin{pmatrix} \frac{dx_1}{dt} \\ \frac{dy_1}{dt} \\ \frac{dz_1}{dt} \end{pmatrix}$$

This is a lot better, but there is one problem. MATLAB, and most other computers in general, can't work with infinitesimally small values so instead of dt, we must use a time step, Δt, and rearrange to get:

$$\Delta v_x = a_x \Delta t, \;\; \Delta x = v_x \Delta t$$

The product of our time step and acceleration will result in a new velocity, while the product of our velocity and time step results in a new position at each interval of our time step. This is Euler's Method in a nutshell.

Escaping the Planet:

The dynamics of this are quite simple. Let's start with the basics. For a satellite or object to escape the gravitational pull of the body it is orbit, it must reach escape velocity. For a body of spherical symmetry with radius r, where G is the gravitational constant and M is the mass of the planet or massive body, the escape velocity can be written as followed:

$$v_e = \sqrt{\frac{2GM}{r}}$$

Escape velocity is the minimum speed required for an object to break free from the gravitational attraction of the body it is orbiting. Notice that it is a scalar, not a vector quantity. How is this so? Well, we can explore this further by first knowing how to derive the escape velocity, which can be done best using the ideas of potential energy.

The Law of Conservation of Energy can be stated as the total energy in any process remains unchanged, the energy can be transformed from one form or another, transferred between objects as long as the total amount will remain constant.

$$\Delta K + \Delta U + [change\ in\ all\ other\ form\ of\ energy] = 0$$

Since we are only dealing with kinetic energy and gravitational potential energy of our satellite, where $U = -\frac{GmM}{r}$ and $K = \frac{1}{2}mv^2$. We know that the body that is being orbited has a final kinetic and potential energy of 0, so we can rearrange our equation:

$$\frac{1}{2}mv^2 + -\frac{GmM}{r} = 0$$

$$\frac{1}{2}m_1v^2 = \frac{GmM}{r}$$

$$v_e = \sqrt{\frac{2GM}{r}}$$

In our scenario, our space colony is assumed to be starting at Earth's orbit, orbiting alongside the planet around the Sun. Plugging in the values for each respective body, we find that the escape velocity for an object leaving the Solar System from Earth with respect to the gravity of the Sun, is about 42.1 km/s. To leave Earth's orbit, with respect to Earth's gravity, one requires a speed of 11.2 km/s.

Our space colony will leave Earth… close to this value, at about 40km/s (with components in all three dimensions). As it travels through the inner planets, it will feel the gravitational pull from the Sun and be pulled back to orbit around the Sun (or potentially fly into it), never to leave the Solar System. Just kidding.

What is an assist?

The gravitational assist, gravitational slingshot, or swing-by maneuver is a great tool for exploration in the solar system. The flyby technique can change the momentum and energy of a spacecraft's orbit. This technique has been used over and over again throughout the history of spaceflight. Rocket fuel is expensive and is limited by the amount of the spacecraft can carry, but with the gravitational assist, one can cut down the fuel required throughout the duration of its space flight, while gaining energy than what it initially started off with. "Whoa, back up, Law of Conservation st"- hold up, here's how it works. Here's a picture first, it will help:

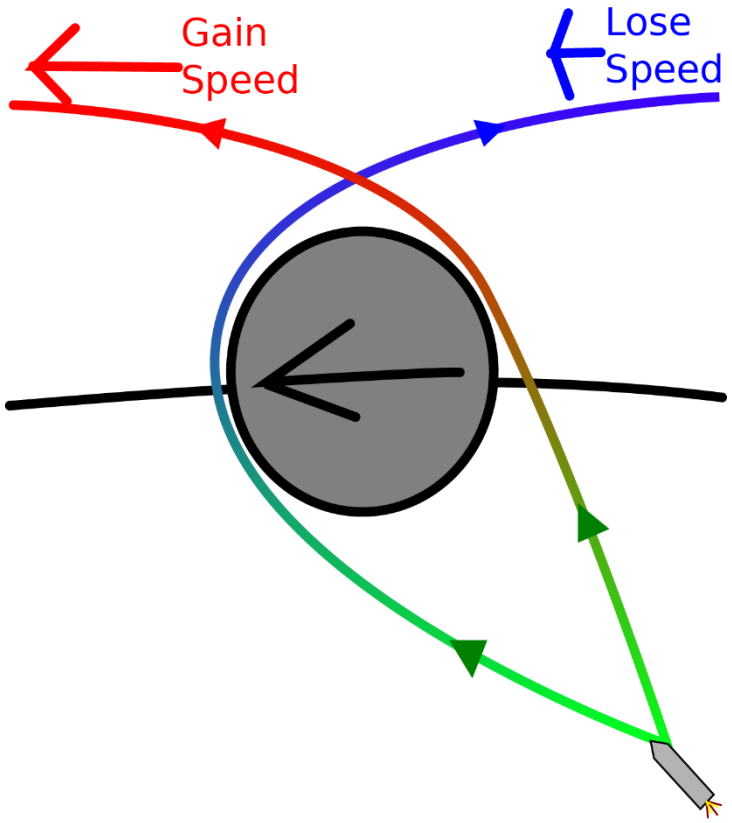


Figure 1: Gravitational Assist simplified.
http://wiki.kerbalspaceprogram.com/wiki/File:Gravity_Assist.svg

The spaceship can either gain energy or lose it, depending on the direction it is heading relative to the planet. In order to gain energy, it flies alongside the planet, to lose energy, it flies against the planet. Due to the pull of gravity from the planet, it will either speed up or slow down as a result. However, our spacecraft has an effect on the

planet as well. In order to speed up, it takes some of the planets orbital energy, since the linear momentum gained by the spacecraft is equal to the magnitude of the momentum lost by the planet. However, since our spacecraft is so minute compared to the mass of the planet, the loss of momentum by the planet is negligible.

And this is what will happen. The downside of this technique is that timing has to be exact, but luckily, our flight plan has been calculated ahead of time. After our launch, as we get pulled back due to the Sun we will *flyby* Jupiter, performing this gravitational assist. We then get pulled around the planet gaining a significant amount of energy and swing around in another direction (assuming towards Proxima Centauri).

The Adventure Begins…

Being a pseudo-second year student and having done this design project before, and with a lot of MATLAB experience, I decided to explore a little bit further on how I could improve on the model from the previous year.

Perfection is the drive behind this model. I want an accurate model of the solar system with a proper gravitational assist in three dimensions even if it takes weeks to create. I did make some assumptions, but each time I completed a milestone I thought to myself, "hmm… but planets don't travel in circular orbits… how do I make this more accurate", and sometimes getting a bit carried away… "Planets are tilted, let's throw in some axial tilts".

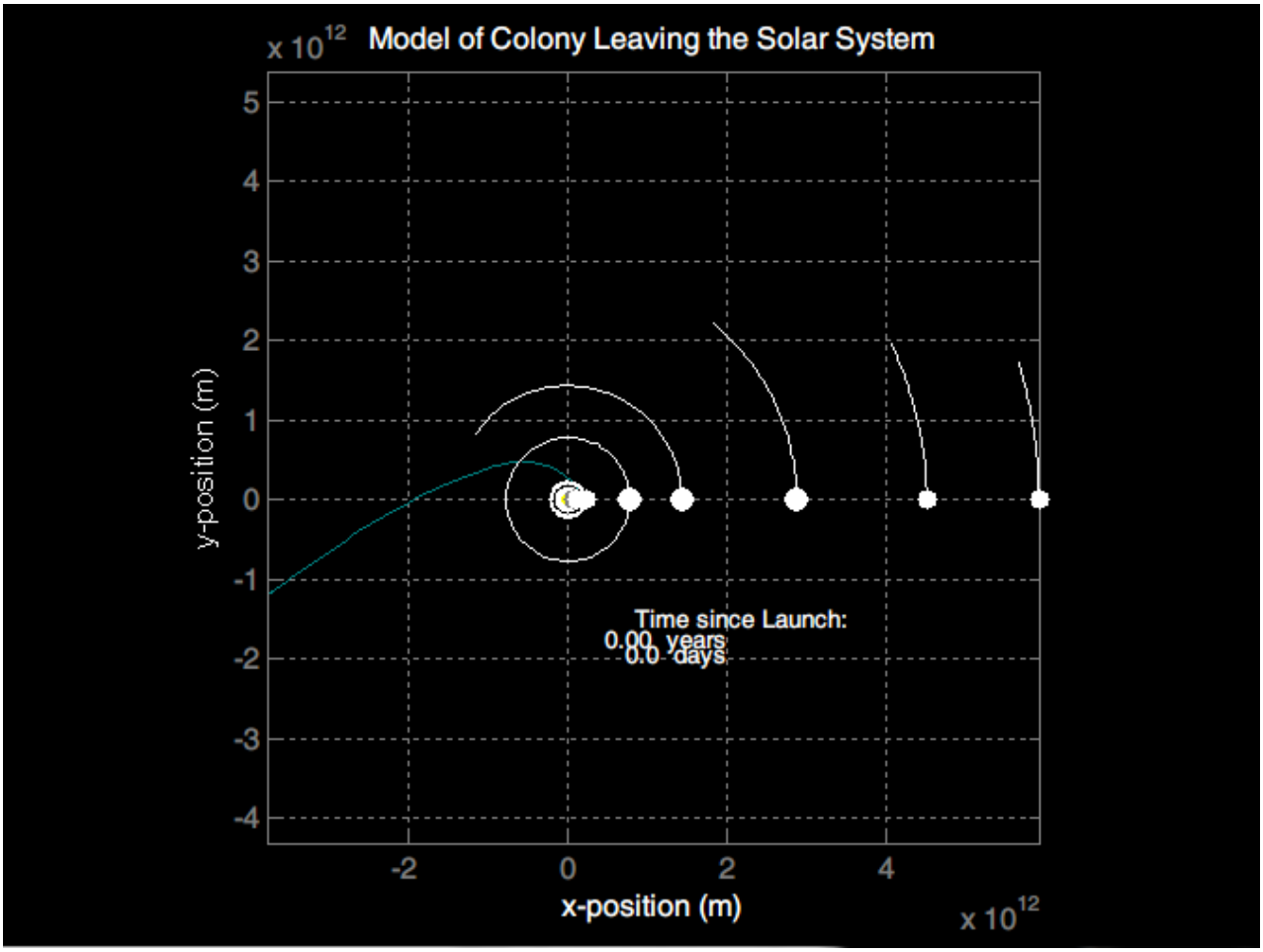


Figure 2: In the beginning…

… I started off with using the old model, see if I could add improvements to what had already existed. In the back of my mind, I had wanted a 3D model, but at this time I was working with a 2D model. When I made assumptions about circular orbits, and plotted the result, the figure above looked… 'meh'. I didn't like it, so I decided to make the orbits elliptical and on that note, I might as well start from scratch as my goal was a 3D model.

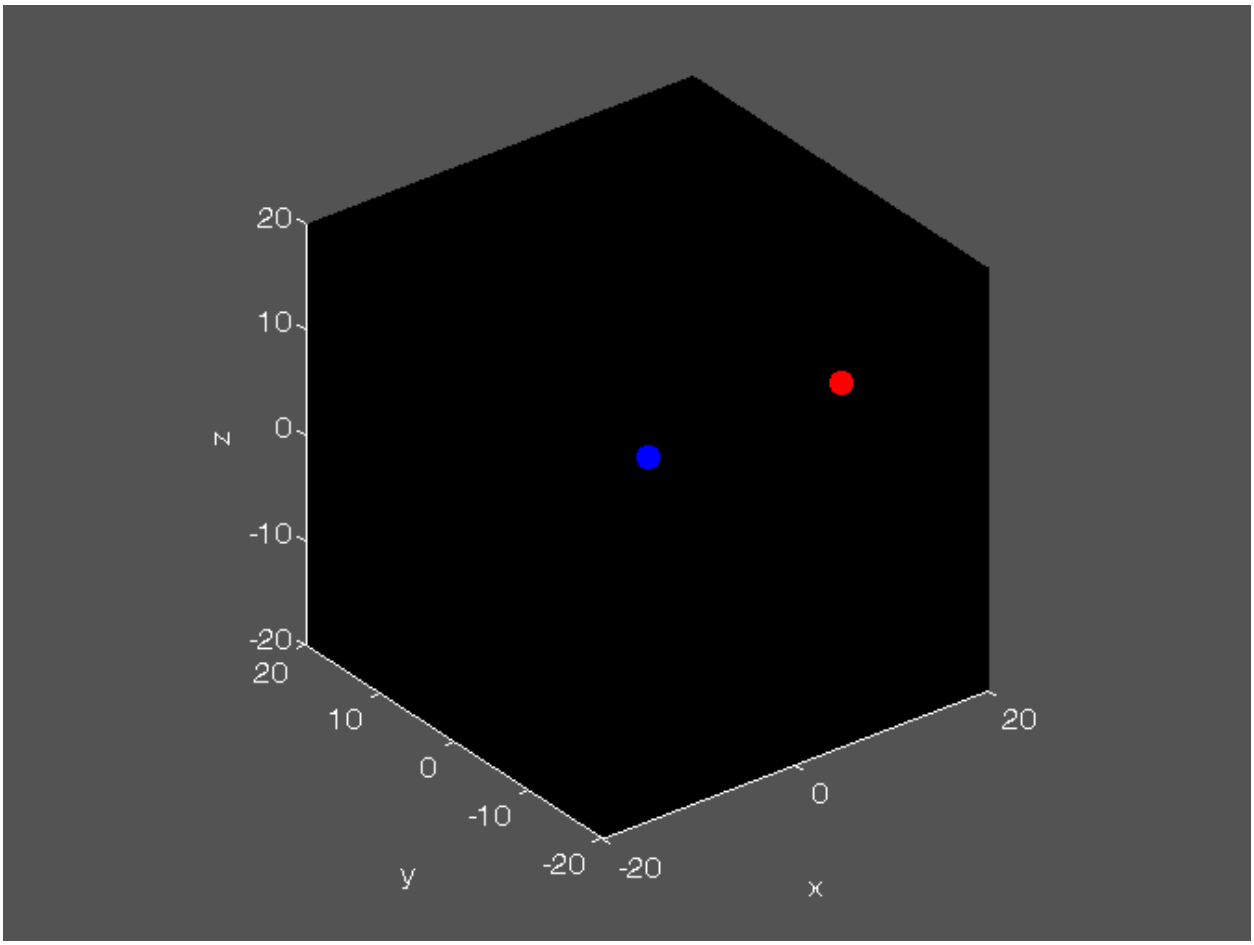


Figure 3: Simple orbit…

Started off with a simple orbit. This was to be the basis of all the modelling work. I liked this method, despite being a little bit tricky to understand but *hgtransform* is the number one reason this model is here. It is efficient on the computer and there is a lot you can do with it, but you can read that up in the references.

So, what was next? Well, at this time I was thinking of elliptical orbits… back when I was still limiting myself to a two-dimensional plane, but I digress. I used the following relationships to determine an elliptical orbital path:

$$p = a(1 - e^2)$$

$$R = \frac{p}{1 + ecos\theta}$$

P is a parameter that defines the semi-latus rectum of the ellipse and depends on the semi-major axis of the orbit and its eccentricity.

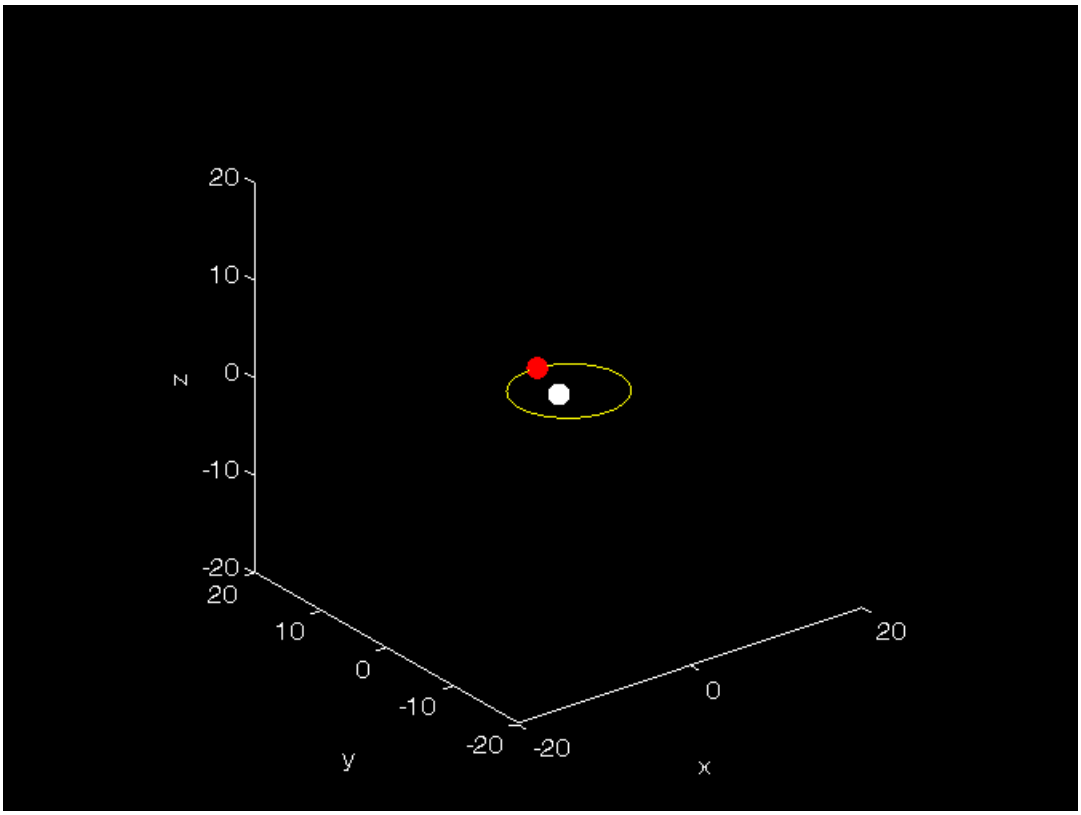


Figure 4: Mercury's elliptical orbit around the Sun.

Fast forwarding, after some time I eventually came up to the first big milestone.

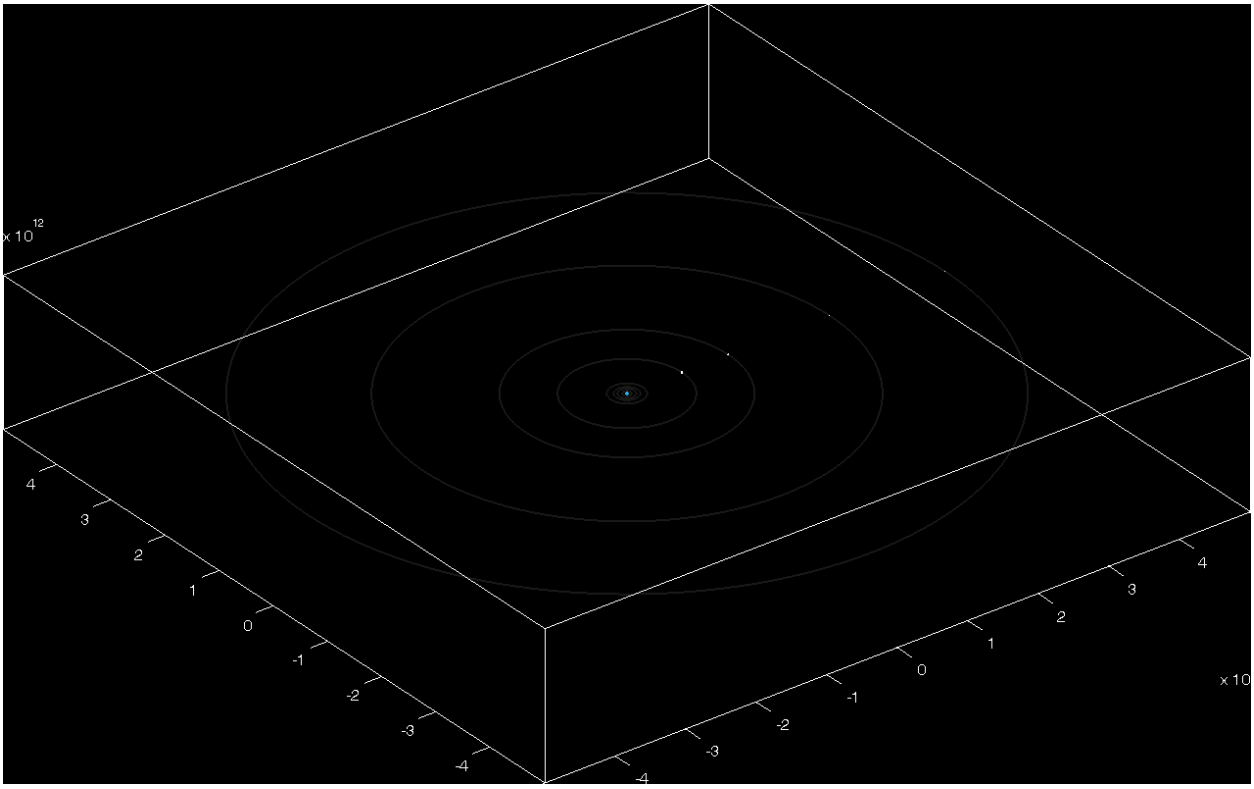


Figure 5: "oh right… space colony, not solar system model".

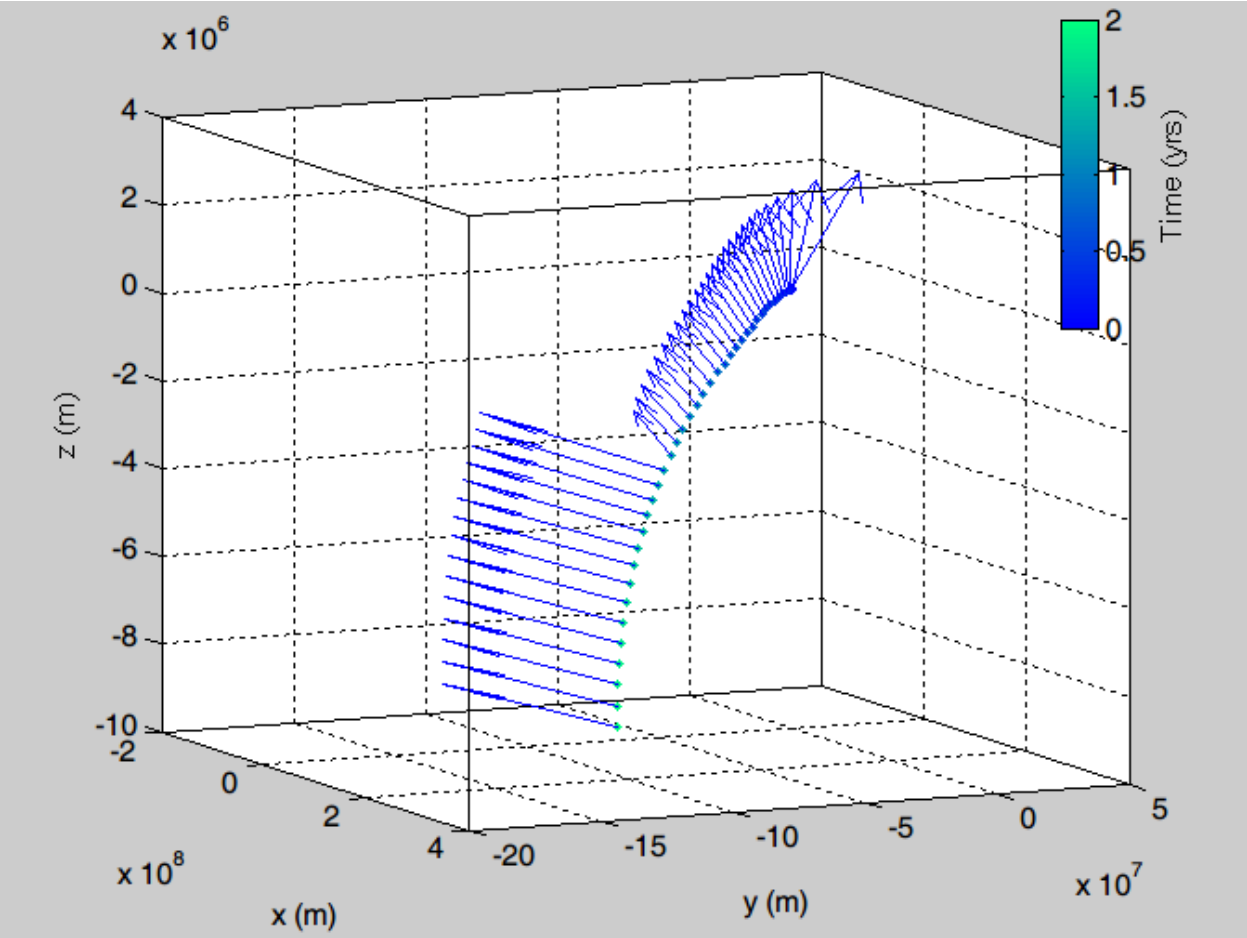
As you can see, it looks like the real deal. I started adding textures and colours to the planets, but as I did I sort of realized I was missing the main attraction, the gravitational assist… d'oh!

Fast forwarding a few more weeks, essentially starting from scratch again, all that was left was solving the three-body problem but with three dimensions. Difference between the work I did before and what I started doing now, is that I now understood how to use hgtransform. After solving the dynamics of the problem, I incorporated the results into a transform identity matrix which would supply the information for each object drawn in MatLab. From there on, it was just a matter of revision, and revision and more revision.

You can see the results. In my opinion, I will probably never be satisfied. I wanted to incorporate custom calculations, determine the speed/direction one must travel in to grab an assist around a planet, and in general find a way to make it just… "better". Sure it isn't Universe Sandbox, or Space Engine, but it is a solar system/gravitational assist simulator in MatLab… MATLAB!

There will undoubtedly be many references, a lot of this was trial and error, some of it physics and some of it computer graphics, but I think I've done my best to reference everything that was used.

Results & Analysis



Here's a quiver plot of the velocity at a constant interval throughout the space colony's journey.

I'll be honest, it's Wednesday and MATLAB is an addiction. I would love to go more in-depth with the analysis as the dynamics of the system are quite interesting. But I will not open up that can of worms, not at this point in time.

Conclusion: No effort done on the analysis, but whatever the task was, we did it. Feel free to play around with the script, let us know if you have any issues with it. Thanks for TA'ing us, it was a great time!

References:

Note: Not all references may have been used, some were readings done in preparation for the results and analysis section (eg. Energy, Gravitational Interactions, etc). At this point, I am unsure of what I used at what section. I apologize!

Strange, N. (2002). Graphical Method for Gravity-Assist Trajectory Design. Retrieved April 1, 2015, from https://engineering.purdue.edu/people/james.m.longuski.1/JournalArticles/2002/GraphicalMethodforGr avity-AssistTrajectoryDesign.pdf

Douglas, H. (1992). Interplanetary Flight Using Planetary Gravity Assist Maneuver. Retrieved April 1, 2015, from http://www.dept.aoe.vt.edu/~cdhall/courses/mech533/Reports92b.pdf

3d Ring in Matlab using Patch. (2014, January 1). Retrieved April 1, 2015, from http://stackoverflow.com/questions/27668343/3d-ring-in-matlab-using-patch

Rotate.m function fixed - File Exchange - MATLAB Central. (2012). Retrieved April 1, 2015, from http://uk.mathworks.com/matlabcentral/fileexchange/35210-rotate-m-function-fixed

Documentation. (2015). Retrieved April 1, 2015, from http://www.mathworks.com/help/matlab/ref/hgtransform.html

Images | 3D Resources (Beta). *NASA*. (2015). Retrieved April 1, 2015, from http://nasa3d.arc.nasa.gov/images

HORIZONS System. *Jet Propulsions Laboratory – NASA*. (2015). Retrieved April 1, 2015, from http://ssd.jpl.nasa.gov/?horizons

Bogan, L. (2004). Relationships of the Geometry, Conservation of Energy and Momentum. Retrieved April 1, 2015, from http://www.bogan.ca/orbits/kepler/orbteqtn.html

Reflections on Relativity. (2015). Retrieved April 1, 2015, from http://mathpages.com/rr/rrtoc.htm

What are the exact physical parameters used to calculate Mercury precession with Einstein theory? (2013). Retrieved April 1, 2015, from http://astronomy.stackexchange.com/questions/8540/what-are-the-exact-physical-parameters-used-to-calculate-mercury-precession-with

MacKay, D. (2001). Kepler's Laws. Retrieved April 1, 2015, from http://www.inference.phy.cam.ac.uk/teaching/dynamics/tex/orbits.pdf

Fowler, M. (2002). Deriving Kepler's Laws from the Inverse-Square Law. Retrieved April 1, 2015, from http://galileo.phys.virginia.edu/classes/152.mf1i.spring02/KeplersLaws.htm

Getframe opengl - Google Search. I used this for something... (2015).Retrieved April 1, 2015, from https://www.google.ca/search?q=getframe opengl&ie=utf-8&oe=utf-8&gws_rd=cr&ei=RzkPVdjtDoa2yQTC1oKYCA

Solar system simulator. Retrieved April 1, 2015, from http://www.astro.gla.ac.uk/honours/labs/solar_system/

The interaction between planets in the solar system. (2014). Retrieved April 1, 2015, from http://personalpages.manchester.ac.uk/staff/paul.connolly/teaching/practicals/solar_system.html

Gravity assist. (2015). Retrieved April 1, 2015, from http://en.wikipedia.org/wiki/Gravity_assist

N-body problem. (2015). Retrieved April 1, 2015, from http://en.wikipedia.org/wiki/N-body_problem

Three-body problem. (2015). Retrieved April 1, 2015, from http://en.wikipedia.org/wiki/Three-body_problem

Energy Potential Analysis. (2015). Retrieved April 1, 2015, from http://upload.wikimedia.org/wikipedia/commons/d/d4/Restricted_Three-Body_Problem_-_Energy_Potential_Analysis.png

N-body simulation. (2015). Retrieved April 1, 2015, from http://en.wikipedia.org/wiki/N-body_simulation

Motion in three dimensions. Retrieved April 1, 2015, from http://lightandmatter.com/html_books/lm/ch06/ch06.html

Thread Subject: Matlab animation? (2007). Retrieved April 1, 2015, from http://www.mathworks.com/matlabcentral/newsreader/view_thread/139785