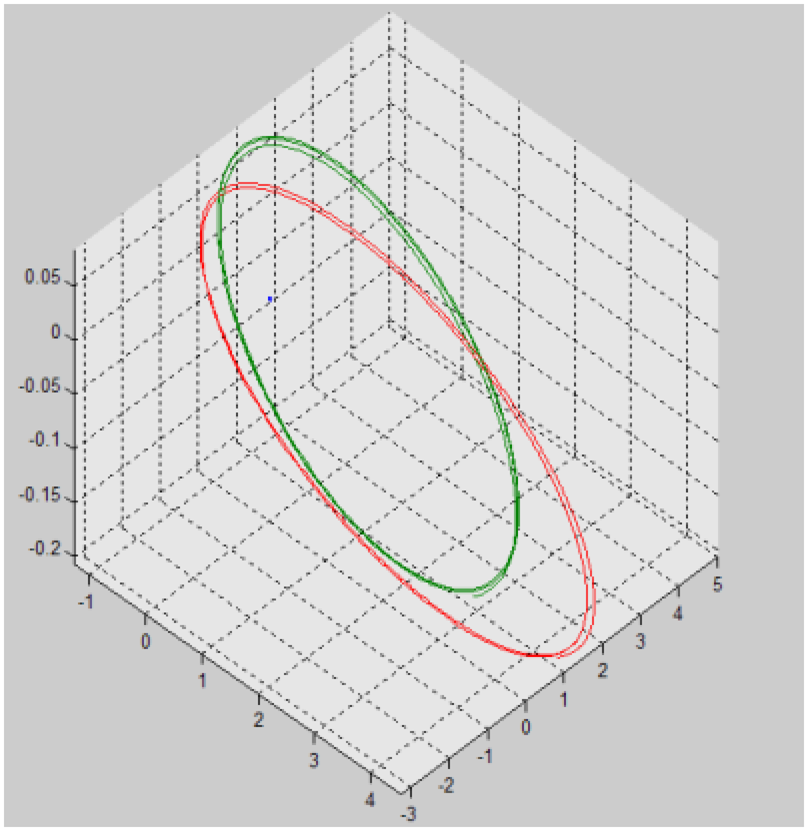
Team #12

Artesia high School

Final Report:

Moon Impact Theory and Accretion Modeling



<u>Members</u>

Shane Wilson

Emily Connelly

Casey Haldeman

<u>Teacher</u>

Randall Gaylor

Table of Contents

Introduction:

Mythology states many stories about the formation of the moon. Myths are stories that are usually of unknown origin and at least partially traditional, that ostensibly relates historical events of such character as to serve to explain some practice, belief, intuition, or natural phenomenon and that is associated with religious rights and beliefs. Science also has many stories to tell. Using science, myths can be proven to may have existed. This paper will examine the creation of the moon using an asteroid or planet (known as Orpheus) entering Earth's orbit and colliding into Earth. This collision projected large amounts of debris that would soon form the moon. Our assumption is that debris' density was not uniform and that debris was ejected into a cone, not orbiting Earth as a portional ring. The assumption is that several pieces of debris afouled into one and other because of gravitational pull. Large pieces of debris attracted smaller fragments of debris with their greater gravitational force. An accretion disk is formed around the Earth. Iron, dust, rock, and etc. are circuiting around Earth within a cementation disk. These particles then concrete over a long period of time into a larger body. Within this time period larger parcels collide with even larger segments, thus all creating one. The moon was created. The truth theory blostered appears to be with facts. Mathematical equations appear to support this conception of the moons origins as both possible and more likely.

Description of Research:

Much of our research has become irrelevant to the project as most of it was done under the assumption that we would be able to create a model that would actually make use of the data we found. We looked into all the data we would need for the lunar body and the object that would hit the Earth before we tackled trying to make a model that could actually collide them.

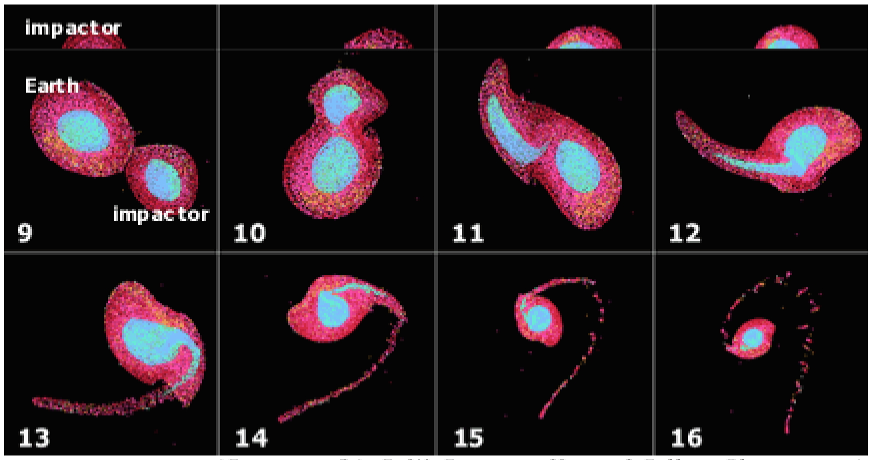
Most of our beginning research consisted of theories on how the moon was formed. There have been many theories on this throughout history from “God did it” to a small planet being captured in Earth's orbit, but we've been focusing on one in particular. This theory involves a large, roughly Mars-sized object violently colliding with the Earth. This theory was published in the 1970's by Dr. William K. Hartmann and Dr. Donald R. Davis. Before this, a theory by Charles Darwin's son, George H. Darwin was widely believed. This theory involved a much larger Earth that began to spin so rapidly that chunks flew off of it and formed the moon. The collision theory is a much more plausible hypothesis as it accounts for the moons comparatively low density and mass. “Earth has a mean density of 5.5 grams/cubic centimeter, but the moon has a density of only 3.3 g/cc. “ (Hartmann) At the time of impact it is believed that most of the liquid iron that resided on the small, young Earth had seeped into it's liquid core. When the object hit the earth must have only thrown the rocky mantle into the ejection cone, and the moon was then formed from the accretion of this matter. The planetary capture theories failed also when we discovered that the moon had the same oxygen isotopes as Earth. These theories tried to explain that the moon must have formed in an area with low iron content before it was captured in orbit, but the isotope content proved that the moon had to form in the same area as the Earth.

With the overwhelming evidence for the accretion theory we decided to try and verify it with a model. This is where we began to hit snags. The n-body physics and mathematics were beyond our math and physics understanding and training. We first had to begin researching how we could even make bodies interact in space. The equation we were given by our local expert was:

$$F_1 = m_1 A \quad F_2 = m_2 A \quad A = \frac{G m_1 m_2}{\left(r_1 - r_2\right) \Delta V}$$

In this equation *r* is the position in space, *m* is the mass of the object, *G* is the gravitational constant, *V* is velocity, and *F* is the force exerted by body one on body two. *A* is the equation we use to combine the effects of mass, velocity, gravity, and distance from the other object to calculate how much force the object can exert on another object. This particular equation only works with two objects at once. However, using a computer simulated algorithm it can be used with more than two objects. The calculation has to be done on each object with every other object in the space. The results can be then printed to screen by reporting the new positions and their rates of change. The python code we are currently working on uses a similar concept.

A computer model has been done in the past by the man who came up with the impact theory. Hartmann and a team of astrophysicists created a computer model to validate the theory. We hoped to re-validate the model with our own. Below are their results:



(Courtesy of A. G. W. Cameron, Harvard College Observatory.)

The two cores fuse upon impact. This explains the moon's lack of an iron core and it's lower density.

Other models have been made on accretion, but few others actually deal with the moon. For a while we contemplated a gas accretion model. A model was done in 2003 by Mayer and Wadsley, but it was only two dimensional and was being used to validate another theory altogether. Here is a screenshot of their results:

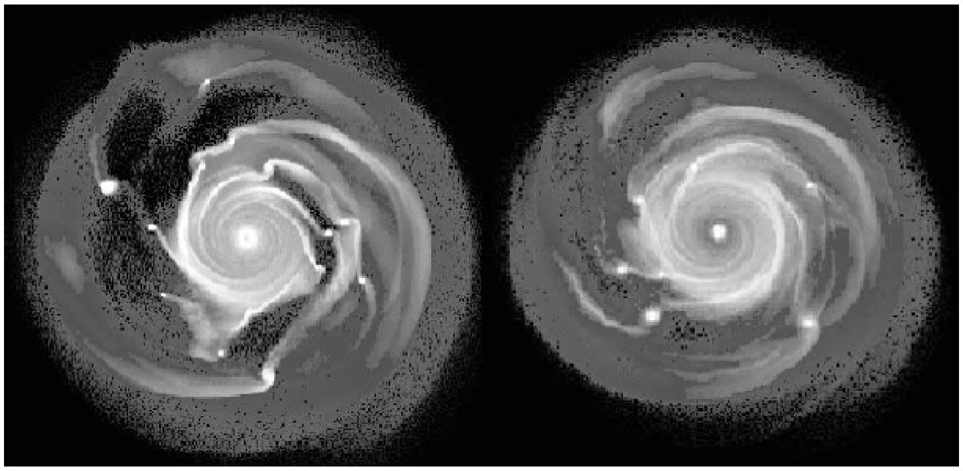
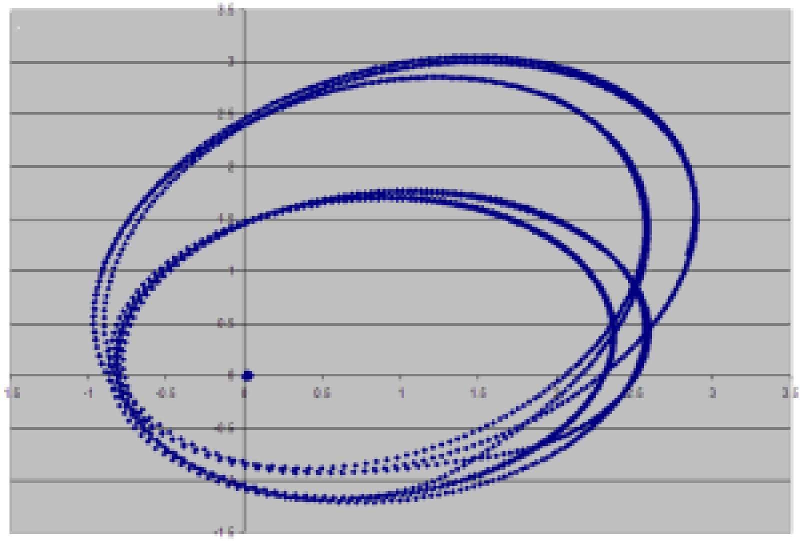


FIG. 5.—Gray-scale density maps of run DISH1 (see Table 1) after 350 yr, when the locally isothermal equation of state is used throughout the entire calculation (*left*) and when the simulation switches to an adiabatic equation of state (*right*) once the overdensities have grown past some threshold (see text, § 4.1) Brighter shades are for higher densities (densities between 10^{-14} and 10^{-6} g cm^{-3} are shown using a logarithmic scale; the same applies to all density maps shown in this paper), and the disks are shown out to 20 AU. [*See the electronic edition of the Journal for a color version of this figure.*]

Project History:

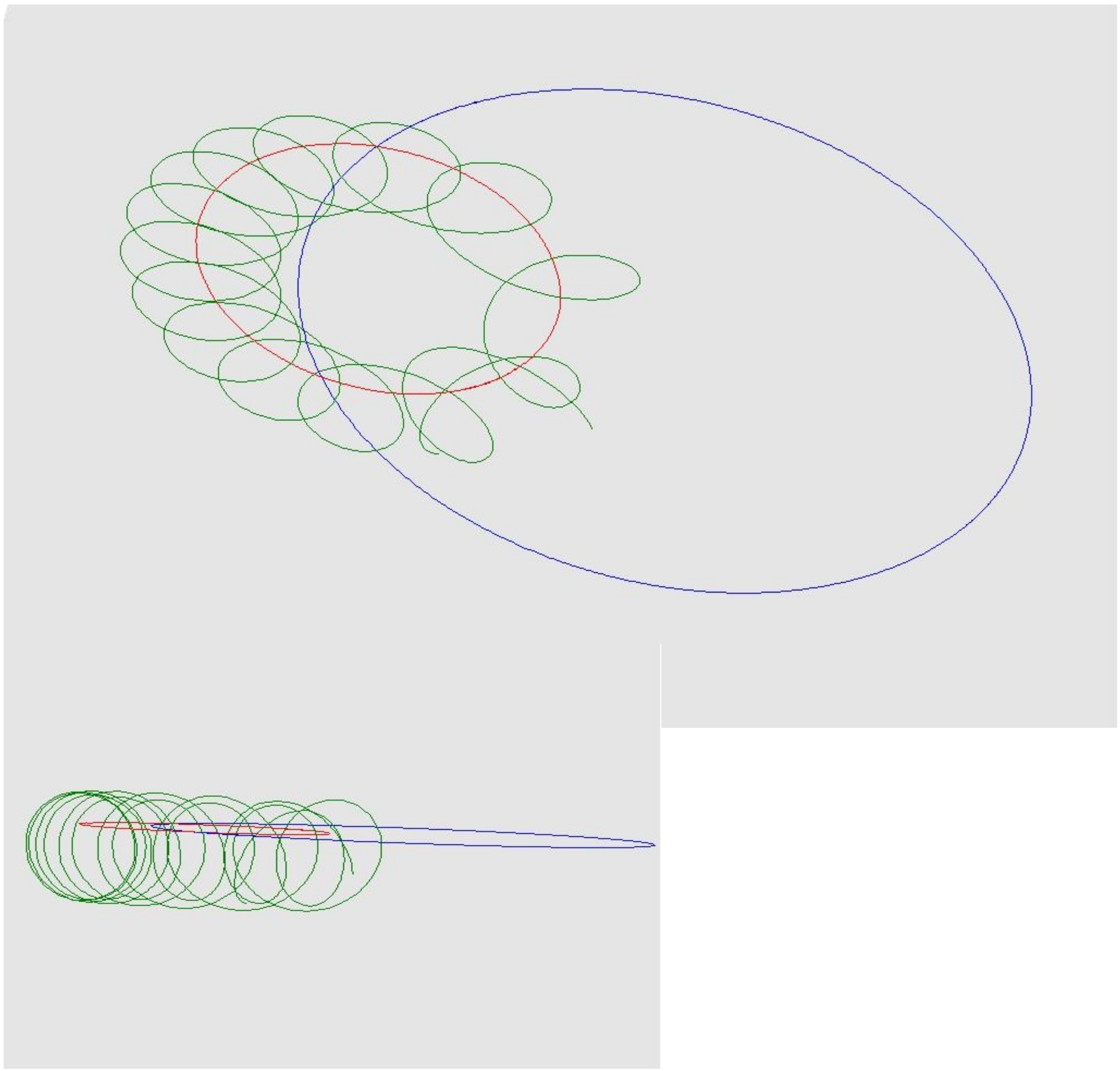
Our team first began to form with two of the members from our fairly successful team last year which disbanded over the summer. Some members were gained and lost throughout the project. We had some trouble with topic selection beginning with a global warming simulation, but we decided that would be too involved. We were introduced to the moon accretion theory and began some research. At Glorieta we spoke to some mentors and decided to stick with the accretion model. We were still having some trouble figuring out exactly how to take on such a programming challenge at the time of the interim, so we decided to continue research and see where that got us. We were introduced to SAGE math to program with Python and display it in SAGE, but we soon abandoned it for SAGE was a linux only program, which ran into problems with the schools internet filter. A python model for the solar system was shown too us shortly after, and immediately we began to tear it down and use its gravitational model. Soon we had it modified to our liking and it began to produce results.

Past:



This is one of the first models we graphed in Xcell by just pasting the coordinates from our Python script. Is only two dimensional.

Present:



This is our current model. We are importing the results from Python into Matlab and graphing it in three dimensions.

Evolution of Program Model:

The problem that we had taken on was so intense that there was no way we could have finished it. Our pseudo code evolved to represent this fact. We started from the ground up, what was needed for a space accretion model to take place? Thus a number of bodies must already be in orbit. To be able to successfully create a realistic model for accretion to take place, the environment first had to be setup, therefore we moved to create this environment. The pseudo code begins at this point. For our model to run we would have first develop the environment. (Refer to Appendix a1 for complete pseudocode.)

Developing the environment for our simulation became a big problem. How could we create a mass, such as the Sun, which other bodies orbited? What language could we possibly use? After evaluating several computer modeling languages, the answer was simple: Python.

```
def main():
n = int(sys.argv[1])
bodies = [sun, body1, body2]
offset_momentum(bodies)
outString = ""
for b in bodies:
outString += ("%f,%f,%f," % (b[0], b[1], b[2]))
print outString
advance(bodies, 0.01, n)
```

Python allowed us to place objects on a three-dimensional plane. We were able to do this by assigning the objects an x, y, and z position relative to the origin, the Sun. We gave them a standard gravitational constant and formulated for a conservation of momentum for each body. These coordinates were the starting positions of the material before the model ran. Next we assigned each body a trajectory and mass. The code finally tells the bodies to move forward, and to print out their new position in the forms of 3D coordinates. The bodies continue to advance the number of times that the user defines. With the known coordinates at each time step recorded into a spreadsheet it is easy to cross reference and check the results of the program.

```
Sun = [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, solar_mass]

Body1 = [
    2.84143144246472090e+00,
    1.16032004402742839e+00,
    -2.03622044471123109e-01,
    1.66007664274403694e-03 * days_per_year,
    7.69901118419740425e-03 * days_per_year,
    -6.90460016972063023e-05 * days_per_year,
    0.005791938424326609 * solar_mass]

Body2 = [
    4.34336671824457987e+00,
    1.12479856412430479e+00,
    -2.03523417114321381e-01,
    -1.10726862411e-03 * days_per_year,
    -5.2801234917238e-03 * days_per_year,
    6.297573763929e-05 * days_per_year,
    .005980666130812 * solar_mass]
```

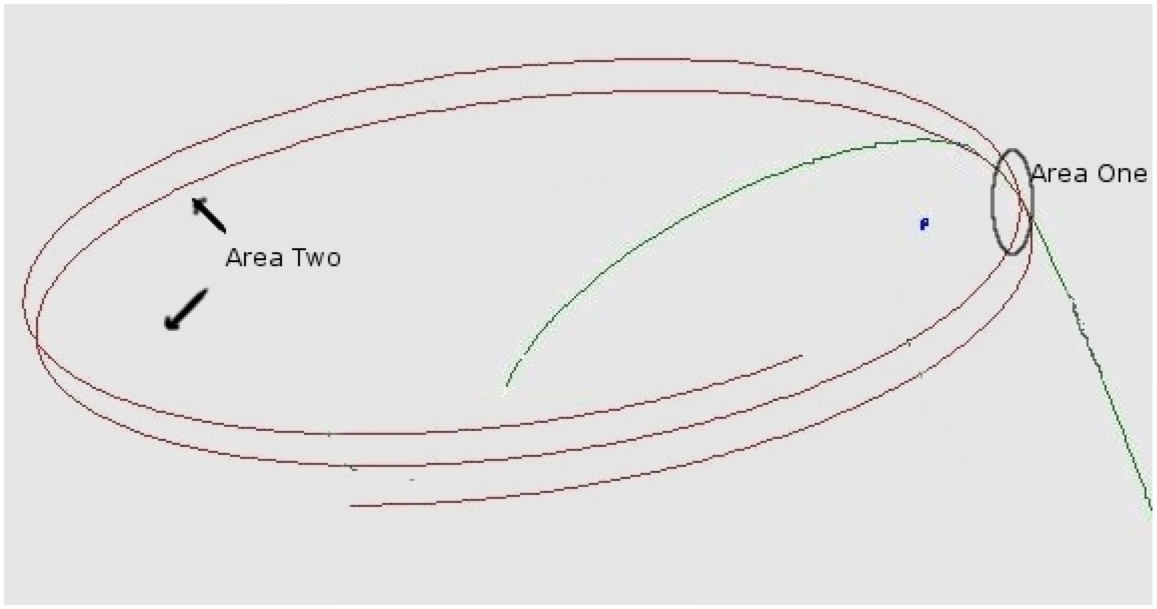
In this code example, the bodies are defined. The first three numbers for each object set their position in space (x, y, and z coordinates respectfully). The next three show their speed and trajectory change every revolution (change in x, y and z coordinates shows their trajectory). This number is set in comparison to the Earth. For example, the Earth takes 365.24 days to travel around the Sun; where as body1 has a beginning constant x-coordinate change of 1.66007664274403694e-03 multiplied by 365.24. The seventh and final number indicates the body’s mass. To be able to keep a gravitational constant each body's mass is multiplied by 39.4784176 or 4pi squared.

The rest of the code is dedicated to the actual behavior of the model (lines 8-65). The code takes each possible pair of coordinates from each object, and applies the forces of gravity exerted from one object onto the next. This changes the body's velocity and trajectory (lines 8-34). To make the model more realistic, a code for the conservation of momentum was utilized (lines 54-65). This is currently where our model ends. The code modeling accretion is not complete and fully functional at this point.

Results and Model Variations:

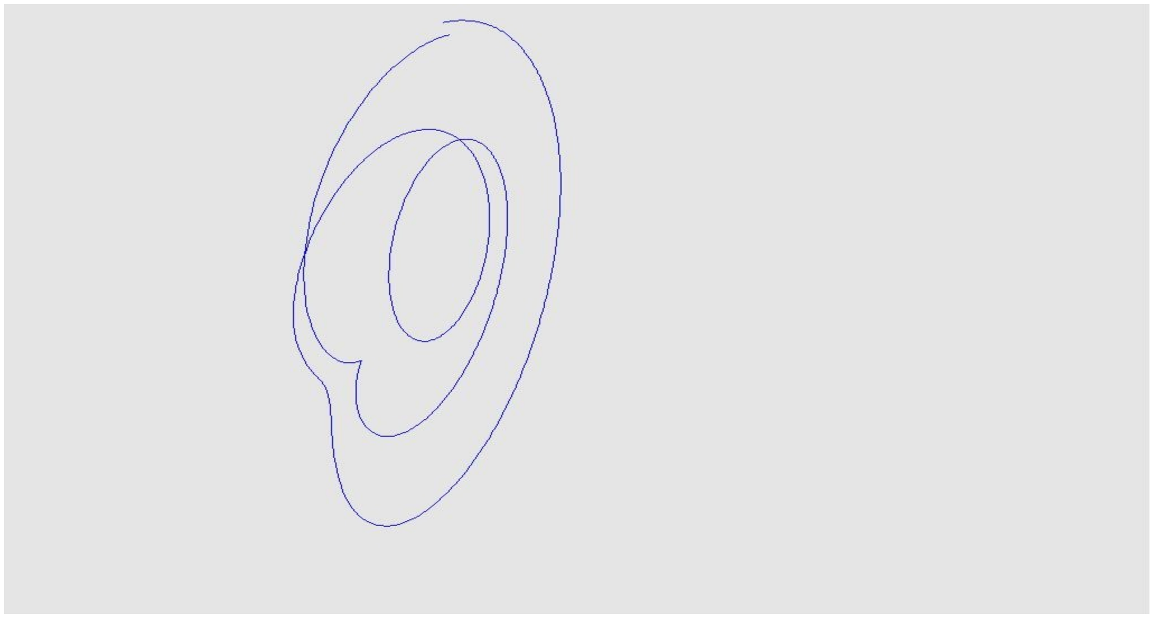
In the process of creating orbitals in space, chaos theory plays a big part. With a few simple rules in place, such as gravitational constants, conservation of momentum, and placement of bodies, a wide variety of outcomes can be derived from a simple code. The effects of one body's gravity on another take up the biggest portion of these outcomes. These outcomes can then be directly linked to scientific research to validate our model. An example of this can be found in orbital variability as well as trajectory and speed change.

After one near miss simulation, consisting of two bodies, one which had a lower mass than the other, we identified the affects of the greater gravitational of the bigger object, as the smaller object was slingshot around it. This can be seen below.

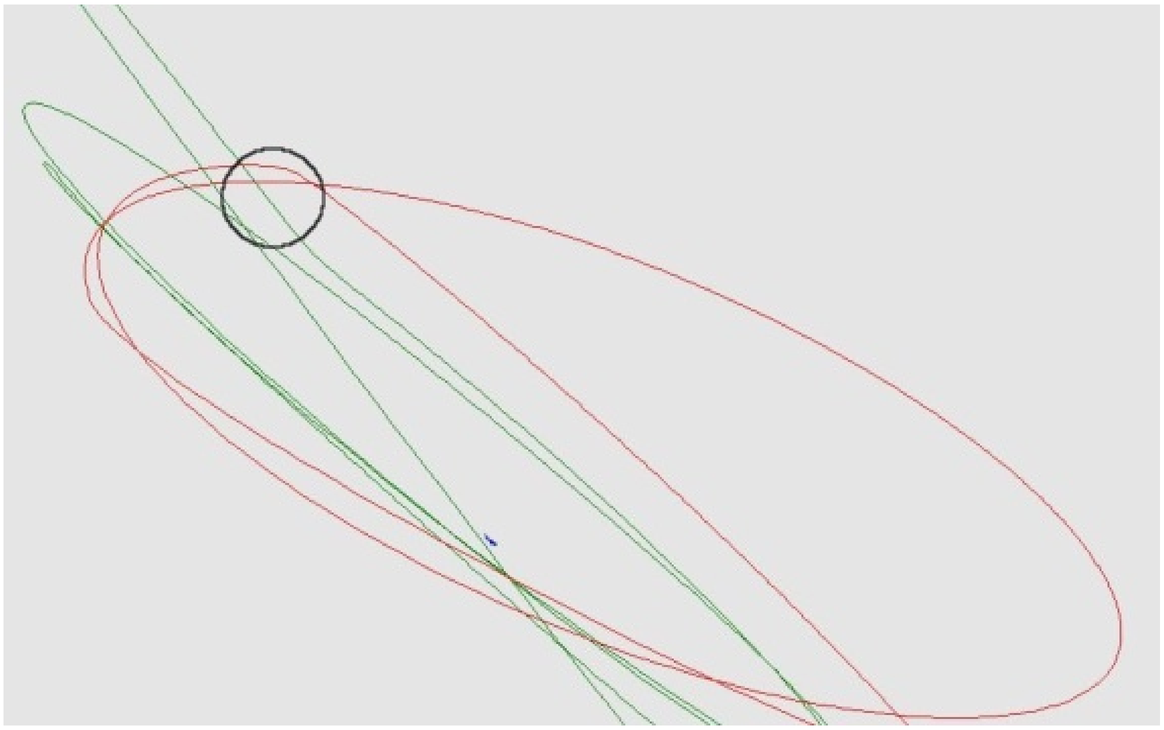


Area One shows the near miss of the bodies. The near miss affected particle two (green line) in a way that it pulled it out of orbit and sling shotted it off the backside of particle one (red). The gravitational variation in particle one's orbit can be accounted for by near miss with particle two- Area Two.

According to research by Carl Payne Tobey, the Sun (blue) is slightly affected by the gravitational pulls upon it by the orbiting bodies; so it 'wobbles' instead of staying perfectly still. Our model showed the same results when zoomed in on the Sun; a significant wobble can be identified.



Another run with bodies close to the same size, shows the affects that would happen if a near miss collision was to occur. Once again the lines represent orbits and positions of the bodies.



The effects on the orbits of the bodies is obvious. The near miss would have been a disaster if either of the planets were populated. So far we have not been able to make our model successfully accrete objects together, although we are making continuing progress in our modeling and research. By getting a better understanding of the physics behind orbitals, we hope to be able to make bodies accrete.

Conclusion:

The project proved our hypothesis correct with what data it did give. It proved it was possible that the moon was formed by particles and asteroids colliding with the Earth. Science in the real world takes a greater amount of knowledge with scientific facts. In order to obtain greater facts for the research you need fantastic knowledgeable resources, which the project lacked. If we could have had more mathematical structure, our could have explained the alignment of the fragments and pushed the project to it's full extent. With computers, resources, time, money, and thrive for scientific knowledge, anything could be created in a computer program. The computer models help us discover what we ponder. Can it be accomplished? With computers, simulations can be done that are not capable to be experimented again, such as our project, the recreation of the moon. Computer programs demonstrate what needs to be discovered. Computer programs, used to their full extent, can better our knowledge and understandings of life.

Appendix A1 (Pseudo Code)**:**

```
Create bodies in space
        -Give bodies masses
        -Give bodies trajectories

Have bodies interact
        -Create gravitational interaction
        -Create conservation of momentum

Run
        -Move bodies forward
        -Print out coordinates to spreadsheet

Develop 3D model of results
        -Transport model into Matlab
        -Graph 3D coordinates that were inputted
```

Appendix A2 (Real Code)**:**

```
import sys

pi = 3.14159265358979323
solar_mass = 4 * pi * pi
days_per_year = 365.24


def advance(bodies, dt, n):
    # Create all possible pairs first
    pairs = [(b, b2) for i, b in enumerate(bodies) for b2 in bodies[i+1:]]
    for i in xrange(n):
        for b, b2 in pairs:
            dx = b[0] - b2[0]
            dy = b[1] - b2[1]
            dz = b[2] - b2[2]

            mag = dt * (dx*dx + dy*dy + dz*dz)**-1.5
            b_mm = b[6] * mag
            b2_mm = b2[6] * mag

            b[3] -= dx * b2_mm
            b[4] -= dy * b2_mm
            b[5] -= dz * b2_mm
            b2[3] += dx * b_mm
            b2[4] += dy * b_mm
            b2[5] += dz * b_mm

        outString = ""
        for b in bodies:
            b[0] += dt * b[3]
            b[1] += dt * b[4]
            b[2] += dt * b[5]
            outString += ("%f,%f,%f," % (b[0], b[1], b[2]))
        print outString



```

```
def energy(bodies):
    e = 0.0
    bodies2 = bodies[1:]
    for b in bodies:
        e += 0.5 * b[6] * (b[3]*b[3] + b[4]*b[4] + b[5]*b[5])
        for b2 in bodies2:
            dx = b[0] - b2[0]
            dy = b[1] - b2[1]
            dz = b[2] - b2[2]
            distance = (dx*dx + dy*dy + dz*dz)**0.5
            e -= (b[6] * b2[6]) / distance
        del bodies2[:1]

    return e


def offset_momentum(bodies):
    global sun
    px = py = pz = 0.0

    for b in bodies:
        px -= b[3] * b[6]
        py -= b[4] * b[6]
        pz -= b[5] * b[6]

    bodies[0][3] = px / solar_mass
    bodies[0][4] = py / solar_mass
    bodies[0][5] = pz / solar_mass



sun = [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, solar_mass]

body1 = [
    2.84143144246472090e+00,
    1.16032004402742839e+00,
    -2.03622044471123109e-01,
    1.66007664274403694e-03 * days_per_year,
    7.69901118419740425e-03 * days_per_year,
    -6.90460016972063023e-05 * days_per_year,
    0.005791938424326609 * solar_mass]
```

```

body2 = [
    4.34336671824457987e+00,
    1.12479856412430479e+00,
    -2.03523417114321381e-01,
    -1.10726862411e-03 * days_per_year,
    -5.2801234917238e-03 * days_per_year,
    6.297573763929e-05 * days_per_year,
    .005980666130812 * solar_mass]



def main():
    n = int(sys.argv[1])
    bodies = [sun, body1, body2]
    #bodies = [sun, body2]
    offset_momentum(bodies)
    outString = ""
    for b in bodies:
        outString += ("%f,%f,%f," % (b[0], b[1], b[2]))
    print outString
    #print "%.9f" % energy(bodies)
    advance(bodies, 0.01, n)
    #print "%.9f" % energy(bodies)

main()
```