A|steroid
M|odeling and
P|rediction

Authored By: Jake Therrien, Zach Levesque, Kimberly Worcester, Luigi Pasto, Jeff Puzzo, Jon Nappi, and Justin Remillard

Sea Grant New Hampshire

AMP

THE UNIVERSITY OF NEW HAMPSHIRE 1866

Contents

Background:

Asteroids are metallic and/or rocky bodies that do not have atmospheres. They orbit the Sun but cannot be classified as planets because of their relatively smaller size. Most of them are congregated in the main asteroid belt. The belt is a wide ring located from 2 to 4 astronomical units (A.U.), placing it between the orbits of Mars and Jupiter. 1 A.U. is the average distance between the Earth and the Sun, which is about 150 million km. The largest asteroid in the main belt, Ceres, is 1,000 kilometers in diameter and the smallest asteroids are just a couple centimeters across.

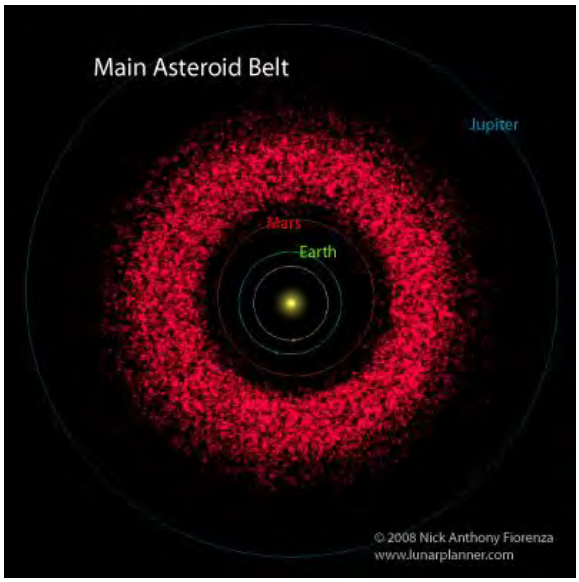

Figure 1: A model of the main asteroid belt with respect to the Sun, Earth, Mars, and Jupiter

Asteroids are the left over material from the creation of our solar system 4.6 billion years ago. They are thought to have been prevented from combining into planet-sized bodies by Jupiter's immense gravity. If all of the asteroids were allowed to coalesce they would produce a body that is about 1,500 km in diameter, less than half the size of Earth's moon. Most of the

asteroids in the main belt have stable elliptical orbits, revolving around the Sun in the same direction as the Earth. However, they take three to six Earth years to completely circle the Sun.[5]

Some asteroids roam the inner region of the solar system. Near-Earth asteroids (NEAs) are asteroids whose orbits bring them within 1.3 A.U. from the Sun. Potentially hazardous asteroids (PHAs) are NEAs with a diameter greater than 140 meters and pass within 0.05 A.U. of the Earth. There are about 10,000 known NEAs, of which 1,340 have the PHA designation.

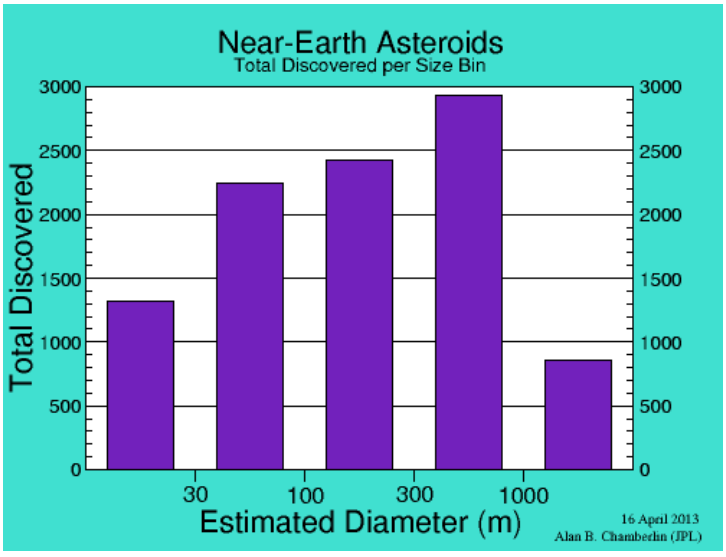

Figure 2: A table showing the amount of near Earth asteroids according to their diameter

NEAs' orbits are relatively young compared to those of the main belt asteroids. These orbits are produced by gravitational interactions with the Sun and planets along with collisions with other asteroids. There are three categories of NEAs: 1221 Amor, 1862 Apollo, and 2062 Aten. Amor asteroids cross Mars' orbit but do not reach Earth's orbit. Apollo asteroids cross Earth's orbit with a period greater than one year. The final category of NEA is an Aten. These asteroids cross Earth's orbit with a period less than one year. The largest NEA that is presently

known is 1036 Ganymed with a diameter of approximately 41 km. On average, asteroids travel at a speed of about 25 km/s, and cause destruction whenever they collide with another object.[1][2]

On June 30, 1908, an asteroid 100 m in diameter burst in the air over an isolated region of Siberia. Its blast destroyed more than 2,000 square kilometers of forest. [5] More recently, on February 13, 2013, a 20 m asteroid impacted Earth in Russia traveling at 18.6 km/s. Its impact injured about 1,500 people and damaged over 7,000 buildings. The low approach angle through Earth's atmosphere dissipated much of its energy and prevented greater destruction. [3] Currently there are 861 large NEAs that are known to be greater than 1 km in diameter.

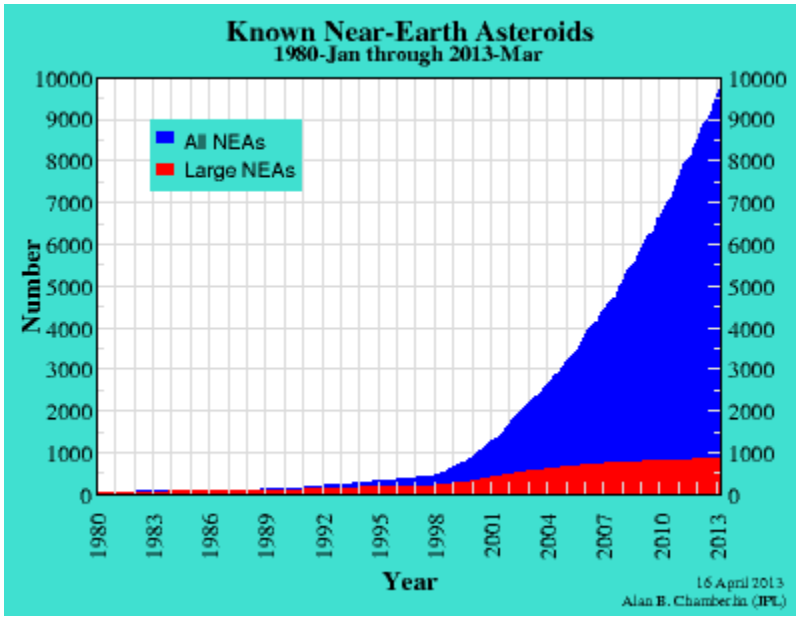

Figure 3: A plot showing the number of both near Earth asteroids, and large near Earth asteroids discovered per year.

A collision between Earth and a NEA with a diameter greater than 1 km would cause wide spread destruction and economic upheaval. If the diameter is greater than 2 km the impact would create a global catastrophic event that would cause earthquakes, tsunamis, and volcanic eruptions around the world. It could also produce huge dust clouds that would cause long term destructive and deadly effects. It was an event similar to this that killed 70% of all life on Earth 65 million years ago near the end of the Cretaceous period and created the 180 km across Chicxulub crater off of the Yucatán Peninsula. [4]

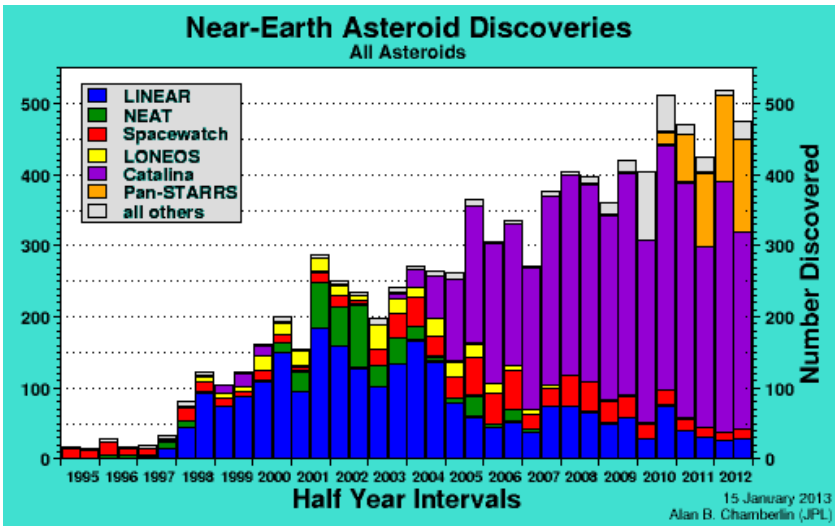

Figure 4: A figure showing near Earth asteroid discoveries.

It has been estimated that at least 300,000 asteroids larger than 30 meters across revolve around the Sun in orbits that cross Earth's orbit. Only about 3% of the estimated total number of NEAs have been discovered. Since only 10% of the sky has been mapped, the probability of discovering a large asteroid that is on a collision course with Earth is still quite high. With this, an effective strategy for preventing asteroid collisions with Earth must be devised. In 2008, the National Research Council asked scientists to find the best strategy to complete this goal. [6]

Introduction:

There are two proposed methods to mitigate an asteroid collision with Earth, destruction and deflection, and there are multiple techniques that could be used to achieve each. Deflection is the preferred method because destruction could cause multiple smaller asteroids to continue to be on a collision course with Earth. Unless the asteroid can be broken in to small enough pieces so that they pose no threat, destroying the asteroid is not a valid solution. Some techniques for the deflection of an asteroid include continuous force thrusters, and instantaneous force missile explosions. Both of these methods have their own flaws. Continuous force thrusters allow a lot of control over an asteroid's path. However, they require a long time to effectively change the asteroid's orbit. A missile explosion can change the path of an asteroid much more rapidly, but has the down side of its effect being harder to predict.

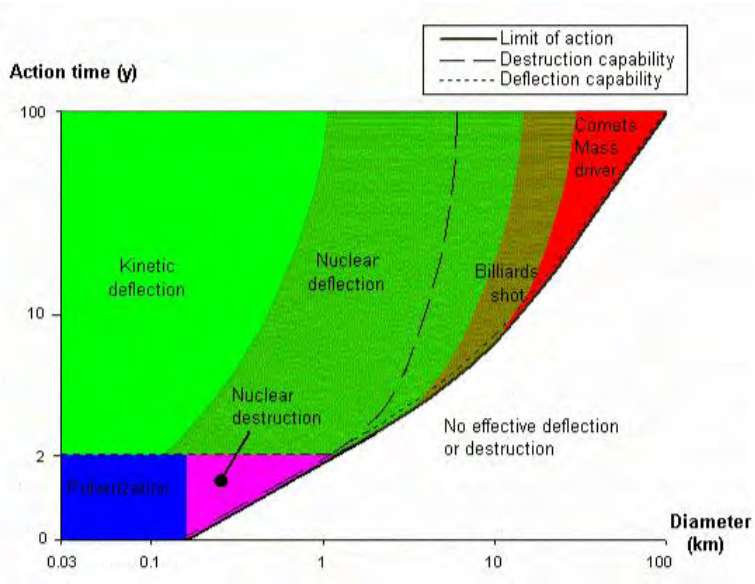

Figure 5: A chart describing mitigation strategies according to the size of the asteroid and the time of the collision

(Source: CSA 2005 Planetary Defense Report)

Focusing on the scenario of an asteroid being discovered with minimal time until impact, an instantaneous force explosion technique would have to be implemented in hopes of diverting the asteroid into a safe orbit. In order to better understand the orbital change that an asteroid would have due to an impact with a Delta Rocket (the largest rocket currently in operation), a zero-gravity simulation was created. As a qualitative study looking at the feasibility of modeling the rocket's ballistic collision with an asteroid underwater, the reaction immediately before, during, and after the impact was observed. Due to the limiting environment of underwater ballistics as being a practical experiment, the project was limited to only analyzing immediately before, during, and after impact, noting the significant drag effects from water. The asteroid was simulated underwater with a neutrally buoyant model. This asteroid model contained a waterproof data collection package which gathered data during impact. A spring-launched glass sphere, launched at the model asteroid, simulated the impulse caused by a Delta Rocket impact. Data was also collected by underwater cameras for comparison with the data collection package inside the asteroid model. If the data collected during the simulations correlates well with the theoretical calculations, then the test platform can be deemed a feasible way to model the instantaneous force a Delta Rocket exerts on an asteroid upon impact. Once feasibility is determined, the model can be rescaled and used to map the asteroid's reactions to different impulses at different angles through the use of a preexisting orbital dynamic simulation that has already been created. This knowledge can be used to predict how a ballistic impact will change an asteroid's orbit.

Theory

Experimental Theory

The goal of this underwater test platform is to observe the effect of a ballistic on an asteroid with different input forces and velocities. To do this, many equations need to be considered to determine the force of the ballistic exerted on the asteroid upon impact. Upon acquiring the object tracking software output, the following equations can be used to determine and relate the acceleration data.

In each experiment, the launched ballistic was shot from above the water, and penetrated the surface of the water in order to collide with the model asteroid. Fluid dynamics equations were used to find the force of the ballistic on the asteroid.

In order to find the initial launch force of the ballistic, the force from the spring had to be calculated using Hooke's Law,

$$F_k = kx \ , \tag{1}$$

where F_k is the force exerted on the ballistic from the spring, k is the stiffness of the spring in pounds per inch, and x is the distance that the spring was compressed in inches. In this experiment, the stiffness of the springs used were given by the manufacturer's specifications.

To calculate the velocity of the ballistic as it hits the water, equations to calculate the projectile motion of the ballistic are needed. In order to calculate the velocity of the ballistic, the velocity components in the X and Y axes could be calculated and then combined. In order to calculate the velocity of the ballistic in the X-direction at a specific position, the following equation was used,

$$V_x = V_0 \cos(\theta), \tag{2}$$

where V_x is the velocity of the ballistic at a certain position in the X-direction, V_0 is the initial velocity of the ballistic, and θ is the angle at which the ballistic is launched. In addition to this, the velocity of the ballistic in the Y-direction at a specific position is calculated with the equation below.

$$V_y = -gt + V_0 \sin(\theta) \tag{3}$$

Where t is the time after launch in seconds, and g is the acceleration due to gravity in ft/sec^2. With the ballistic being launched from above the water and hitting the asteroid underwater, an equation to calculate the surface tension of water must be considered.

$$\sigma_{SF} = \frac{F}{L} \tag{4}$$

In the equation above, σ_{SF} is the surface tension, F is the force along the fluid surface, and L is the length of the fluid surface, which is the diameter of the ballistic in this case. Lastly, the drag

acting upon the ballistic as it travels through the water has to be considered. The following equation can be used to calculate the drag force of a fluid on a sphere.

$$F_d = 6\pi\mu V d \tag{5}$$

In this drag equation, F_d is the drag force of the fluid on a sphere, μ is the absolute fluid viscosity, V is the velocity of the sphere relative to the fluid, and d is the diameter of the sphere.

To combine the previously mentioned equations, Newton's Second Law was applied.

$$\sum F = ma \tag{6}$$

Where F is the sum of the forces of the ballistic on the model asteroid, m is the mass of the model asteroid, and a is the changes in acceleration of the ballistic and the asteroid upon the collision. The accelerations were found in two different ways in order to compare results. With the output from the object tracking software, acceleration was found by taking the second derivative of the positional data acquired through video analysis. The second way the acceleration of the asteroid was found was through the data collection package within the model asteroid. The acceleration of the ballistic was found by simultaneously using the coefficient of restitution equation,

$$C_R = \frac{v_{p,1} - v_{a,1}}{v_{a,2} - v_{p,2}}\ , \tag{7}$$

and the conservation of momentum equation,

$$m_a v_{a,1} + m_p v_{p,1} + Impulse = m_a v_{a,2} + m_p v_{p,2} \quad . \tag{8}$$

In the equations above, C_R is the coefficient of restitution between glass and ABS plastic, approximately 0.65, v is velocity, m is mass, and the subscripts a and p denote asteroid and projectile respectively. The subscripts 1 and 2 denote the values immediately before and after the impact.

To calculate the overall density of the sphere, the following equation was used,

$$\rho_{assembly} = \frac{m_{assembly}}{V_{assembly}} \tag{9}$$

Where $\rho_{assembly}$ is the density of the assembly, $m_{assembly}$ is the mass of the sphere, electronics, and waterproof box combined, and $V_{assembly}$ is the total volume of the model asteroid.

Space Theory

There are many different factors to consider when impacting an actual Delta Rocket with an asteroid as opposed to impacting a ballistic at a model asteroid underwater. The following effects should be considered in space, opposed to in this underwater experiment.

One effect that will have an impact on how scientists will calculate launching a rocket at an asteroid is Yarkovsky's Effect. This effect is primarily due to the presence of the Sun. Due to

the Sun's heat, an asteroid experiences a very small force which can lead to long-term effects on the orbit and axial rotation of the asteroid.

In addition to Yarkovsky's Effect on the asteroid, another impact on objects in space is the gravitational pull from other large objects. For example, the planet Jupiter will have a major gravitational effect on an object in space. The two major bodies that would have an effect on an asteroid are the Sun and Jupiter. Every celestial body will have a slight gravitational pull, however, it may be small enough to neglect. The figure below shows the gravitational pulls of all of the planets, in addition to Pluto.

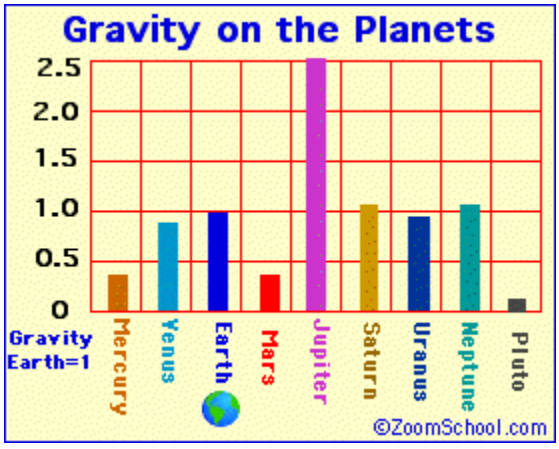

Figure 6: This figure shows the differences in gravitational pull for each planet and Pluto. It can be seen that Jupiter's gravitational pull is the most significant.

Additional effects to be taken into account are magnetic torque and ambient magnetic fields. One equation that should be considered in order to account for the magnetic disturbance torque on an object in space can be seen below,

$$T_m = M \; x \; B \tag{10}$$

where T_m is the magnetic disturbance torque, M is the effective dipole moment on the object in question, and B is the local magnetic induction, or flux density.

Solar winds should also be taken into account. A solar wind is a supersonic outflow of plasma into space from the Sun's corona, its most exterior layer. A solar wind is made up of an approximately equal amount of ions and electrons, and could affect the orbit of an object in space. The reason that a solar wind could affect an object in space is due to the fact that within the solar wind plasma, there is a weak magnetic field, which is also known as the interplanetary magnetic field. With each solar cycle, the properties of solar wind change, such as their temperature, velocity, etc. The average velocity of a solar wind is approximately 468 km per second.

Iterative Designs

Test Platform Iterations

Due to the fact that Asteroid Modeling and Prediction was a new project this year, there was no test platform already in place in order to carry out the experiment. Therefore, much of the beginning of this year (Fall Semester 2012) was spent brainstorming ideas on how to create a test bed to make this project feasible. Due to the fact that this was a very open ended problem, many options were analyzed.

When the project initially began, there were a few main ideas in place that were suggested by the project advisor, Professor May-Win Thein. Most importantly, this experiment was going to be conducted underwater to simulate the zero-gravity environment of space. It was soon realized that there were a few reasonable options where it would be possible to conduct the experiment; the wave tank and engineering tank in the Chase Ocean Engineering Building were among these options. In addition to this, the usage of a 3'x3'x5' glass, open-top tank was considered.

To accomplish the mission of hitting a model asteroid with different forces and directions, it was known that the test platform would have to surround the model asteroid to get a wide range of data. It was then decided that a rig would be created which very closely resembles a spinning globe holder which can be seen in the figure below.

Figure 7: A globe stand in which the globe inside can rotate

After many hours were put into the design of the rig, a SolidWorks model was created, which can be seen in the figure below.

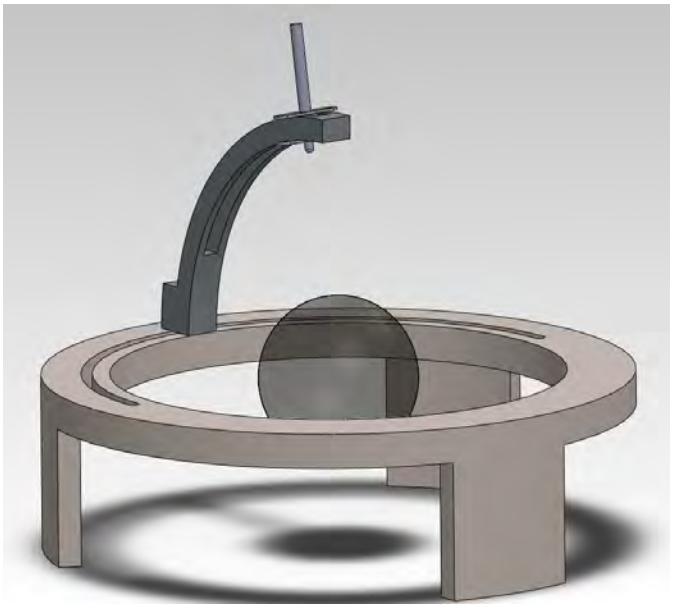

Figure 8: A SolidWorks model of the initial design of the rig to be used in the experiments

Launching Mechanism Iterations

The next stage of brainstorming involved figuring out what would be used for the launching mechanism and ballistic. The ideas for the launching mechanism ranged from using a coil gun, to compressed air, to a simple spring. Due to the fact that the rig design was already in place, it was decided that a compression spring would work best for the scale of the project. Once this was decided, there were multiple designs that were considered. The first of these designs involved a compression spring, a plunging mechanism, and a slot with a notch in the launching tube. In this design, the plunging mechanism is pulled back, both compressing the spring and pulling the ballistic back at the same time. The plunger would then be twisted into the notch of the slot, thereby holding the setup in place. When it was time to launch, the plunger would simply have to be turned out of the notch, and would allow decompression, launching the ballistic. The figures below depict this design.

Figure 9: The top view of the notched-slot launching mechanism idea. The black represents the launching tube, and the white represents the slot in which the plunging mechanism would travel.

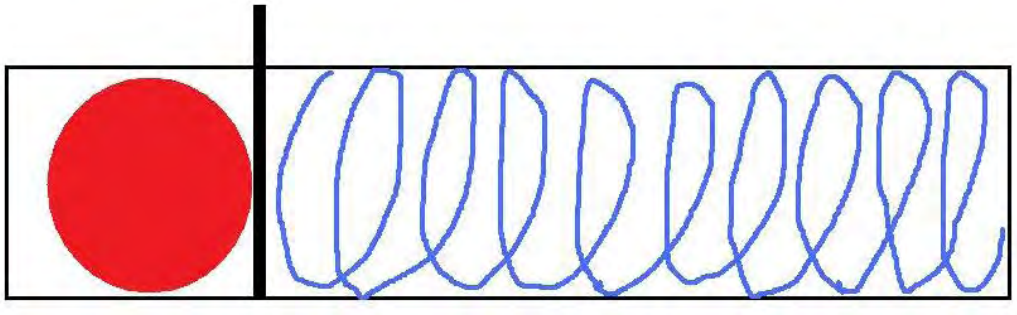

Figure 10: A side (and transparent) view of the notched-slot launching mechanism option. The blue represents the spring, the black represents the plunger, and the red ball represents the ballistic.

The notched-slot design was not chosen to be the launching device for this project. This was due to the fact that there was no reasonable way to hold the ballistic in place before launch. In addition to this, the plunging mechanism would have only allowed the spring to be compressed to one specific distance every time. This was not desired because to shoot the ballistic at different forces and velocities, multiple springs would have to be purchased with varying degrees of stiffness, and would cost more money than necessary.

The modified design of the spring launching system that was created fixed the problems that were brought up from the first design. As it can be seen in the figure below, this modified design had a pin which would hold the ballistic in place before launch. This pin would also serve as a trigger for the experiment. When the pin was pulled out and released, the ballistic would then be shot. Similar to the previously described design, this design also had a plunging mechanism to pull back and compress the spring to its desired stiffness.

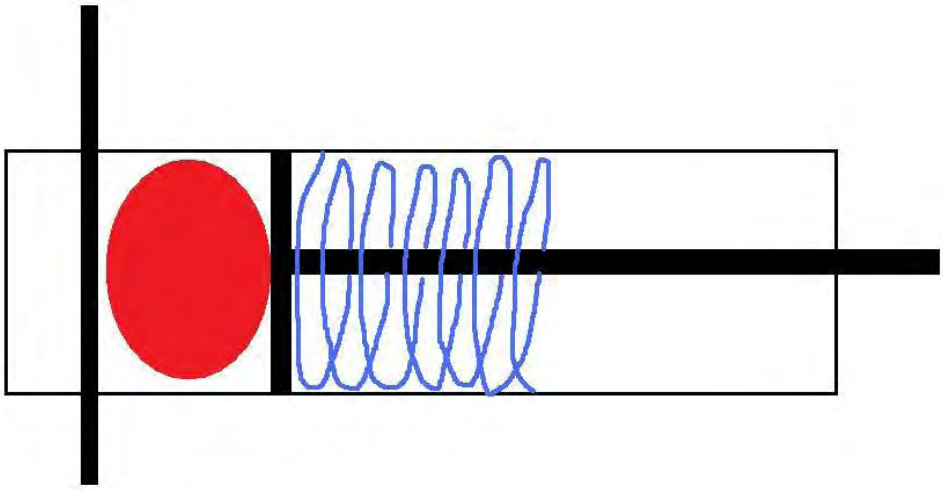

Figure 11: A side (and transparent) view of the modified launching mechanism. The long vertical black line represents the pin holding the ballistic in place.

The previously described design was modified one last time, and was finally the simplest and most convenient way to launch a ballistic at our model asteroid with different forces and velocities. As seen below, the final design resembles the second design in many ways. It includes a ballistic in a launching tube along with a compression spring. There is also a pin holding the ballistic in place. In this design, however, there were holes drilled through the launching tube, so that the pin could be inserted in any desired compression distance. Instead of needing a plunging system to pull the asteroid back, there was a removable end cap with the spring attached to it. The ballistic was then dropped into the launching tube, resting against the pin, and the end cap (along with the attached spring) was inserted, compressing the spring. This design proved to be the most convenient way to launch the ballistic at various forces using only one spring by simply inserting the pin at different distances along the launching tube.

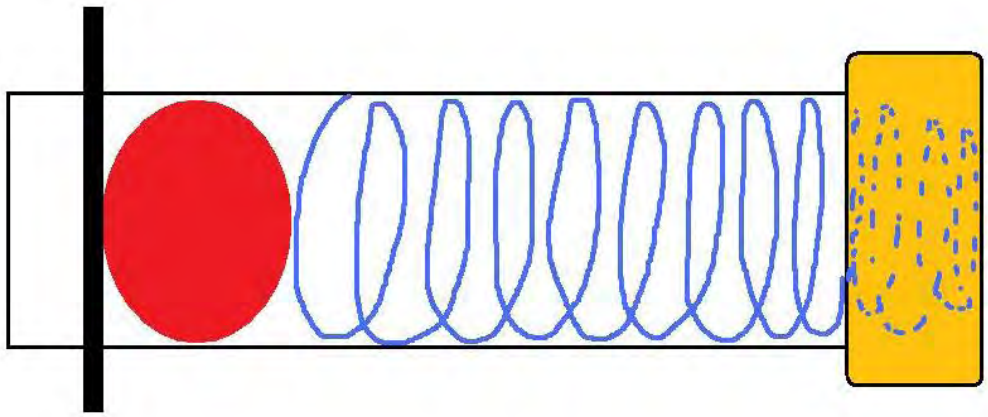

Figure 12: The final design of the spring launching system (the yellow represents the cap of the launching tube).

Model Asteroid Iterations

The following iteration is an excerpt from a technical report [17]

This design is made of aluminum. It has honeycombed internal aluminum grid with a space for the electronics. This design uses the proper thicknesses of aluminum in order to achieve neutral buoyancy while also making the center of gravity at the direct center of the sphere. The issues that came along with the design were the difficulty in creating this design and the high cost of manufacturing this model.

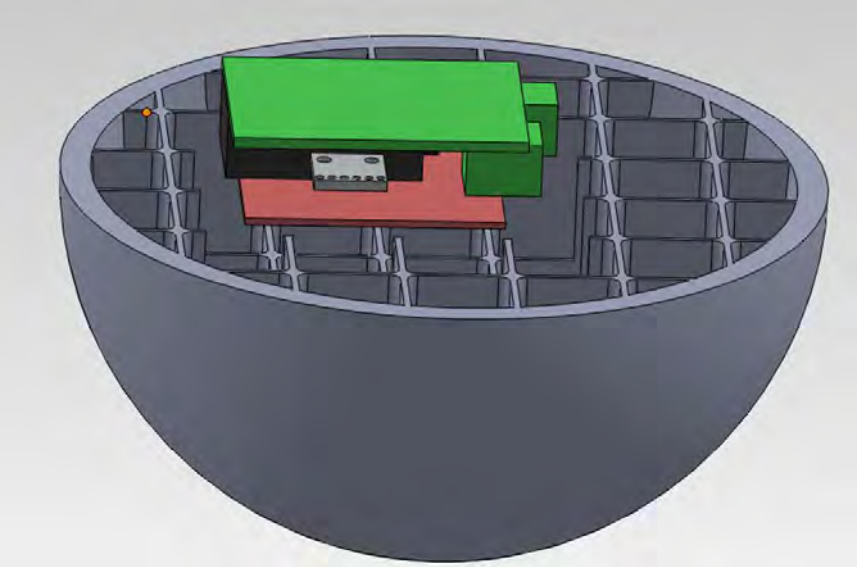

Figure 13: An alternative design for the asteroid model; an aluminum honeycombed sphere

This model is actually the better model for the project. However, upon viewing the final result of the sphere, it was determined that the model was overly complicated and would be very difficult and expexpensive to complete. Future groups trying to further the project should consider this design as a possible method to create a neutrally buoyant sphere that would also be a viable way to include gyroscopic information without bias.

Creating Model

The desired data collection sphere is to be 7" in diameter. The sphere is modeled as two identical halves using SolidWorks. Each half of the sphere is produced by revolving the sketch shown below about the axis shown in blue. The thickness of the shell (A) and radius of the sphere (B) are both linked dimensions (indicated by ∞) so that they can be easily referenced and changed throughout the optimization.

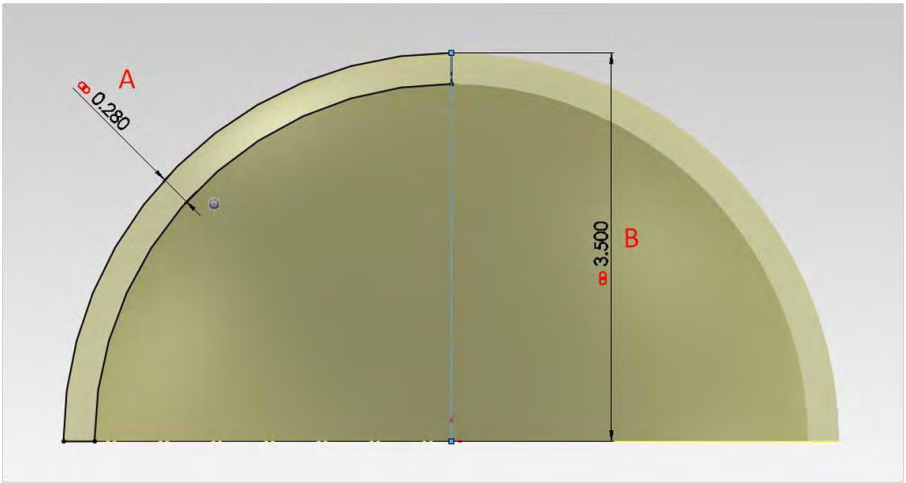

Figure 14: (Black) Revolved sketch about (Blue) axis, (Gold) Profile of created shape. (A) Shell thickness dimension, (B) Outer sphere radius. Dimensions in inches.

An internal grid outlined in Figure 15 is then added. grid dimensions 0.938 and 0.063 indicate 15/16" and 1/16" which are the width of the squares and web respectively. The grid also includes 1/10" fillets to allow for realistic machinability later in production. The grid, Figure 15 (Left), is produced by drawing the single square seen in Figure 15 (Right) and then creating a *Linear Sketch Pattern* to reproduce that sketch every 1 inch.

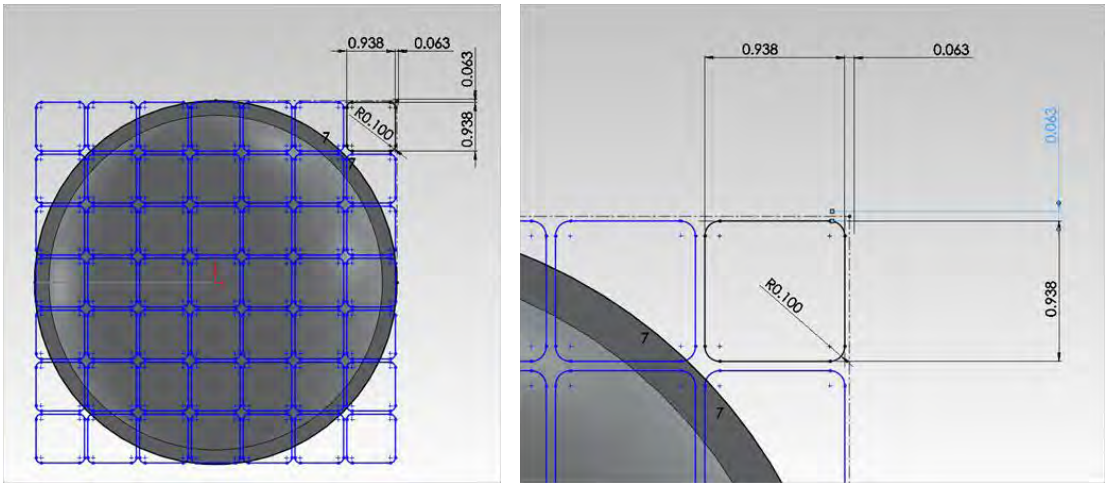

Figure 15: (Left) Internal structural grid added to data acquisition sphere (Right) Detail of dimensions, dimensions in inches.

The resulting grid is then *Boss Extruded* up to the inner surface of the hollow half-sphere, producing the structure shown in Figure 16.

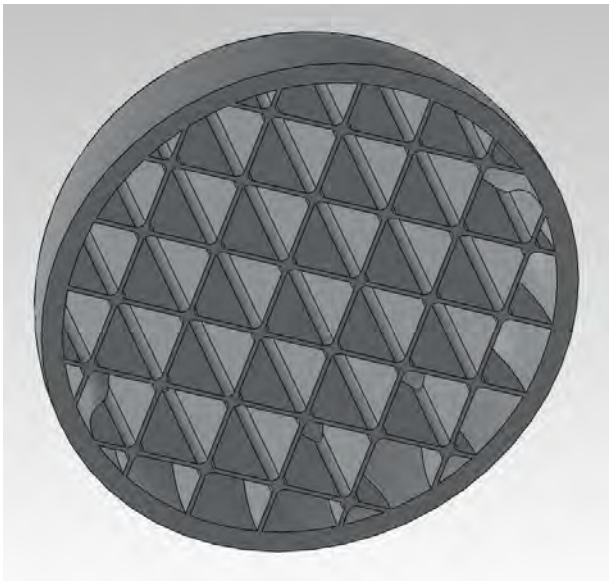

Figure 16: Hollow half-sphere with internal grid structure.

Two of these halves were then mated in an assembly such that their open faces were *Coincidently* mated to the *Top Plane,* the outer circumferences of their faces were *Concentrically* centered about the *Origin*, and the internal grids were mated to each other and the *Right Plane* using a *Parallel* mate to ensure that the assembly is symmetric about all 3 axes. The assembly is shown in Figure 17.

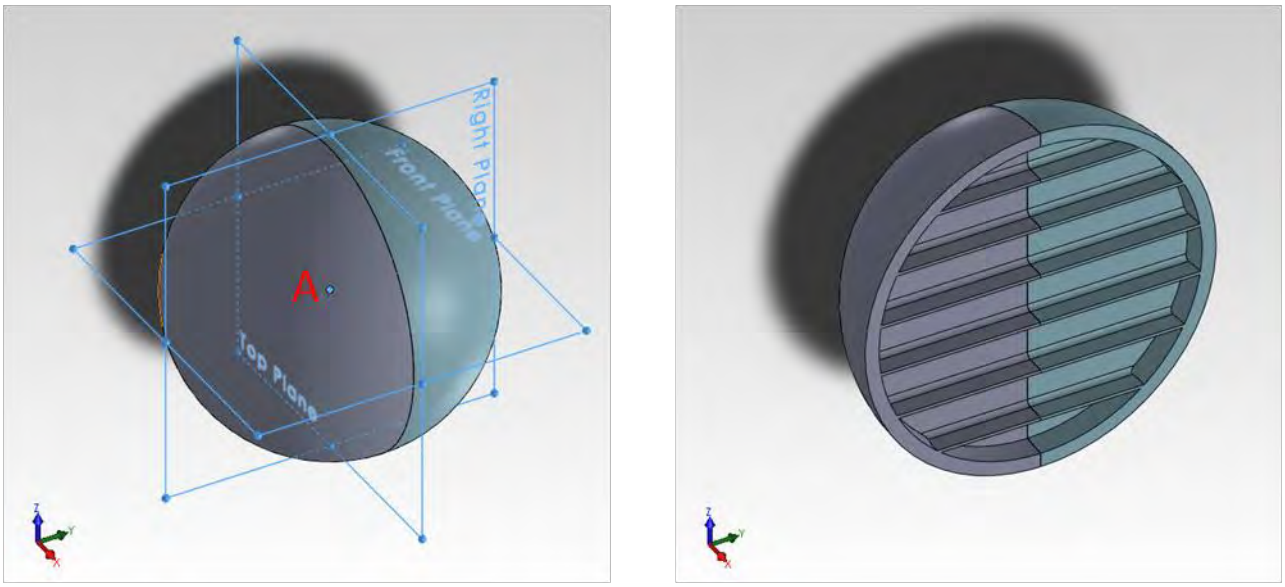

Figure 17: (Left)Two faces of data collection sphere mated together such that the origin (A) is at the center of the sphere.

(Right) Section view of internal structure to show alignment of grid.

To make a void for the data acquisition "black box", a sketch is placed on the top plane within the assembly and then extruded in both directions as seen in Figure 18. The location of the cut sketch can be changed by altering the values of "z_position" and "x_position", which are linked values relating the position of the cut to the outer circumference of the sphere.

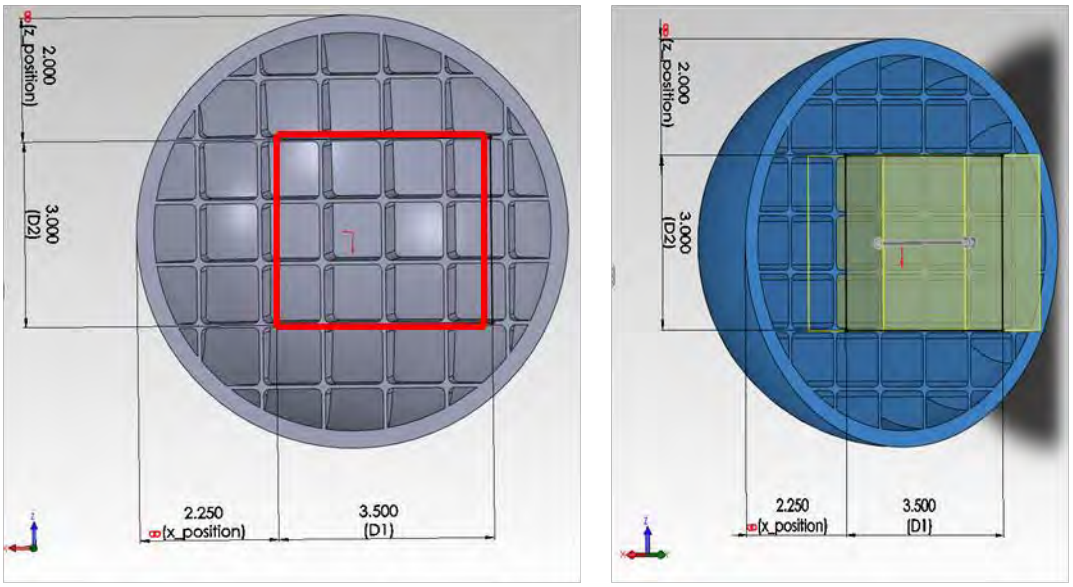

Figure 18: Location of cut for data acquisition "black box". Dimensions in inches.

The depth of the cut in the -Y direction was given the dimension name "depth" so that it could be easily referenced during optimizing. The depth of cut in the +Y direction was then driven by an assembly equation such that:

$$+Y_{depth} = t_{box} - \text{"}depth\text{"}\ , \tag{11}$$

where t_{box} is the thickness of the black box (2"). This equation ensures the void created for the box is always the same depth as the thickness of the box.

The data acquisition "black box" is then mated inside of the void using three *Coincident* surface mates. The mates anchor the back, top, and side of the black box to the respective faces within the cut.

The properties of the black box are detailed below in **Error! Reference source not found.**. The center of mass (C.O.M.) values were found when the box was centered at the origin.

Table 1: Black box properties

Height (Z-direction)	3"
Width (X-direction)	3.5"
Thickness (Y-direction)	2"
X-C.O.M.	-0.0289"
Y-C.O.M.	0.0672"
Z-C.O.M.	-0.0277"
Mass	1.146 lb

Mass Optimization Using SolidWorks

Because the design of the data collection sphere is performed outside of the Senior Project by a non-member, the overall design and material choices do not take into account the cost of materials or machining of the final product.

Aluminum and ABS plastic were both considered for the model optimization. The density of ABS plastic is 0.0368 lb/in^3, which is very close to the density of water, 0.0361 lb/in^3. This makes it a very poor material to optimize with because a solid sphere is nearly neutrally buoyant, even before the data collection "black box" is taken into account. For this reason the SolidWorks model is created from Aluminum AISI 1060 which has a density of 0.0975 lb/in^3.

The goal of the mass optimization was to have the assembly be neutrally buoyant in water. The desired mass can be determined from Equation 12:

$$m_{assebly} = \rho_{water} * \frac{4}{3}\pi R^3 \tag{12}$$

The desired mass was determined to be 6.482lb.

The optimization was then performed by varying the thickness of the shell (A in Figure 14) from 0.01" to 0.5" in increments of 0.01". The cut for the black box was centered on the face of each of the half-spheres, and then kept there for the entire optimization. The results for the mass optimization can be seen in Figure 19. The Mass3 values are the mass of the entire assembly including the black box, and Parameter1 is the shell thickness.

		Current	Initial	Optimal (28)	Scenario 1	Scen
Parameter1		0.01in	0.01in	0.28in	0.01in	0.02in
Center of Mass X2	Is exactly 0	-0.0103in	-0.0103in	-0.00509in	-0.0103in	-0.009
Center of Mass Y2	Is exactly 0	0.02393in	0.02393in	0.01183in	0.02393in	0.023ir
Center of Mass Z2	Is exactly 0	-0.00985in	-0.00985in	-0.00487in	-0.00985in	-0.009
Mass3	Is exactly 2.94	3.22001 lb	3.22001 lb	6.51529 lb	3.22001 lb	3.3498

Figure 19: Optimization results field for mass optimization. Parameter1 is the shell thickness.

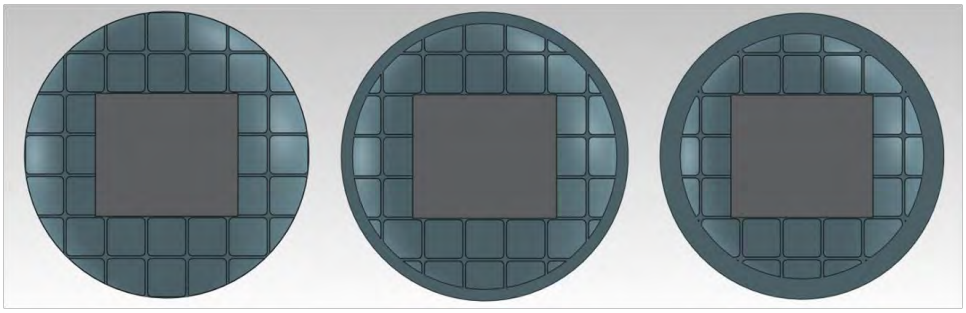

Figure 20: (Left) Min. shell thickness 0.01" (Center) Optimized shell thickness 0.28" (Right) Max shell thickness 0.5"

The overall density of the assembly can be determined from

$$\rho_{assembly} = \frac{m_{assembly}}{V_{assembly}}, \tag{13}$$

where the effective volume of the assembly is the volume of the water displaced by the sphere.

$V_{assembly}$ is given by

$$V_{assembly} = \frac{4}{3}\pi(3.5")^3 \tag{14}$$

The effective volume of the assembly is 179.59 in^3. The results of the optimization can be seen below in Table 2.

Table 2: Mass optimization results based on varying shell thickness

	Thickness [in]	Mass [lb]	Density [lb/in^3]	% Difference
Desired	**Unknown**	**6.482**	**0.0361**	**0%**
Thin Shell	**0.01**	**3.220**	**0.0179**	**67.4%**
Thick Shell	**0.5**	**8.798**	**0.0490**	**30.3%**
Optimized Shell	**0.28**	**6.515**	**0.0363**	**0.6%**

The best result from the optimization is a shell thickness of 0.28 inches, which produces a sphere who's density is 0.6% greater than that of water.

Center of Mass Optimization Using SolidWorks

For data collection t is important that the center of mass for the sphere is located at its origin. This ensures that forces normal to the surface won't produce rotational effects.

The center of mass optimization is performed by varying the "z_position", "x_position", and "depth" dimensions. The shell thickness used is 0.28".

Table 3: Range and step of iterations for center of mass optimization

	Min [in]	Max [in]	Step [in]
Z_position	**1.5**	**2.5**	**0.25**
X_position	**1.25**	**2.25**	**0.25**
depth	**0.5**	**1.5**	**0.25**

Each of the dimensions is stepped in increments of 0.25 inches, which is a fairly coarse step size, but even so there are 150 iterations in the optimization. Decreasing the magnitude of the step size produces an error due to too many iterations. The results of this optimization can be seen below in Figure 21.

		Current	Initial	Optimal (63)	Scenario 1	Scen
z_position		2in	2in	2in	1.5in	1.75in
x_position		1.75in	1.75in	1.75in	1.25in	1.25in
depth		1in	1in	1in	0.5in	0.5in
Center of Mass X2	Is exactly 0	-0.00509in	-0.00509in	-0.00509in	0.06351in	0.0636i
Center of Mass Y2	Is exactly 0	0.01183in	0.01183in	0.01183in	-0.05684in	-0.0569
Center of Mass Z2	Is exactly 0	-0.00487in	-0.00487in	-0.00487in	0.06374in	0.0242

Figure 21: Optimization results for center of mass. Center of Mass X2, Y2, and Z2 are the components of the center of mass with respect to the origin.

The results from the optimization indicate that the best location for the black box is centered at the origin as seen in Figure 22. This isn't surprising due to the large step size and the small asymmetries in the black box's center of mass.

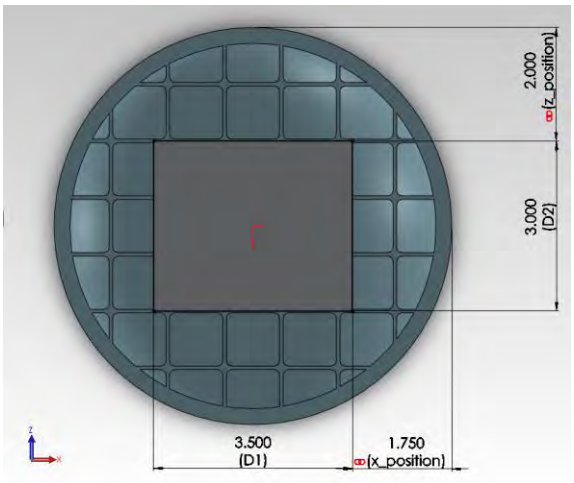

Figure 22: Optimized location of black box. Units are inches.

Overall Optimization Using SolidWorks

SolidWorks limits the number of iterations in an optimization, so shell thickness and center of mass were optimized separately. The results from these optimizations provide a good starting point for the final optimization which will encompass both the mass and center of mass of the assembly using a finer step. The range and step size of the parameters can be found below in Table 4, and the results in Figure 23.

Table 4: Results from previous optimization as well as range and step for final optimization.

	Results from Opt.	Min	Max	Step
Z_position	**2"**	**1.9"**	**2.1"**	**0.05"**
X_position	**1.75"**	**1.7"**	**1.8"**	**0.05"**
Depth	**1"**	**0.9"**	**1.1"**	**0.05"**
Shell Thickness	**0.28"**	**0.27"**	**0.29"**	**0.005"**

		Current	Initial	Optimal	Scenario 1	Scena
z_position		2in	2in	1.9in	1.9in	1.95in
x_position		1.75in	1.75in	1.7in	1.7in	1.7in
depth		1in	1in	0.9in	0.9in	0.9in
Parameter1		0.28in	0.28in	0.28in	0.27in	0.27in
Center of Mass X2	Is exactly 0	-0.00509in	-0.00509in	0.00272in	0.00274in	0.00274
Center of Mass Y2	Is exactly 0	0.01183in	0.01183in	-0.00138in	0.00139in	0.0014i
Center of Mass Z2	Is exactly 0	-0.00487in	-0.00487in	-0.00015in	0.00016in	0.00256
Mass3	Is exactly 2.94	6.51529 lb	6.51529 lb	6.51529 lb	6.56009 lb	6.56009

Figure 23: Second optimization results for mass and center of mass.

Table 5: Final optimization results for mass and center of mass.

	Desired	1st Optimization	2nd Optimization	%Improvement
X-C.O.M.	**0"**	**-0.00509"**	**0.00272"**	**145.9**
Y-C.O.M.	**0"**	**0.0118"**	**-0.00138"**	**755.1**
Z-C.O.M.	**0"**	**0.00487"**	**-0.00015"**	**3146.7**
Mass	**6.482 lb**	**6.515 lb**	**6.515 lb**	**0**
Density	**0.0361 lb/in^3**	**0.0363 lb/in^3**	**0.0363 lb/in^3**	**0**

The second optimization produces improved results for the center of mass without changing the overall mass of the assembly. The percent improvement was calculated as an error between the magnitude of the first and second optimizations using Equation 15.

$$\%Improvment = \frac{|Opt_2|-|Opt_1|}{|Opt_1|} * 100 \quad (15)$$

The optimized model can be seen in Figure 24, with the corresponding values in Figure 23.

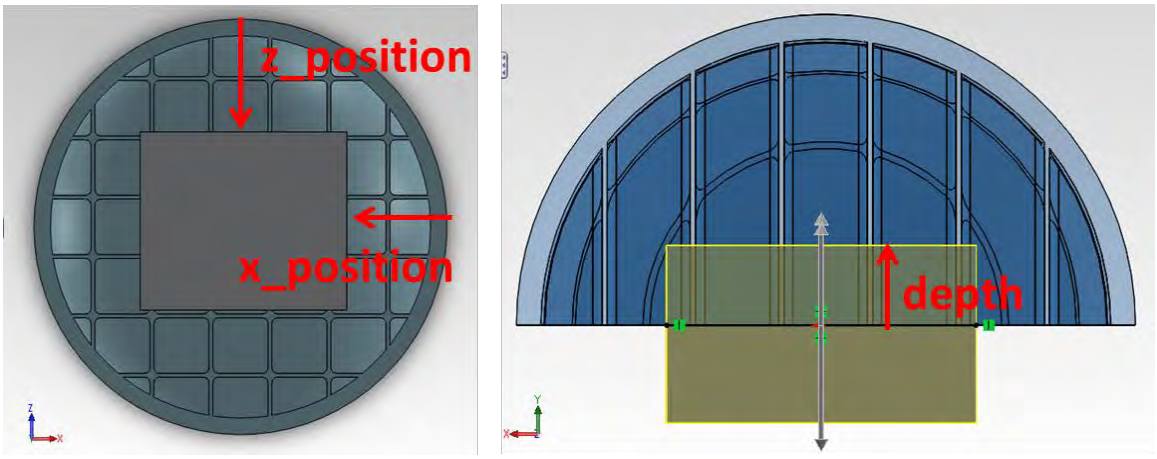

Figure 24: Final optimized model. Dimensions of z_position, x_position, and depth can be taken from Figure 23 (Optimal).

Verifying Mass Optimization Results

The mass of the assembly can be verified by breaking the complex geometry of ball into pieces, determining their respective volumes, and then summing them together to determine the final volume of the assembly.

The volume of the outer shell can be calculated using Equation 16:

$$Vol_{shell} = \frac{4}{3}\pi(R^3 - r_{shell}^3) \tag{16}$$

The internal grid of the sphere can be thought of as two sets of pancake-like cylinders that intersect each other at right angles. These cylinders intersect forming "pillars", which are the square columns formed by the intersections. The length of these pillars (Y in Figure 25) can be found using Equation 17.

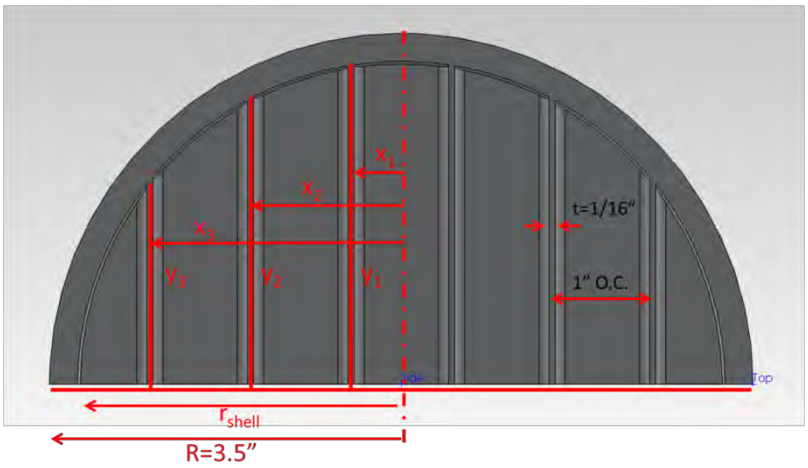

Figure 25: Breaking down grid into circles with radii (y) that vary based on distance from central axis (x)

$$Y_n = \sqrt{r_{shell}^2 + X_n^2 + Z_n^2} \tag{17}$$

X and Z are vectors ranging from -3.5" to 3.5" in 1" increments as defined by the grid in Figure 15. The value of r_{shell} was determined to be 3.22" during the mass optimization.

The following matrix is constructed of the values of Y in the X-direction (left to right), and Z-direction (up to down), for the pillars. Because the matrix is symmetrical, these directions become arbitrary.

0	0	0	0	0	0	0	0
0	0	1.37	1.97	1.97	1.37	0	0
0	1.37	2.42	2.81	2.81	2.42	1.37	0
0	1.97	2.81	3.14	3.14	2.81	1.97	0
0	1.97	2.81	3.14	3.14	2.81	1.97	0
0	1.37	2.42	2.81	2.81	2.42	1.37	0
0	0	1.37	1.97	1.97	1.37	0	0
0	0	0	0	0	0	0	0

Because the pillars don't ever fall along the diameter of the half-spheres, as is apparent by the 8x8 matrix which has no central row or column, a set of "ghost pillars" must be constructed between these pillars to determine the radii. The radii of the cylinders fall down the central axes of the matrix below, shown highlighted in red. It can be observed that the central value is equal to the outer radii of the sphere (3.5") minus the thickness of the shell (0.28"), or "r_{shell}".

0	0	0.61	1.17	0.61	0	0
0	1.54	2.32	2.52	2.32	1.54	0
0.61	2.32	2.89	3.06	2.89	2.32	0.61
1.17	2.52	3.06	3.22	3.06	2.52	1.17
0.61	2.32	2.89	3.06	2.89	2.32	0.61
0	1.54	2.32	2.52	2.32	1.54	0
0	0	0.61	1.17	0.61	0	0

The total volume of all of the cylinders within the grid can then be expressed by

$$Vol_{cylinders} = \pi t \sum_{i=1}^{7} (y_{4,i}^2 + y_{i,4}^2), \tag{18}$$

where t is 1/16", the thickness of the web, and the y values are taken from the second matrix.

Because the cylinders overlap, the pillars produce redundancies that get counted twice. The volume of the redundant pillars can be found from the first matrix using Equation 19:

$$Vol_{redundant} = 2t^2 \sum_{i=1}^{8} \sum_{k=1}^{8} y_{i,k} \quad (19)$$

Where the coefficient 2 accounts for both halves of the sphere, and t^2 is the cross sectional area of the pillars. A similar equation can be used to account for the volume of the fillets:

$$Vol_{fillets} = 2[(2r_{fillet})^2 - (\pi r_{fillet}^2)] \sum_{i=1}^{8} \sum_{k=1}^{8} y_{i,k}, \quad (20)$$

where r_{fillet} is 0.1", the radius of the fillets. The bracketed term before the summation represents the cross sectional area of the fillets, which is explained in Figure 26.

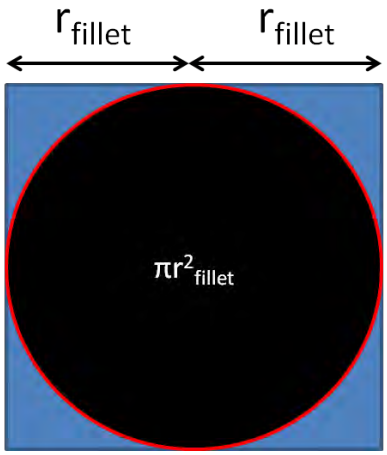

Figure 26: (Blue) Cross sectional area of fillets

The portion of the grid which is removed to make room for the black box can be found using Equation 21:

$$\boldsymbol{Vol_{box} = N_x(h_{box} * t_{box}) + N_z(w_{box} * t_{box}) - N_{pil}(t_{box} * t_{web}^2) + N_{pil}[(2r_{fillet})^2 - (\pi r_{fillet}^2)]} \quad (21)$$

where N_x and N_z are the number of webs the box interrupts in the X and Z directions respectively. N_{pil} is the number of pillars the box interrupts. The values of h_{box}, w_{box}, and t_{box}, represent the height (Z), width (X), and thickness (Y) of the box respectively as seen in Figure 22.

The total volume of the assembly can then be expressed by Equation 22:

$$Vol_{total} = Vol_{shell} + Vol_{cylinders} - Vol_{redundant} + Vol_{fillets} - Vol_{box} \quad (22)$$

And the total mass of the assembly can be expressed by Equation 23:

$$Mass_{total} = [Vol_{total} * \rho_{Al}] + Mass_{box} \quad (23)$$

where ρ_{Al} is 0.0975 lb/in^3, the density of Aluminum AISI 1060 .

The results from the hand calculations with comparisons to the SolidWorks optimization can be seen in **Error! Reference source not found.**.

Table 6: Comparison of mass result from SolidWorks model and hand calculations.

	Mass [lb]	Density [lb/in^3]
Water	6.482	0.0361
SolidWorks Model	6.515	0.0363
Hand Calculations	6.529	0.0364
% Difference	1.53%	0.28%

The percent difference is calculated from Equation 24, and the density is calculated from Equation 13.

$$\%dif = \frac{val_1 + val_2}{avg(val_1, val_2)} * 100 \tag{24}$$

The hand calculation values fall within a few percent difference of the SolidWorks model, which corroborates the mass values determined from the optimization found in **Error! Reference source not found.**.

Verifying Center of Mass Optimization

The center of mass found from the SolidWorks optimization can be verified using simple hand calculations. The effective center of mass for two bodies can be broken into its X Y and Z components. The equation for the X component is given as Equation 25.

$$x_{cm} = \frac{m_1 x_1 + m_2 x_2}{m_1 + m_2} \tag{25}$$

The entire shell, and the majority of the grid are symmetric about the origin, so their mass and center of mass can be considered m_1.The asymmetrical core of the assembly can be considered m_2. In order to isolate m_2 symmetry conditions were applied.

Within the assembly a set of mirrored planes was constructed equidistant from each of the three major planes of the assembly. One plane of each pair was made coincident with the farthest face of the black box, as with the uppermost plane in Figure 27. An extruded cut was then made away from the origin on each of the six planes, removing all symmetrical material from the assembly. An example of the plane orientation can be seen in Figure 27 and the final results can be seen in Figure 28.

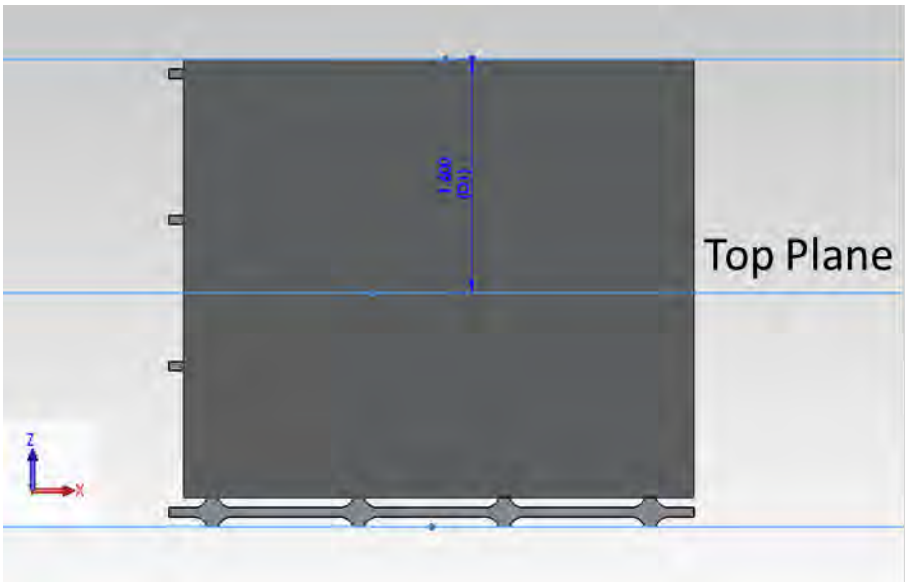

Figure 27: Example of symmetrical plane cuts applied to reduce volume of assembly.

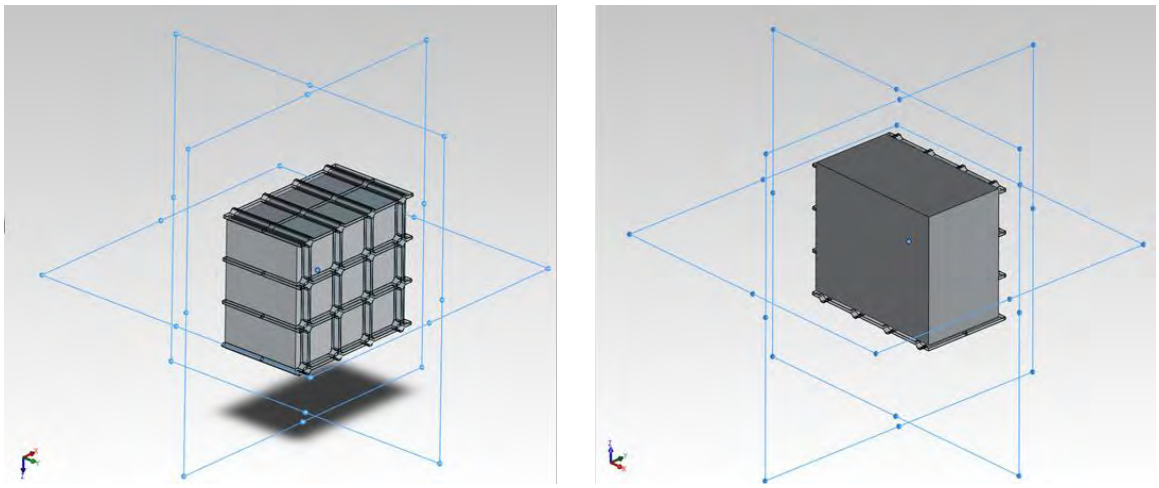

Figure 28: Final assembly after all symmetry conditions have been applied.

Subsequently this new m_2 can be broken down into two masses: one for the box, whose center of mass is outlined in **Error! Reference source not found.**, and the remaining portion of the grid shown in Figure 29.

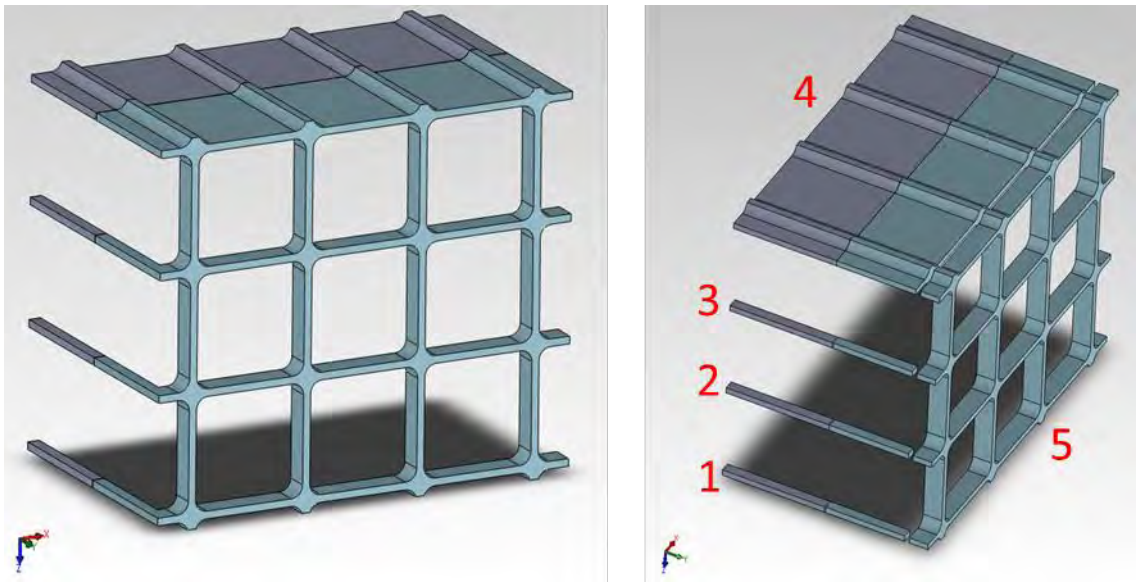

Figure 29: (Left) Asymmetric portion of grid. (Right) Broken into pieces to be analyzed individually.

By breaking the asymmetric grid into the five symmetric pieces in Figure 29 (Right), each piece's center of mass is at its geometric center. Because the geometries of these pieces are rather complicated, it would be very time consuming to calculate the masses of and center of mass for each component. Instead these values were determined from SolidWorks. Each part was analyzed by suppressing the other four components, and then observing the mass and center of mass of the assembly (which is still centered at the origin).

Table 7: Mass and center of mass values for each piece in the assembly. "Part 1, Part 2…" refer to **Error! Reference source not found.**. "Symmetric Sphere" is the portion of the assembly that was suppressed when applying symmetry conditions. "SW Assembly" is the completed SolidWorks assembly after the optimization was performed.

	Mass [lb]	X-C.O.M. [in]	Y-C.O.M. [in]	Z-C.O.M. [in]
Part 1	**0.0012**	**-1.75**	**-0.1244**	**1.5**
Part 2	**0.0012**	**-1.75**	**-0.1244**	**0.5**
Part 3	**0.0012**	**-1.75**	**-0.1244**	**-0.5**
Part 4	**0.0557**	**0**	**-0.1244**	**-1.5**
Part 5	**0.0346**	**0**	**1.00**	**0**
Symmetric Sphere	**5.2739**	**0**	**0**	**0**
Black Box	**1.146**	**0.0211**	**-0.0328**	**0.0723**
SW Assembly	**6.5153**	**0.0027**	**-0.0014**	**-0.0002**

Using Equation 25 and its respective equivalencies for Y and Z it is possible to determine the collective center of mass for Parts 1-5, the Symmetric Sphere and the Black Box. The results can be seen in **Error! Reference source not found.**.

Table 8: Results of center of mass calculations and comparisons made to SolidWorks optimized assembly.

	Mass [lb]	X-C.O.M. [in]	Y-C.O.M. [in]	Z-C.O.M. [in]
Collective C.O.M.	6.5138	0.0027	-0.0016	0.00017
SW Assembly	6.5153	0.0027	-0.0014	-0.0002
% Difference	0.02%	0%	13.3%	2467%

The percent differences for the mass and X and Y component for the center of mass all fall within a reasonable percent difference with the SolidWorks model. The Z component is a huge percent different, but the magnitude of the Z component is so small that even a small error can result in a large % difference. Overall the hand calculations do corroborate the SolidWorks results.

Rig Design

Once the team's initial designs for the rig and the launching mechanism were created, with the help of Paul Lavoie, the design work for the rig started. To start the design, it was determined that that entire rig would be made of a material that would not rust underwater. For the base ring and the overhead arches of the rig, the material was chosen to be 6061 aluminum, not only for its resistance to rust, but also because of its low cost. Due to the fact that it was already determined that the outer diameter of the model asteroid was to be 8 inches, it was known that the inner diameter of the base ring should be larger than that. The figure below shows the final design of the rig.

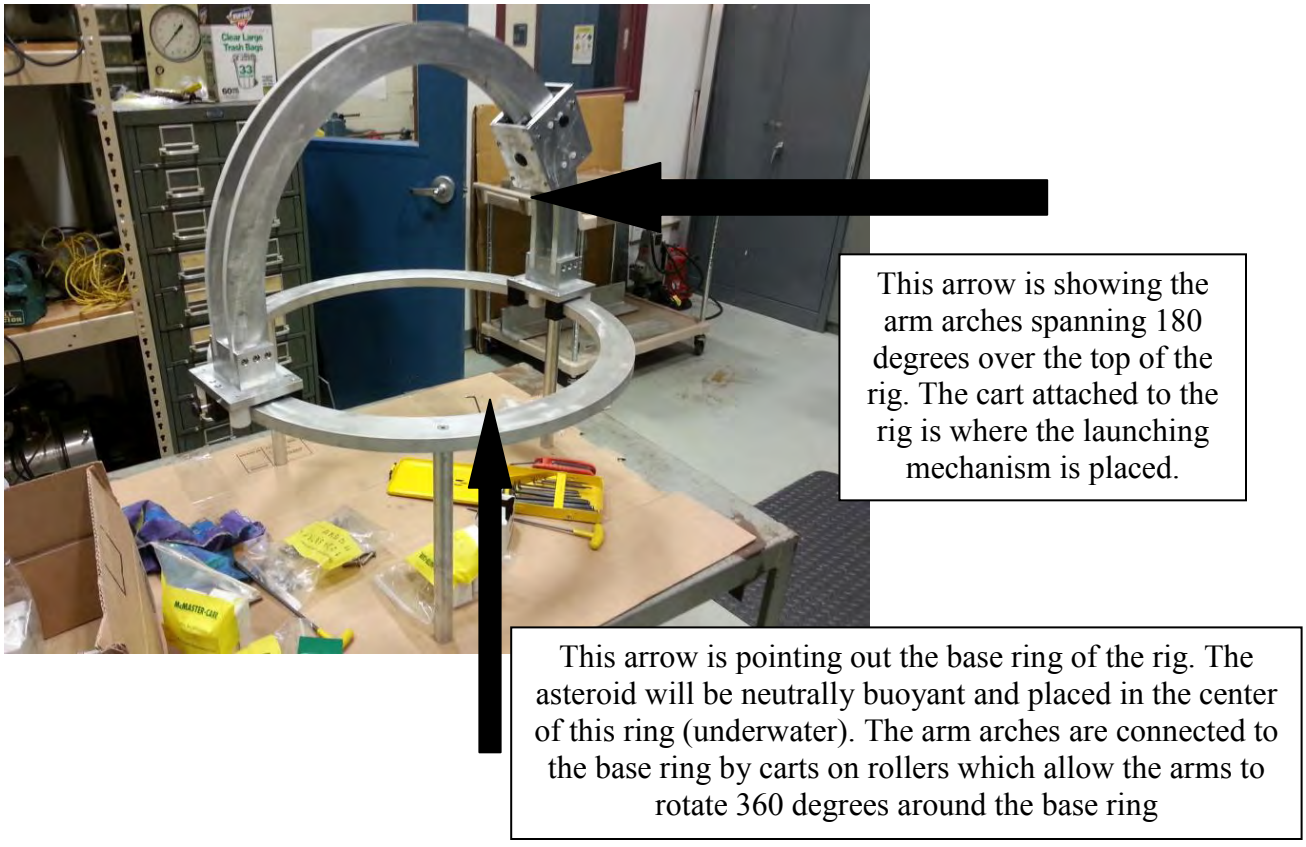

Figure 30: This is a picture of the finished rig, with descriptions and arrows pointing to the major design aspects.

After measuring the inside of the test tank and taking into account extra space for the cameras and launching mechanism, it was determined that the outer diameter of the base ring would be 26 inches. The arms were also chosen to have the same diameter, (the half arch having an outer radius of 13 inches. A sheet of ¾" thick 6061 aluminum was purchased to machine the base ring, and a sheet of ½" thick 6061 aluminum was purchased to machine both arm arches, and the three carts that the launching mechanism and the arms travel on. Additionally, both the base ring and the arms have a width of 2 inches. The figure below shows the design for the traveling carts of the rig.

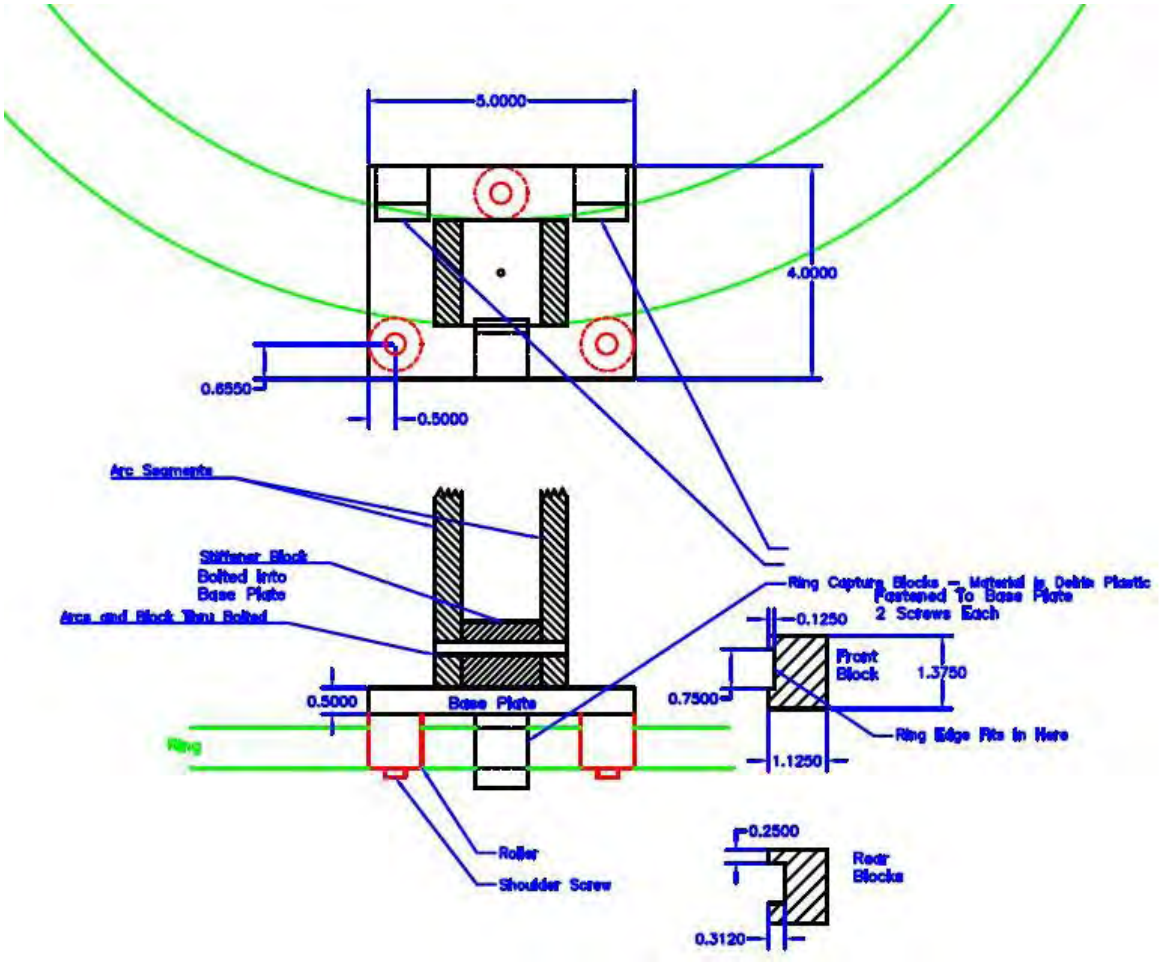

Figure 31: The design of the rig, created in Autocad by Paul Lavoie.

The top portion of the figure above shows a top view of one of the carts that would connect the arms to the base ring. Both of the carts travel around the base ring with three Delrin plastic rollers. In addition to the rollers, there is a spacer in between the cart and the base ring, keeping a quarter of an inch of space between the two. The purpose of this spacer is to reduce the amount of drag and scraping between the cart and the base ring.

To attach the arm arches to the carts on the base ring, a stiffness block with a width of 1.5 inches was placed in between the arms, resting on the cart, as can be seen in the figure above. The block was then screwed into the cart, and the arms were screwed into the stiffness block. This stiffness block has two purposes in the rig design. The first is to keep an even spacing between the arms on both sides of the arch. The second reason the stiffness block was used was due to the fact that the arms were both only a half inch thick, and would have been difficult to screw into the cart plate.

Launching Mechanism

The launching tube is made of PVC pipe and was cut to a foot long. Holes were drilled at half inch intervals in the pipe for pin holes in order to be able to exert various forces on the ballistic. A compression spring with a stiffness of approximately 3.4 lb/in was used. A glass marble with a diameter of approximately 0.73 inches was used as the ballistic in this experiment. A pin with a 1/8” diameter was used in the pin holes in order to hold the ballistic in place. The setup of this design can be seen above in Figure 12.

For each experiment, the launching mechanism was set to the desired spring compression, and the launching system was then inserted into a hole, fixed on the cart along the arm arches. The cart was then placed at the desired angle along the arch.

Model Asteroid

Initial Goals for Design:

The first criterion for creating a model asteroid and simulating a zero-gravity atmosphere was to build a sphere capable of being submerged into water and remaining neutrally buoyant. The sphere must also be capable of withstanding an impact from the glass projectile mentioned above. The idea behind this project is to be able to recreate any simulation time and time again. If the sphere dents or scratches at all upon impact, the longevity of this project could be compromised. Lastly, the sphere needs to be able to safely house the Arduino Uno, accelerometer and other electronics responsible for recording the resulting accelerations due to the impact.

A sphere was used in the project due to its simple geometry. Being underwater, the sphere will be subject to many different aspects not seen in space. The resulting accelerations caused by the glass projectile impacting the asteroid model must be modified to more accurately represent a collision made in space. Any movement the sphere undergoes will be effected by drag through water. The impact between the glass and the material used for the asteroid will require a different coefficient of restitution than that of a delta rocket and, for example, a granite asteroid such as Apophis.

The chosen design for the model asteroid was the most simple design of any iteration. The sphere was made out of ABS plastic and was rapid prototyped. The sphere consisted of two

separate hemispheres that screwed together by the use of threading. On the inner pole of each hemisphere was a pedestal. A waterproof box, described at the end of this section, was glued to one hemisphere and enclosed within the two pedestals when the halves were screwed together. The waterproof box was the housing for the data acquisition electronics.

Each of the two hemispheres was created using similar methods in SolidWorks until the steps where threading was created. The first step was to create an arch. An 8 inch line was drawn that was connected into an arch using the circle feature in SolidWorks. The arch was then revolved 180 degrees around the straight line in order to create a solid hemisphere. Both of these steps can be seen in Figure 32 left and right, respectively.

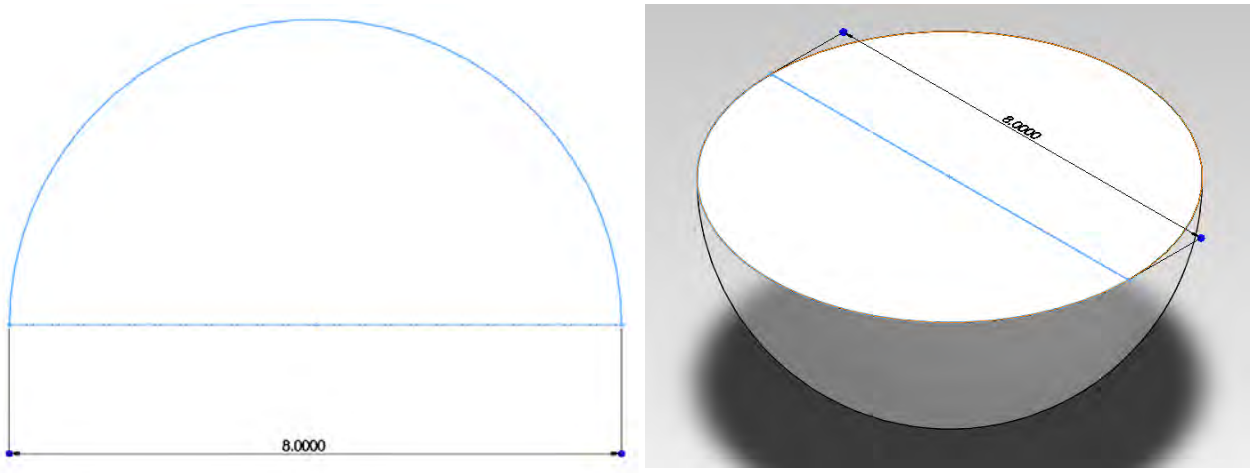

Figure 32: (Left) This arch shows the first step toward making a solid sphere.

(Right) This is the solid model of a hemisphere which both the male and female parts used as a base for building

The shell feature was then used to create a 0.25 inch wall thickness for the hemisphere. A Boss-Extrude feature was used in order to create a circular bottom base in which to sketch from.

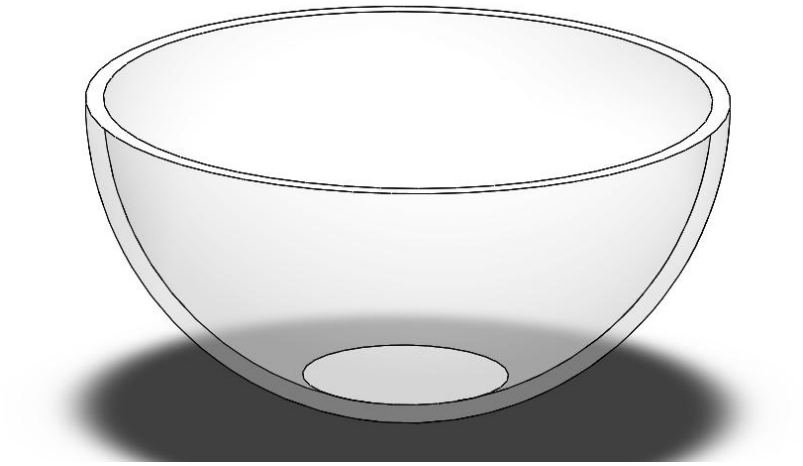

Figure 33: The hemisphere with a 0.25 inch shell and a flat sketching surface built at the inner pole

Next, for each hemisphere, four 0.4 inch circles were created on the new sketch platform at the pole of the sphere. These circles are made equidistant from the center of the sketch plane and equidistant from each other, within a 1.75 inch diameter. The Boss-Extrude feature was used to extrude the circles 0.46 inches. Using the tops of the four circles as a plane, a circle of diameter 1.85 inches was created and extruded 0.2 inches. These distances were chosen because when the two hemispheres are screwed together, the vertical distance between the tops of each pedestal is 5.89 inches, the vertical height of the waterproof box. Filets of 0.2 inches are applied to the connections from the legs to the sketch platform and the top of the pedestal. The hemispheres completed to this step can be seen below in Figure 34 .

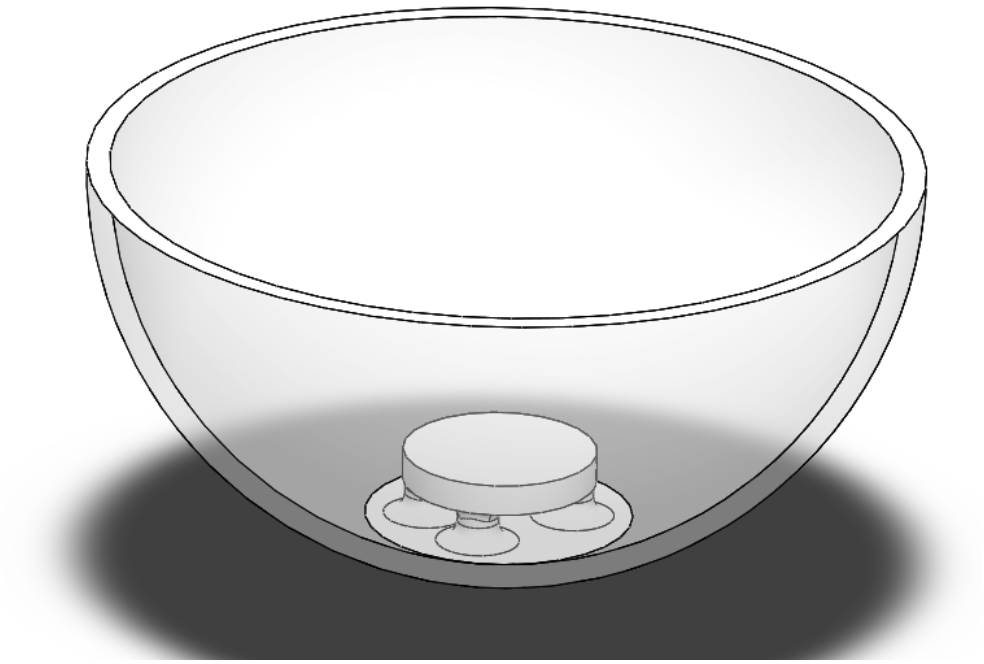

Figure 34: The hemisphere of the asteroid model completed to the step where both male and female models overlap.

Male only steps:

Using the flat top part of the rim as a sketch plane, a circle 0.5 inches from the inner lip was created and extruded downward toward the pedestal using Up To Next as a direction. This will create a sturdy platform from which to build off of to create the male section of the threading. This step can be seen below in Figure 35 (left and right) is the model after a 1.0 inch extrusion is made from a circle sketch .225 inches from the inner ring of the lip.

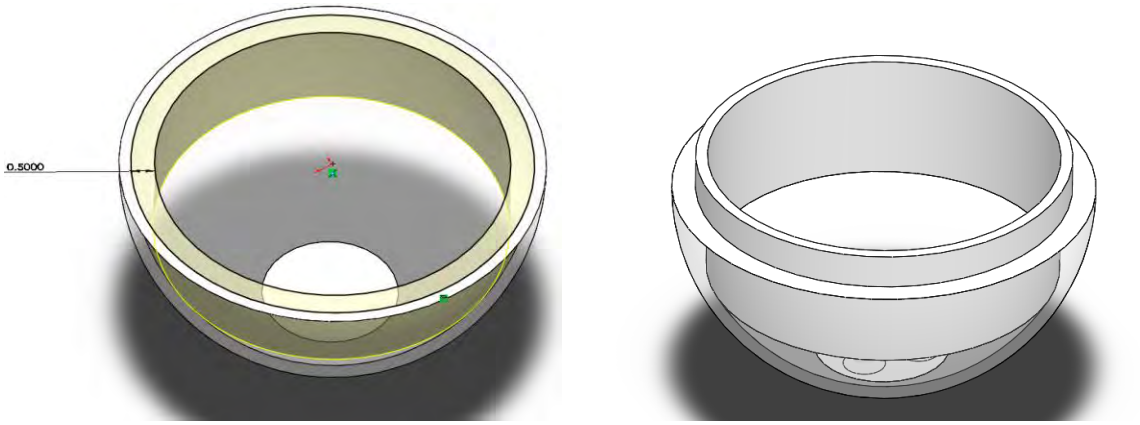

Figure 35: (Left) Creating a sturdy platform from which to build further upward from. **(Right)** The model after a vertical extrusion

Next a plane was created 0.1 inches above the top surface and a circle was sketched that had a diameter 0.5 inches larger than the outer lip of the top extrude (spiral path). This step was due to the un-sketchable nature of the top outer lip. Next a threading shape was selected to be a triangle with a base of 0.2 inches and a height of 0.25 inches where the peak of the triangle was in the center of the base. Using a Helix/Spiral function beginning at the spiral path circle, a helix was created. The geometry began 0.35 inches below the spiral path and went around the surface 3 full times. The helix was constrained by pitch and revolution, having a pitch of 0.22 inches. The sweep command was used with the helical path and the triangle in order to create the threading. This hemisphere in its entirety can be seen below in Figure 36.

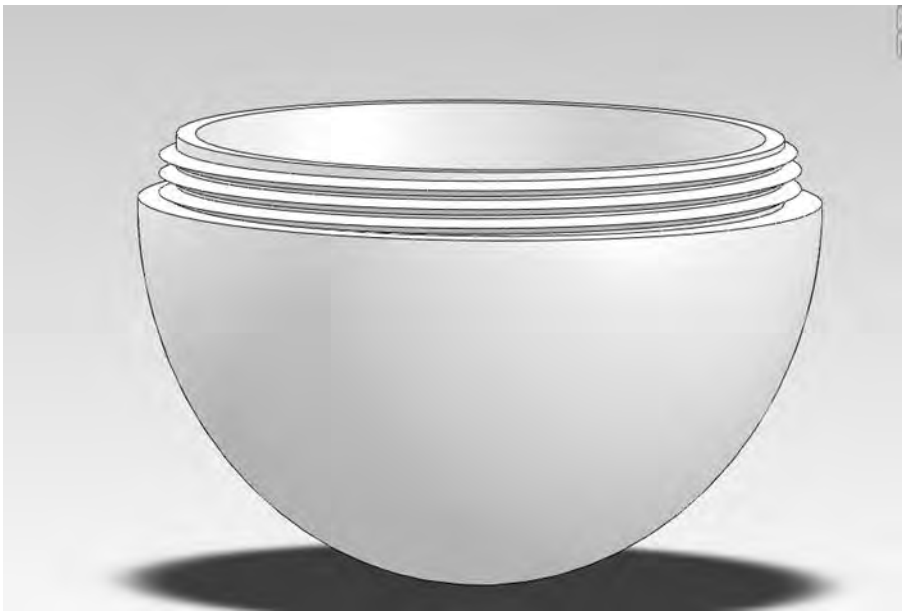

Figure 36: Male half of the sphere used to represent the asteroid.

Female Only Steps:

The inner lip of the top surface, using the convert entities command, was converted into a sketch. This sketch was then used with the Extrude Cut command to cut 1 inch downward into the sphere. This cut did not remove much material, keeping the majority of the integrity of the hemisphere, but did create a surface from which threading could easily be added. The model after this step can be seen in Figure 37 below.

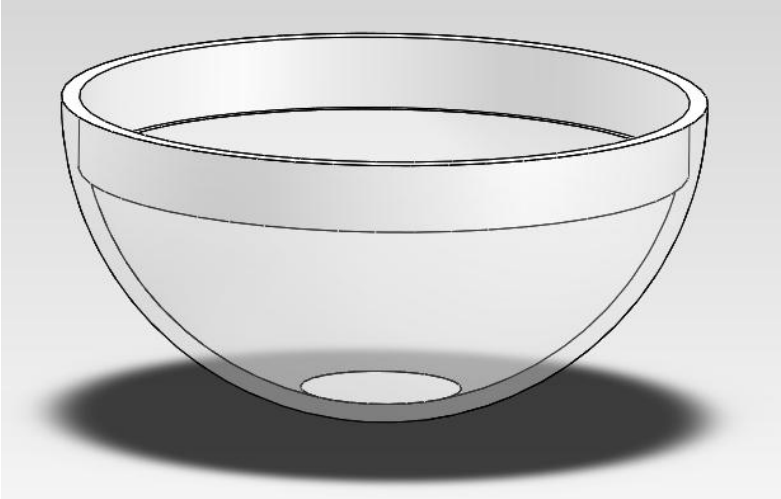

Figure 37: The female hemisphere model just prior to the addition of threading.

Next the same 2.0 inch by 2.5 inch triangular shape was created. Then a circle the same diameter as the inner lip was created on a plane that was 0.1 inches below the lip. Using the Helix/Spiral function a helix was created, defined by pitch and revolution, having a pitch of 0.22 inches and revolving four times from the aforementioned circle sketch to the bottom of the ridge. The Sweep command was used referencing the triangle and the spiral, creating the threading for the second hemisphere of the model asteroid. The model in its entirety can be seen in Figure 38 below.

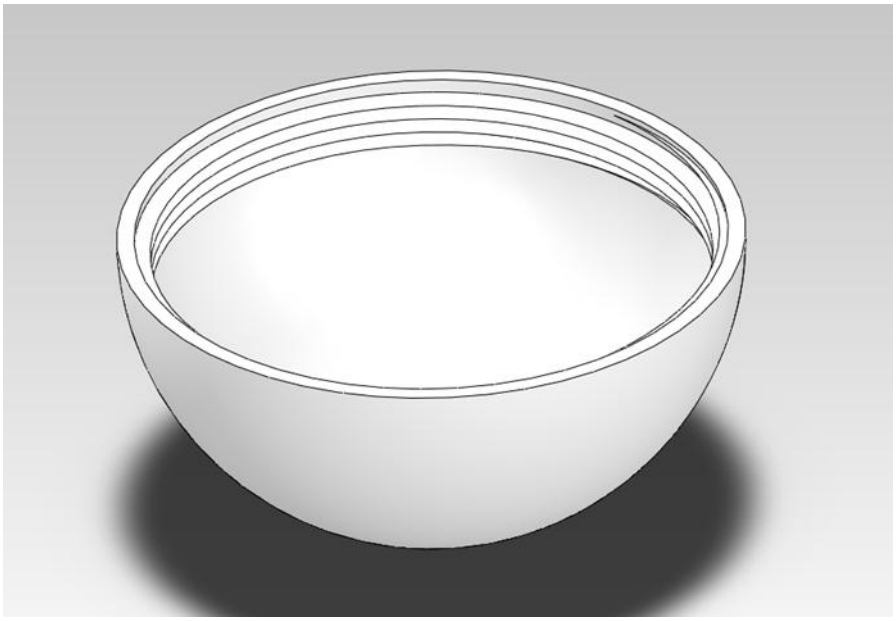

Figure 38: Female half of the sphere used to represent the asteroid.

In order to confirm that the threading would allow both hemispheres to screw together, an assembly was created and both parts were inserted. A concentric mate was used on both outer lips of each hemisphere and it was ensured that the insides of the hemispheres were facing each other. This mate established that the threading would line up. The faces of the threading that were showing toward the center of the two hemispheres were mated coincidently. This meant that the threads would always be touching, but would allow the hemispheres to screw together. Upon the sphere being completed, it was seen that the threading matched up without issue. The assembly can be seen in Figure 39 below.

Figure 39: The two sphere halves assembled together. The transparency of the sphere was changed so the pedestals could be seen on each inner pole of the hemispheres. The solid wall just above the bottom pillar is the face that needed to be made in order to create the male threading. This wall could not be made transparent without the entire male half being made invisible.

In order to make our 8 inch diameter sphere neutrally buoyant, it must have the same mass as an 8 inch diameter sphere of water, 18.50 pounds. The density of the ABS plastic used to make this waterproof box is about 63.59 $\frac{lb}{ft^3}$ which is about 1.161 $\frac{lb}{ft^3}$ or 1.8% more dense than water. Therefore, by filling this sphere entirely with water will make it very slightly denser than water and will sink, by itself. The total mass of just the spherical housing was 2.268 pounds,

which is .04138 pounds heavier than an equal volume of water. Therefore, something needs to be added to the sphere to bring down the mass.

The aforementioned waterproof box will be glued to the bottom pedestal, and as the two hemispheres are screwed together, it will tightly lock the box into place. The box has external dimensions of 5.89 x 3.74 x 3.16 [inches] and internal dimensions of 5.42 x 3.31 x 2.52 [inches]. The largest external dimension is, as was stated above, the distance between the pedestals of the assembled sphere. The internal dimensions are large enough to hold the electronics loosely. The waterproof box with the data acquisition components inside is very positively buoyant, being made mostly of non-dense plastic and air. The box with the components measured in at 0.88 pounds, which is 1.633 pounds lighter than the same volume of water, which would measure 2.51 pounds. As was stated above, the mass of the sphere filled with water is 0.04138 pounds heavier than an equal volume of only water. Therefore, the waterproof box was weighted using lead weights adding a total of just less than 1.592 pounds. By combining the assembled sphere along with the waterproof box, neutral buoyancy was achieved.

A cross sectional view of the sphere holding the waterproof box, which is also holding the electronic components can be seen below in Figure 40.

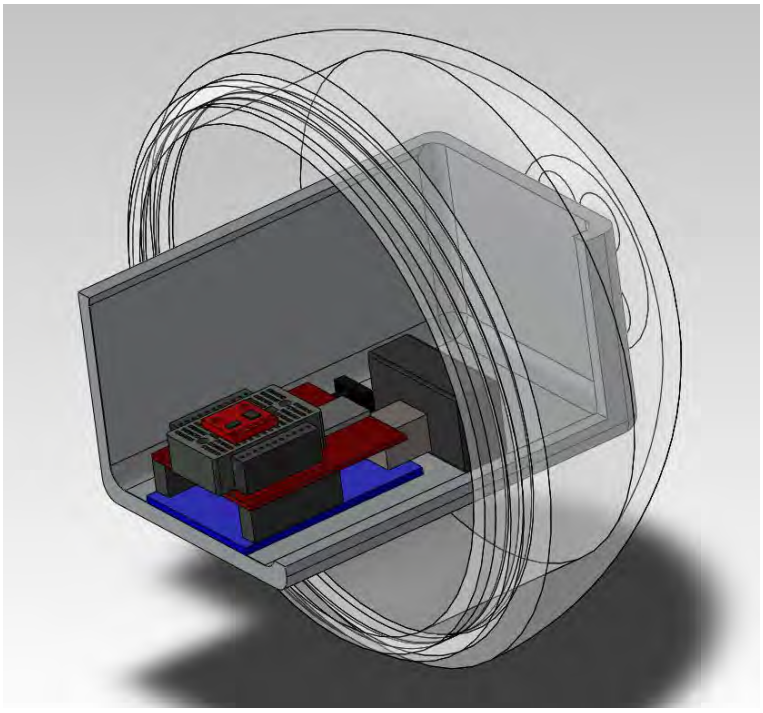

Figure 40: A semi -cross sectional view of the waterproof box, opened for display, housing the electronic components, sitting inside half of the sphere. This SolidWorks assembly shows how everything will be enclosed inside the sphere. The added weights and wires are not shown in the picture.

An actual picture of the waterproof box used can be seen below in Figure 41.

Figure 41: The waterproof housing used to house the data acquisition electronics. Can be found floating around the Chase lab rooms.

Electronics Design & Implementation:

The first step in development was to determine what needed to be recorded. For the purposes of this experiment, accelerations were the main focus. The objective was to create an untethered datalogger to record the internal accelerations yielded from perturbing the underwater asteroid model with a ballistic. It was determined that a computation unit was required to accomplish such a task, and the chosen device was an arduino uno. The arduino uno is comprised of an ATmega328 microcontroller, 14 digital I/O pins, 6 analog inputs, and 32k flash memory. It has an input voltage that ranges from 7 to 12 volts, and a clock speed of 16 MHz.

This physical computing platform provided an outlet to the other components involved in the design. A sensor with the capability of monitoring accelerations was then sought out, one that would be compatible with the arduino. After some investigation, Sparkfun's inertial measurement unit (IMU) digital combo board was selected. This breakout board interacted with the Arduino over I^2C, and had a built in ADXL345 accelerometer and ITG-3200 gyro which provided 6 degrees of freedom. These degrees of freedom consist of forward/back, up/down, left/right, pitch, yaw, and roll. The first three are perpendicular, spatial axes, and the last three account for rotational motion. Combined, they account for all possible directions of movement.

For a means of saving the data collected from the IMU, a microSD shield was purchased as the last essential component of the design. The shield established communication with the Arduino Uno over an SPI interface. The SCK, DI, and DO pins of the microSD socket are broken out to the ATmega168/328's standard SPI pins, while the CS pin is broken out to Arduino's D8 pin. A SanDisk microSD card was used and inserted in the socket of the shield. To complete this untethered design, a 9v battery was bought and connected to the Arduino via a barrel jack adapter.

A block diagram was created to give a upper level view of communication between devices (Figure 42):

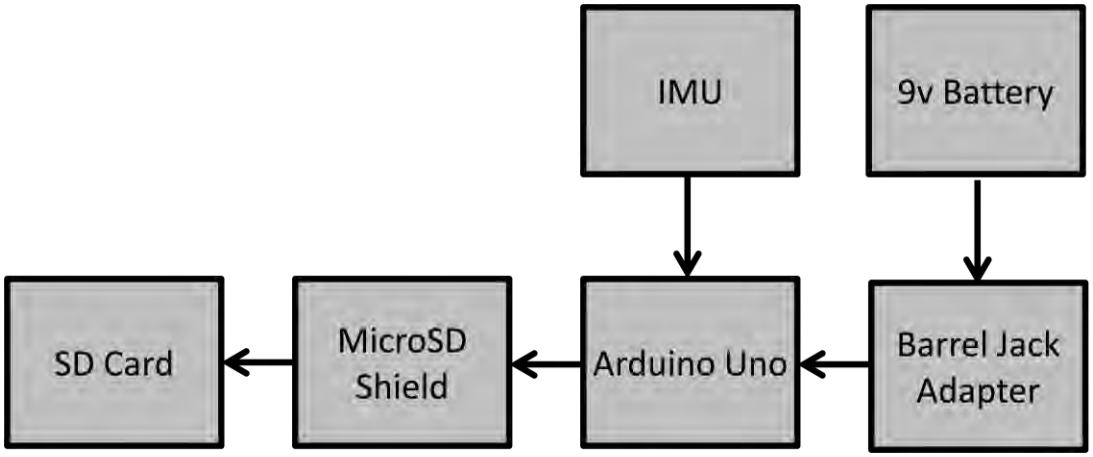

Figure 42: Block Diagram (Flow of Communication)

Also, a piece of prototyping software called Fritzing was used to model this design. Below (Figure 43) is an image of the setup to depict how each component linked together:

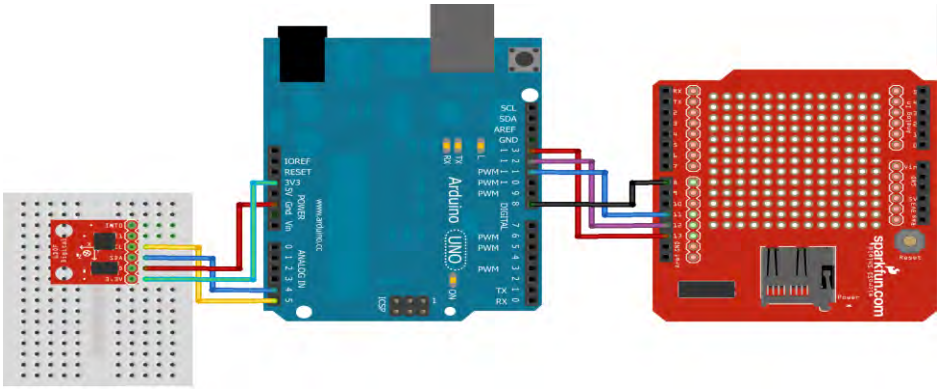

Figure 43: Virtual Prototype of Electronics

After conceptual and prototyping stages of design, the final product was built and is shown in the photograph below (Figure 44).

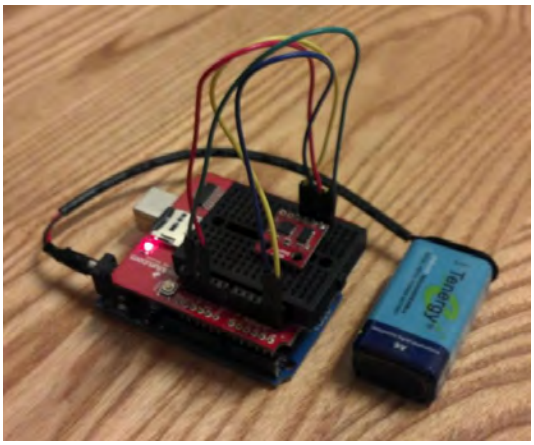

Figure 44: Photo of Finalized Electronics

Software for Electronics:

Following the design and implementation of the physical components of the datalogger, the Arduino needed to be programmed to record, print, and save what was needed. Searching through Arduino's website, the appropriate IDE version was downloaded and installed. The open source libraries provided some understanding of the language associated with the device. After using some simple open source examples that come in the Arduino IDE package and sifting through open source programs dealing with accelerometers, a program was written that printed accelerations and gyro data to the serial monitor. These were unformatted and merely raw values that needed to be scaled in the future.

After having successfully established communication with the IMU, code to utilize the microSD Shield was then incorporated into the software. There were many examples that dealt with this connectivity, but few for this particular shield. Before trying to save the IMU data, the recognition of the SD card was tested. It was discovered that the shield heavily favors pin 8 as a breakout to the Arduino, and any other pin assigned for outputting wouldn't consistently print data. The functionality of extracting sensor data from the IMU and the saving of data using the SD shield to the microSD card was then combined and formatted. The chosen data rate for the Arduino was set to 9600 bits per second (baud rate), or rather symbols per second, for serial data transmission. The flow chart below (Figure 45) describes the process of the arduino code which logs the data collected from the IMU. The datalogger accounts for the x, y, and z components of acceleration and angular velocity (gyroscope). All data is time stamped in milliseconds.

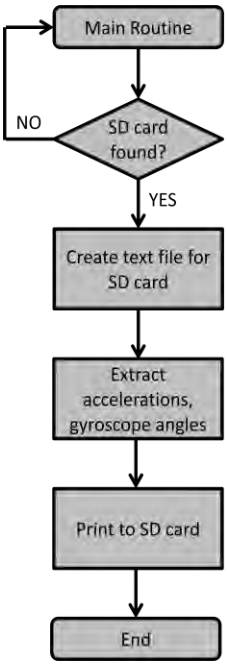

Figure 45: Flow Chart of Arduino Code

With the given output accelerations from the Arduino in this experiment, it was clearly determined in the sections of data where the impact occurred. This can be seen below in Figure 46, where the steady acceleration values depict before the impact (up to approximately 5.8 milliseconds), the peak of the data is the point of impact, and immediately following that shows post-impact data.

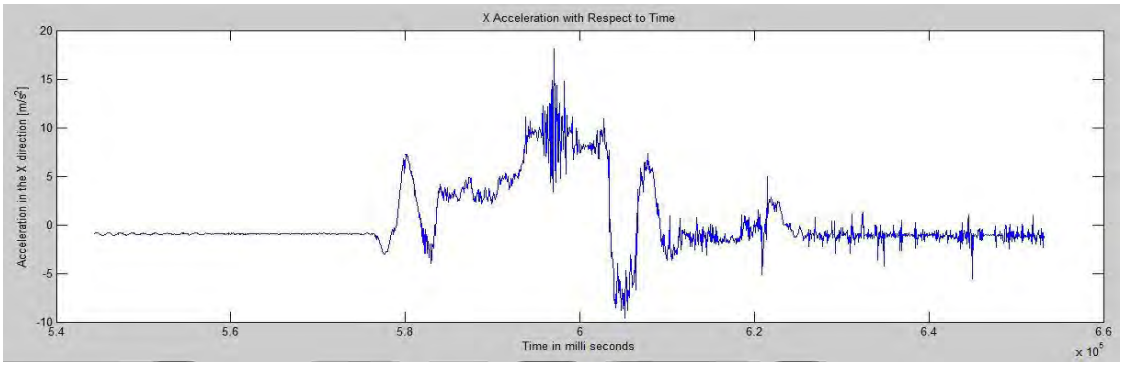

Figure 46: A sample of the acceleration data along the X axis

Video Processing Platform

Video Hardware

Equipment Used:

-3 Seaview Underwater Cameras

-DVR

-Computer

-12 volt DC power source

-Camera suspension rig

The three cameras were aligned with each axis so that all positional translations could be recorded. The Seaview cameras were chosen because they perform well underwater and were readily available. The cameras were positioned underwater because a location outside the testing cube was not feasible due to the opaque cube walls. To position the cameras within the cube an overhead support structure was constructed from a cross of PVC pipe. The cameras were suspended by string from this support structure. Using string, the X and Z axis cameras were positioned in the corners of the tank, giving more room and allowing a wider area of the tank to be recorded by each camera. The third camera was suspended from the center of the support structure over the testing rig. The cameras were powered by the 12 volt DC power source.

After the video was record it was displayed on the DVR monitor to verify the cameras were positioned correctly. The video data was extracted from the DVR to be used as an input for the motion tracking software to be processed.

Object Tracking Software

Due to the existence of its object tracking libraries, the software was programed in Matlab. The software post processes the video, rather than analyze it in real time. It does this because there are far fewer limitations when post-processing video as opposed to performing real-time analysis.

The video data, extracted from the DVR, came encoded in the Audio Video Interleave (AVI) format, which is universally compatible with Matlab. The data files of the experiment were edited before they were processed so that the frames of interest could be isolated. Then each truncated data file was run through the software to calculate the model asteroids positional data.

The program initialized and configured a Kalman Filter, Video Reader, Video Player, Image Foreground Detector, and Blob Analyzer. The Kalman Filter was chosen due to the large quantity of existing documentation on the use of this filter for object tracking and the quality of the output when it is properly configured. The Kalman Filter can, however, be one of the more difficult to configure properly. For this reason a function was written that was dedicated to configuring the filter.

The Kalman Filter is an estimation algorithm that bases its estimations on a series of measurements observed over time. These measurements are made up of noise and various other inaccuracies, and produce more precise estimates of unknown variables than estimations based on a single measurement alone. The algorithm is made up of two parts, it first predicts what the

output will be, and then corrects itself based on the next part of the input. The filter is utilized to analyze one frame of video and determine where the object is. Then, after reading the next frame, doubles back to correct its initial predictions based on new data gathered from the following frame.

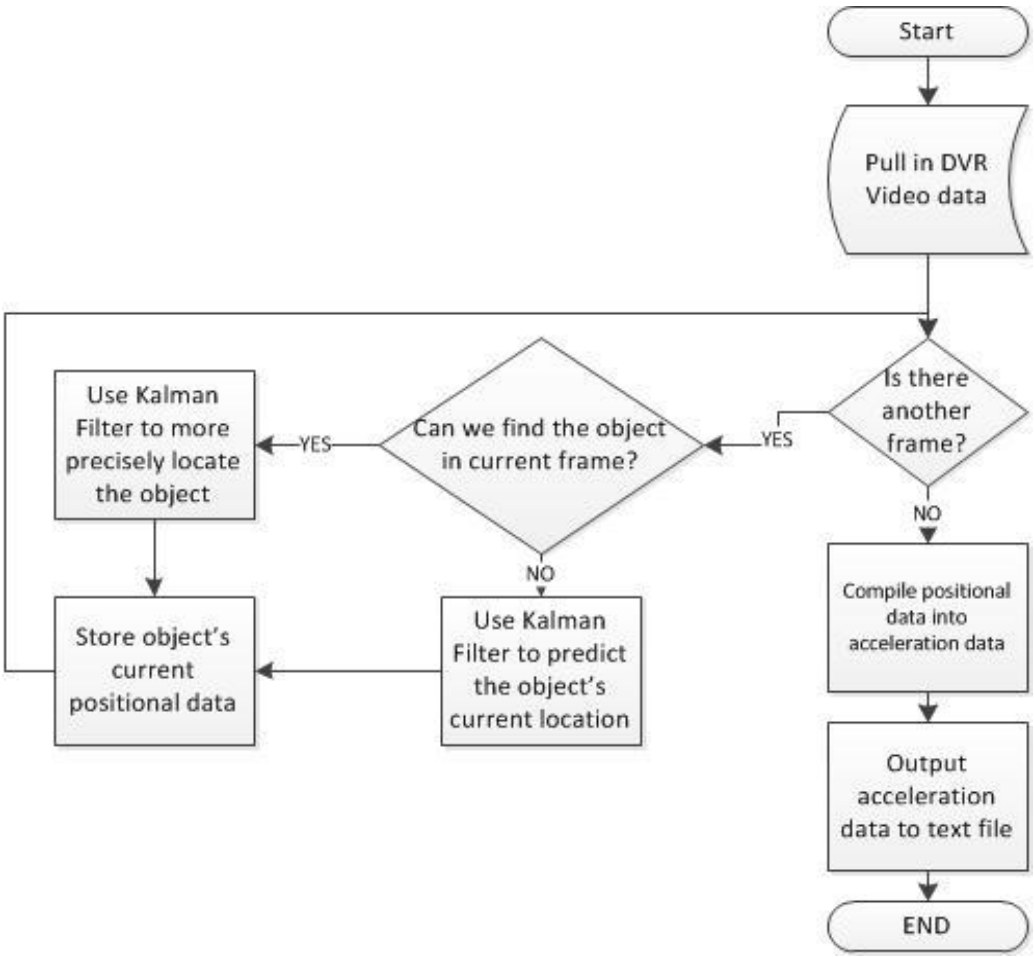

Figure 47: A flow chart describing the video tracking software's computational path

The most prominent issue found was learning how to properly configure the Kalman Filter. Another issue was the video cameras catching the reflection of the asteroid on the top of the water. This confused the object tracking software by making the video appear to have an additional asteroid (Figure 48). More issues arose when the ballistic entered the water and the ripples on the water's surface disrupted the reflection of the asteroid. This was resolved by repositioning the cameras during testing to not pick up the reflection of the asteroid.

Figure 48: Asteroid Reflection Being Captured By Camera

Experimental Error and Limitations

Test Platform

The cube and the rig account for much of the experimental error for the test platform. Due to the fact that water in the cube has a small volume, only 3'x3'x2', minor disturbances in the water have a major effect on the asteroids acceleration and displacement. A test could not begin without first waiting for the water to settle to a calm state. The time between tests led to a loss of data due to the Arduino running out of power between tests. Due to the cube's design flaws the floor of the cube is not level. This means that the angles used and the locations of the x, y and z axis were not accurate. This also may have accounted for some error in the final test data.

The experimental data error created by the cube and rig can be fixed by redesigning the cube. The test cube could be made larger for future tests. This would disperse any water disturbances across a larger area and effectively reduce its effect on the model asteroid. The floor of this new cube should also be made level. This would make any angle measurements and axis locations much more accurate.

Asteroid Model

The asteroid model, which houses the electronics, accounts for some error as well. The model should be neutrally buoyant in order to properly simulate a collision in space. Unfortunately it was very difficult to make the asteroid model perfectly neutrally buoyant.

Instead, the model was slightly positively buoyant and would very slowly rise in the water. This may have added resistance to the projectile. The electronics were not completely secure within the waterproof box. This may have caused them to wobble when the model was hit with the projectile. Due to setbacks in the creation of the model, the asteroid was not a perfect sphere. The two model halves did not fit perfectly together and had to be connected using tape. This tape and added surface area from the rapid prototyping support material caused drag that was not accounted for in the calculations. This would cause error when comparing the underwater model to the ideal empty vacuum conditions. The weight within the model sphere is also not uniform. This causes the asteroid to be bottom heavy and may cause error in the spin of the model.

Ballistic and Launcher

The ballistic launcher also account for experimental error. The spring within the launcher is not replaced for each test. This means that the spring constant changes with each subsequent test. This may lead to some error in the force of the ballistic. The marbles used as the ballistic for the test have a smaller diameter than the PVC pipe. The makes the launcher not as accurate as desired and may have led to some of the ballistics to not hit perfectly center on the asteroid. This would cause the displacement and acceleration data to be slightly skewed.

Electronics

Minor error also occurred with the Arduino Uno and IMU. These electronic were not properly secured within the waterproof box. This may have led to the IMU shaking out of place and creating false data. There is also no way to check this because the electronics are not visible from outside the model asteroid. The waterproof box was also not properly secured inside the

model. The acceleration of the model asteroid should be zero along the X axis when it is at rest. It can be observed in Figure 46 that this is not the case. To fix this error, putty could be used to attach the electronics to the waterproof box and the waterproof box to the model asteroid. This would make the acceleration data received from the Arduino more accurate.

Seaview Cameras

The three Seaview cameras were in the water with our setup for each test. They record the test and then the recordings were used to get the positional data of the model asteroid. When the ballistic was shot out of the launcher, however, it caused a shockwave in the water. This shockwave disturbed the water and caused the cameras to shake. This may have led to serious error when determining the position and acceleration of the ballistic and model asteroid. It was also very difficult to make the cameras perfectly level. This may have caused error to occur when comparing the videos for each axis. These issues could be corrected by potentially using the cameras outside the water. This way they could be placed on a level surface and not be affected by what is happening within the cube.

Results and Conclusions

Rig&Ballistic

For this experiment, the rig that was designed was an ideal setup, in which any point on the top hemisphere of the model asteroid could be hit with varying forces and directions. With the equations in the theory section, the final velocity of the ballistic as it impacted the model asteroid could be determined. (It was considered that this velocity differed from the ballistic's initial velocity).

Model Asteroid

The model asteroid that was created had initial specifications to not fracture, have simple geometry, and to be neutrally buoyant. All of these criteria were met in experimental testing. It was also found that the model asteroid sufficiently held the electronics and kept them dry and intact during testing. The numeric gyroscopic values of the asteroid could not be acquired with the use of this specific asteroid model due to the inertial properties of the internal water of the asteroid. However, it was able to determine the existence of spin during testing was negligible.

Arduino Data

With the given output accelerations from the Arduino in this experiment, it was clearly determined in the sections of data where the impact occurred. This can be seen above in Figure 46, where the steady acceleration values depict before the impact, the peak of the data is the end of impact, and immediately following that shows post-impact data. The gyroscopic output of the tests confirmed that there was no significant spin in the asteroid during testing.

Video Processing

The video processing portion of this experiment is still a work in progress, due to multiple errors to work through in the output of the object tracking software. Once these errors are corrected, it is expected that the output of the software should provide reasonably accurate location data of the asteroid and ballistic with respect to time. In addition to this, the acceleration data could be solved from the location output of the software, and then correlated with the Arduino acceleration data. Because of this, the asteroid motion can be discerned from motion of the sensor in the asteroid. Once video processing has been obtained, correlation of visual data and the Arduino data will enable the evaluation of the feasibility and practicality of this experimental test platform to be determined.

Future Work

The success of this project in the future requires more scenarios to be observed and a more in-depth look into space and orbital mechanics. It is known that there are many differences between an underwater model and the mechanics of a body going through space. To properly assess the forces that act upon the asteroid model and how this compares to a real asteroid in space, more variables need to be considered. For simplicity reasons these were not considered in the first year of the project because the purpose of this project was to ascertain the feasibility of this experiment. Once the feasibility of this experiment can be confirmed, the experiment can gain more complexity and become a more accurate model of an asteroid moving through space.

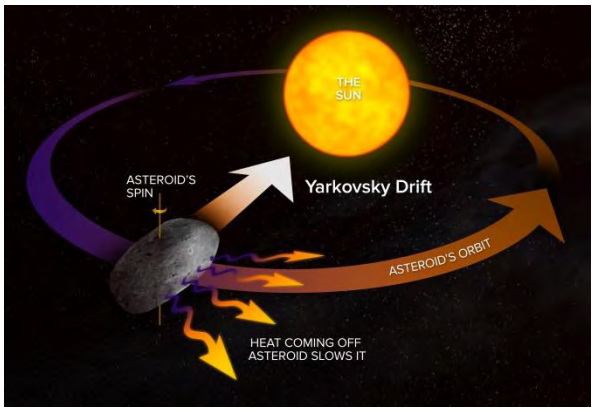

Figure 49 : A figure displaying Yarkovsky's effect

Yarkovsky's Effect:

This effect causes the body in question to rotate in a zero gravity environment, and can be seen above in (Figure 49). This effect occurs on asteroids with a diameter ranging from 10 cm to 10 km. The asteroids that were modeled in this experiment are well within this diameter range. In the experiments so far, the asteroid model has been stationary with no rotation. Future stages of this project should include rotation to better simulate an object that is traveling in space. This will have a large impact in the results of the experiment due to the fact that the ballistic will not hit a rotating body evenly which may cause deflection. This could be done by having the asteroid spinning at a certain fixed rotational velocity. Some complications that may arise with doing this is the drag cause by the water. This drag will cause the asteroid to resist rotation and ultimately cause it to slow down and stop spinning.

Initial Velocity:

When doing the experiments on the asteroid, the model was at a fixed position with no initial velocity. This is not the case for asteroids moving through space. The asteroids initial

velocity and momentum will have a drastic impact on the results of this experiment. The model asteroid could be pulled along a tow tank and shot at the moment the model reaches the previously determined velocity. A problem that may come from doing this is that the timing and accuracy of the shot will need to be perfect in order to hit a moving sphere at the desired angle. A computer program and new rig may need to be implemented to time the shot correctly and hit the desired location on the asteroid.

Solid Sphere:

The asteroid that was modeled is based off a solid spherical asteroid made of granite. The model asteroid created for this experiment is instead a hollow sphere filled with water. This creates problems when trying to measure rotation. A hollow body will rotate differently than a solid one. A solid asteroid will also react differently to getting hit with a ballistic. This type of impact could not be modeled with the current asteroid design. If a new solid asteroid was made it could also be created with different materials. A brittle material would be best due to the fact that this would also allow for a failure analysis to be done determining if the asteroid would crack or deform under the stress from the impact. The asteroid could also be made to break apart upon impact and the resulting accelerations and velocities could be measured using the tracking software. A hollow sphere was used was so that the Arduino Uno and IMU could be placed inside the asteroid during impact. This would be impossible with a solid sphere and another method for measuring acceleration would need to be implemented to compare to the video tracking software.

Acknowledgements

This work is the result of research sponsored in part by the New Hampshire Sea Grant College Program through NOAA grant # NA10OAR4170082, the UNH Marine Program, and Professor May-Win Thein.

We would also like to thank Paul Lavoie, Donya Frank, Dave Folta (aerospace engineer of the flight dynamics analysis branch of the NASA Goddard Space Flight Center), and Custom Welding for their generous assistance in our project.

References:

[1] "NEO Discovery Statistics." *NEO Discovery Statistics*. N.p., n.d. Web. 28 Apr. 2013. <http://neo.jpl.nasa.gov/stats/>.

[2] "Frequently Asked Questions." *Frequently Asked Questions*. N.p., n.d. Web. 28 Apr. 2013. <http://neo.jpl.nasa.gov/faq/>.

[3] "NASA Asteroid News." *NASA*. N.p., n.d. Web. 28 Apr. 2013. <http://www.nasa.gov/mission_pages/asteroids/news/asteroid20130215.html>.

[4] Hildebrand, Alan R. "Chicxulub Crater." Editorial. *Geology* 1991: 867-71. Print.

[5] "Asteroids." NASA, n.d. Web. 28 Apr. 2013. <http://nssdc.gsfc.nasa.gov/planetary/text/asteroids.txt>.

[6] Reit, Emily, Trevor Barton, Mark Fischer, Eric Swank, and Garrett Bael. *Asteroid Mitigation Strategy*. N.p., n.d. Web. <http://lunar.earth.northwestern.edu/courses/351/asteroid.team.pdf>.

[7] "News Article: Planetary Defense Conference, April 2009." *News Article: Planetary Defense Conference, April 2009*. N.p., n.d. Web. 28 Apr. 2013. <http://impact.arc.nasa.gov/news_detail.cfm?ID=181>.

[8] "Facilities." *Ocean Engineering*. University of New Hampshire, n.d. Web. 28 Apr. 2013. <http://www.unh.edu/oe/facilities/wavetowtank.htm>.

[9] "Yarkovsky Effect." *Yarkovsky Effect*. Princeton, n.d. Web. 28 Apr. 2013. <http://www.princeton.edu/~achaney/tmve/wiki100k/docs/Yarkovsky_effect.html>.

[10] "NRAO: National Radio Astronomy Observatory." *Getting the Right Spin on a Close-Passing Asteroid*. N.p., n.d. Web. 29 Apr. 2013. <http://www.nrao.edu/pr/2013/asteroid/>.

[11] "Please Enable JavaScript in Your Browser." *OpticsPlanet.com*. N.p., n.d. Web. 29 Apr. 2013. <http://www.opticsplanet.com/s3-t4000-waterproof-dry-protective-cases.html>.

[12] "Facilities." *Ocean Engineering*. N.p., n.d. Web. 29 Apr. 2013. <http://www.unh.edu/oe/facilities/wavetowtank.htm>.

[13] "Yarkovsky Effect." *Yarkovsky Effect*. N.p., n.d. Web. 29 Apr. 2013. <http://www.princeton.edu/~achaney/tmve/wiki100k/docs/Yarkovsky_effect.html>.

[14] "Planets - Zoom Astronomy." *Planets - Zoom Astronomy*. N.p., n.d. Web. 29 Apr. 2013. <http://www.enchantedlearning.com/subjects/astronomy/planets/>.

[15] "Spacecraft Magnetic Torques." NASA, n.d. Web. <http://www.dept.aoe.vt.edu/~cdhall/courses/aoe4065/NASADesignSPs/sp8018.pdf>.

[16] "Solar Wind." *Solar Wind*. N.p., n.d. Web. 29 Apr. 2013. <http://pluto.space.swri.edu/IMAGE/glossary/solar_wind.html>.

[17] Levesque, Zachery, and Thomas Paquet. *Asteroid Mitigation: Impact Motion Analysis and Data Collection Sphere Optimization*. Tech. N.p.: n.p., n.d. Print.

Appendix

Arduino Datalogger Code:

```
//Datalogger for AMP simulations

#include <CommunicationUtils.h>
#include <DebugUtils.h>
#include <FIMU_ADXL345.h>
#include <FIMU_ITG3200.h>
#include <FreeSixIMU.h>

#include "SD.h"
#include <Wire.h>
#include "RTClib.h"


//Declaring RTC (time) device
RTC_DS1307 RTC;

//select breakout: SD shield to Arduino
const int chipSelect = 8;

// Set the FreeSixIMU object
FreeSixIMU sixDOF = FreeSixIMU();

float g = 9.81; //gravity
float gyroScale = 14.375; //gyro sensitivity scale factor
float accelScale = 259.1548944; //accel sensitivity scale factor

//-----------------------------------
void setup()
{
	Serial.begin(9600); //data rate

	sixDOF.init(); //begin the IMU

	Serial.print("Initializing SD card...");
	// make sure that the default chip select pin is set to
	// output, even if unused:
	pinMode(8, OUTPUT);

	pinMode(0, OUTPUT);   //for sleep on the accelerometer

	pinMode(1, OUTPUT);   //for 6g select on the accelerometer

	// see if the card is present and can be initialized:
	if (!SD.begin(chipSelect))
	{
```

```
		Serial.println("Card failed, or not present");
		// don't do anything more:
		return;
	}
	Serial.println("card initialized.");

	File dataFile = SD.open("datalog.txt", FILE_WRITE);
	dataFile.println("Millis  aX    aY    aZ    gX    gY    gZ");
	dataFile.close();

	Serial.println("Millis  aX    aY    aZ    gX    gY    gZ");

}

//-------------------------------------
void loop()
{

	Wire.begin();

	float imu[6]; //declare array of values

	sixDOF.getValues(imu);

	//accelerations
	float X = imu[0];
	float Y = imu[1];
	float Z = imu[2];

	//gyro angular velocities
	float V1 = imu[3];
	float V2 = imu[4];
	float V3 = imu[5];

	DateTime now;

	// open the file.
	File dataFile = SD.open("datalog.txt", FILE_WRITE);

	// if the file is available, write to it:
	if (dataFile) {

		// fetch the time
		now = RTC.now();

		dataFile.print(millis());
		dataFile.print("  ");

		dataFile.print((aX/accelScale)*g);
		dataFile.print("  ");
		dataFile.print((aY/accelScale)*g);
```

```
                dataFile.print("  ");
                dataFile.print((aZ/accelScale)*g);
                dataFile.print("  ");

                dataFile.print(gX/gyroScale);
                dataFile.print("  ");
                dataFile.print(gY/gyroScale);
                dataFile.print("  ");
                dataFile.println(gZ/gyroScale);

                dataFile.close();

                // print to the serial port too:

                // log time
                Serial.print(millis(), DEC);
                Serial.print("  ");

                Serial.print((aX/accelScale)*g);
                Serial.print("  ");
                Serial.print((aY/accelScale)*g);
                Serial.print("  ");
                Serial.print((aZ/accelScale)*g);
                Serial.print("  ");

                Serial.print(gX/gyroScale);
                Serial.print("  ");
                Serial.print(gY/gyroScale);
                Serial.print("  ");
                Serial.println(gZ/gyroScale);
        }
        // if the file isn't open, pop up an error:
        else
        {
                Serial.println("error opening datalog.txt");
        }
}
```

Matlab Object Tracking Code

```
function f = objecttrack( filename )

%{
    initialize all the variables to be used in the script
%}

frame                   = [];  % A video frame
detectedLocation  = []; % The detected location
trackedLocation   = [];  % The tracked location
label                   = '';   % Label for the asteroid
utilities               = [];  % Utilities used to process the video
```

```
    % The procedure for tracking a single object.
%         -Creates a vision.KalmanFilter object by using
configureKalmanFilter
%         -Use predict and correct methods in a sequence to eliminate any
noise
%         present in the environment
%         -Use predict method by itself to estimate asteroids's location
should
%         it become occluded or temporarily lost
%
%         The selection of the Kalman filter parameters can be challenging,
hence
%         the use of configureKalmanFilter to simplify this problem
%
%         The trackSingleObject function includes nested helper functions.
The
%         previous top-level variables are used to transfer the data between
the
%         nested functions.
    function tracksingleobject(param)
    % Create utilities used for reading video, detecting moving objects, and
    % displaying the results.
      utilities = createUtilities(param);

      isTrackInitialized = false;
      while ~isDone(utilities.videoReader)
        frame = readFrame();

        % Detect the asteroid.
        [detectedLocation, isObjectDetected] = detectObject(frame);

        if ~isTrackInitialized
          if isObjectDetected
            % Initialize a track by creating a Kalman filter when the
asteroid
            % is detected for the first time.
            initialLocation = computeInitialLocation(param,
detectedLocation);
            kalmanFilter = configureKalmanFilter(param.motionModel, ...
              initialLocation, param.initialEstimateError, ...
              param.motionNoise, param.measurementNoise);

            isTrackInitialized = true;
            trackedLocation = correct(kalmanFilter, detectedLocation);
            label = 'Initial';
          else
            % clear location and label for detection
            trackedLocation = [];
            label = '';
          end

        else
          % Use the Kalman filter to track the asteroid.
          if isObjectDetected % The asteroid was detected.
            % Reduce the measurement noise by calling predict followed by
```

```
            % correct.
            predict(kalmanFilter);
            trackedLocation = correct(kalmanFilter, detectedLocation);
            label = 'Corrected';
          else % The asteroid was missing.
            % Predict the asteroid's location.
            trackedLocation = predict(kalmanFilter);
            label = 'Predicted';
          end
        end

        annotateTrackedObject();
      end % while

      showTrajectory();
    end

    % Create utilities for reading video, detecting moving objects, and
    % displaying the results.
    function utilities = createUtilities(param)
      % Create System objects for reading video, displaying video, extracting
      % foreground, and analyzing connected components.
      utilities.videoReader = vision.VideoFileReader(filename);
      utilities.videoPlayer = vision.VideoPlayer('Position',
[100,100,500,400]);
      utilities.foregroundDetector = vision.ForegroundDetector(...
        'NumTrainingFrames', 10, 'InitialVariance',
param.segmentationThreshold);
      utilities.blobAnalyzer = vision.BlobAnalysis('AreaOutputPort', false,
...
        'MinimumBlobArea', 70, 'CentroidOutputPort', true);

      utilities.accumulatedImage      = 0;
      utilities.accumulatedDetections = zeros(0, 2);
      utilities.accumulatedTrackings  = zeros(0, 2);
    end

    % Get default parameters for creating Kalman filter and for segmenting
the
    % asteroid
    function param = getDefaultParameters
      param.motionModel           = 'ConstantAcceleration';
      param.initialLocation       = 'Same as first detection';
      param.initialEstimateError  = 1E5 * ones(1, 3);
      param.motionNoise           = [25, 10, 1];
      param.measurementNoise      = 25;
      param.segmentationThreshold = 0.05;
    end

    % Read in the next video frame from the video file.
    function frame = readFrame()
      frame = step(utilities.videoReader);
    end

    % Detect and annotate the asteroid in the video.
```

```
    function showDetections()
      param = getDefaultParameters();
      utilities = createUtilities(param);
      trackedLocation = [];

      idx = 0;
      while ~isDone(utilities.videoReader)
        frame = readFrame();
        detectedLocation = detectObject(frame);
        % Show the detection result for the current video frame.
        annotateTrackedObject();

        % To highlight the effects of the measurement noise, show the
detection
        % results for the 40th frame in a separate figure.
        idx = idx + 1;
        if idx == 40
          combinedImage = max(repmat(utilities.foregroundMask, [1,1,3]),
frame);
          figure, imshow(combinedImage);
        end
      end % while

      % Close the window which was used to show individual video frame.
      uiscopes.close('All');
    end

    % Detect the asteroid in the current video frame.
    function [detection, isObjectDetected] = detectObject(frame)
      grayImage = rgb2gray(frame);
      utilities.foregroundMask = step(utilities.foregroundDetector,
grayImage);
      detection = step(utilities.blobAnalyzer, utilities.foregroundMask);
      if isempty(detection)
        isObjectDetected = false;
      else
        % To simplify the tracking process, only use the first detected
object.
        detection = detection(1, :);
        isObjectDetected = true;
      end
    end

    % Show the current detection and tracking results.
    function annotateTrackedObject()
      accumulateResults();
      % Combine the foreground mask with the current video frame in order to
      % show the detection result.
      combinedImage = max(repmat(utilities.foregroundMask, [1,1,3]), frame);

      if ~isempty(trackedLocation)
        shape = 'circle';
        region = trackedLocation;
        region(:, 3) = 5;
        combinedImage = insertObjectAnnotation(combinedImage, shape, ...
          region, {label}, 'Color', 'red');
```

```
      end
      step(utilities.videoPlayer, combinedImage);
    end

    % Show trajectory of the asteroid by overlaying all video frames on top
of
    % each other.
    function showTrajectory
      % Close the window which was used to show individual video frame.
      uiscopes.close('All');

      % Create a figure to show the processing results for all video frames.
      figure; imshow(utilities.accumulatedImage/2+0.5); hold on;
      plot(utilities.accumulatedDetections(:,1), ...
        utilities.accumulatedDetections(:,2), 'k+');

      if ~isempty(utilities.accumulatedTrackings)
        plot(utilities.accumulatedTrackings(:,1), ...
          utilities.accumulatedTrackings(:,2), 'r-o');
        legend('Detection', 'Tracking');
      end
    end

    % Accumulate video frames, detected locations, and tracked locations to
    % show the trajectory of the asteroid.
    function accumulateResults()
      utilities.accumulatedImage      = max(utilities.accumulatedImage,
frame);
      utilities.accumulatedDetections ...
        = [utilities.accumulatedDetections; detectedLocation];
      utilities.accumulatedTrackings  ...
        = [utilities.accumulatedTrackings; trackedLocation];
    end

    % For illustration purposes, select the initial location used by the
    % Kalman filter.
    function loc = computeInitialLocation(param, detectedLocation)
      if strcmp(param.initialLocation, 'Same as first detection')
        loc = detectedLocation;
      else
        loc = param.initialLocation;
      end
    end
end
```