Internship report

On

Understanding Interaction Between Celestial Bodies Using Python

By

Vineet Vijayan

[PNR No:2019033800147125]

Under the mentorship of

Dr. Balkrishna P. Shah

Assistant Professor

Department of Physics

Faculty of Science

The Maharaja Sayajirao University of Baroda,

Vadodara 390002, INDIA

July 2023

CERTIFICATE

This is to certify that **Vineet Vijayan**, BSc (Graduated, PRN No:2019033800147125) Student of the Department of Physics has satisfactorily completed the FOSSIP (Faculty of Science Summer Internship Program) under the guidance of **Dr. Balkrishna P.Shah**. Completion of internship project entitled as **'Understanding Interaction Between Celestial Bodies Using Python'**, in the academic year 2022-23 and submitted the report on 15th July, 2023.

Dr. Balkrishna P.Shah
Assistant Professor
Department of Physics

ACKNOWLEDGEMENT

I express my gratitude to Dr. Balkrishna P.Shah, under whose mentorship I did my internship work. His constant guidance and support during the entire phase of the summer internship program made it possible for me complete the given objective within the time and to prepare a brief summary in terms of this report.

I am thankful to the Faculty of Science and Department of Physics for giving me the opportunity to join FOSSIP program that made me familiar to the research field. This opportunity helped me understand the basics of scientific approaches used worldwide for application-based research.

INDEX

❖ UNDERSTANDING INTERACTIONS BETWEEN CELESTIAL BODIES USING PYTHON (GLOWSCRIPT)

INTRODUCTION

Code simulation is essential for scientific research, as it allows scientists to create virtual models that mimic real-world systems and phenomena. It also allows scientists to test hypotheses and validate theories before conducting costly experiments. It is also useful for optimizing and designing complex systems and processes, such as engineering, material science, and chemistry. Code simulations are essential for scientific research, as they enable scientists to study complex phenomena, test hypotheses, optimize designs, generate and analyze data, make predictions, and enhance reproducibility. Simulations can be used to generate synthetic data for analysis, predict and forecast future outcomes, explore "what-if" scenarios, and assess the potential impacts of different factors or interventions. They also provide an accessible platform for researchers worldwide. Stimulating space events on computer programs can have several important benefits, such as enhancing education and training, supporting research and development, lowering risk, improving cost effectiveness, and increasing public participation. Here are a few reasons why it is important:

Education and Training

Simulating space events enables astronauts, engineers, and scientists to have immersive and realistic training experiences. They may hone their abilities to manage space missions, spacecraft operations, and numerous circumstances they might run into in space by using it to practice and practice. The complexity and difficulties of space habitats can be replicated in simulations, which can aid with decision-making, problem-solving, and teamwork.

Research and development

Simulations are a useful tool for work on space-related projects. Computer programs can be used by scientists and engineers to simulate and investigate a variety of space-related phenomena, including celestial mechanics, astrophysics, and planetary exploration. These simulations support the design of next space missions, the testing of hypotheses, and the understanding of the behavior of space systems.

Public Engagement and Outreach

Simulations of space events can also serve as educational tools to engage and inspire the general public. By making space exploration more accessible and interactive, computer programs can help people understand and appreciate the wonders of the universe. Virtual simulations, visualizations, and games can create a sense of excitement and curiosity about space, encouraging interest in science, technology, engineering, and mathematics (STEM) fields.

In conclusion, enhancing education and training, supporting research and development, lowering risk, improving cost effectiveness, and increasing public participation are just a few benefits of exciting space events on computer systems. We can increase our understanding of space, enhance our ability to explore space, and motivate the upcoming generation of space enthusiasts and professionals by utilizing the power of simulations. This report study and analyze three programs that are

a) VISUALIZING THE NEAR-EARTH ASTEROID & CALCULATING DEFLECTION ANGLE
b) REPLICATING DART: DOUBLE ASTEROID REDIRECTION TEST
c) REPLICATING ARTEMIS 1 (EXPLORATION MISSION-1 (EM-1))

Basic idea for this program has been borrowed from [1] [2] [3] and further developed and enhanced the program. In order to help readers comprehend the foundations in depth and the concepts involved in the construction of the program and have also included a full overview of a basic understanding of all the subtopics of python glow script and the space missions. anticipate that this paper will benefit the scientific community by advancing knowledge of Python glow-script and the underlying physics. The study concludes its discussion by comparing our findings to those of [1] [2] [3] and discussing the improvements we made. We will now briefly discuss all these programs one by one along with their fundamentals.

THEORY AND CALCULATIONS

1. FORCE CALCULATION

Newton's law of universal gravitation is a fundamental principle in physics that describes the gravitational force between two objects. It was formulated by Sir Isaac Newton in the late 17th century and revolutionized our understanding of gravity. The law states that every particle or object in the universe attracts every other particle or object with a force that is directly proportional to the product of their masses and inversely proportional to the square of the distance between their centers.

Mathematically, Newton's law of universal gravitation can be expressed as:

$$\mathbf{F} = \mathbf{G} * \frac{(\mathbf{m1} \times \mathbf{m2})}{\mathbf{r^2}} \qquad \textbf{[1]}$$

Where:

- F represents the gravitational force between the two objects.

- G is the gravitational constant, a fundamental constant of nature that determines the strength of the gravitational force. Its value is approximately 6.67430×10^{-11} N(m/kg)^2.

- m1 and m2 are the masses of the two objects.

- r is the distance between the centers of the two objects.

The law tells us that the gravitational force between two objects is directly proportional to the product of their masses. If one of the masses increases, the force of attraction between them also increases. Similarly, if the distance between the objects decreases, the force of attraction becomes stronger.

It is important to note that the gravitational force acts along the line joining the centers of the two objects and it is an attractive force. In other words, it pulls the objects towards each other. Additionally, Newton's law of universal gravitation applies to all objects with mass, from apples falling from trees to planets orbiting the Sun.

This law has profound implications for our understanding of the motion of celestial bodies, such as planets, moons, and stars. It explains why the planets in our solar system orbit around the Sun, why the Moon orbits around the Earth, and why objects fall towards the Earth when dropped. Newton's law of universal gravitation forms the foundation of classical mechanics and has played a crucial role in the development of modern physics.

2. MOMENTUM CALCLATION

➢ What is momentum?

Momentum is a fundamental concept in physics that represents the quantity of motion possessed by an object. It is defined as the product of an object's mass and its velocity. The momentum of an object is a vector quantity, meaning it has both magnitude and direction.

Mathematically, momentum (p) is given by the formula:

$$\mathbf{p} = \mathbf{m} \times \mathbf{v} \qquad \textbf{[2]}$$

where:

- p is the momentum vector,

- m is the mass of the object,

- v is the velocity vector of the object.

The momentum vector points in the same direction as the velocity vector. The magnitude of the momentum vector is equal to the product of the mass and the magnitude of the velocity.

The SI unit of momentum is kilogram-meter per second (kg·m/s). In terms of basic dimensions, momentum has the dimensions of mass times velocity, which are $[M][L][T]^{-1}$.

Momentum is a conserved quantity in isolated systems, meaning the total momentum of a system remains constant unless acted upon by external forces. This conservation principle is known as the law of conservation of momentum and is particularly useful in analyzing collisions and interactions between objects.

It's worth noting that momentum is a classical concept and is valid within the framework of classical mechanics. In the context of relativistic physics, the concept of momentum is extended and modified in the theory of special relativity, accounting for the effects of high velocities and relativistic mass.

- **UPDATING MOMENTUM**

To update the momentum of an object based on the force acting on it, you can use Newton's second law of motion. Newton's second law states that the force acting on an object is equal to the rate of change of its momentum. Mathematically, it can be expressed as:

$$\mathbf{F} = \frac{\mathbf{dp}}{\mathbf{dt}} \qquad \textbf{[3]}$$

Where:

F is the force acting on the object,

dp is the change in momentum of the object, and

dt is the change in time.

From this equation, we can derive an expression to update the momentum of the object:

$$\mathbf{dp} = \mathbf{F} \times \mathbf{dt} \qquad \textbf{[4]}$$

This equation shows that the change in momentum (dp) of an object is equal to the force (F) acting on it multiplied by the change in time (dt).

To update the momentum, you need to know the initial momentum of the object and the force acting on it. The initial momentum ($p_{initial}$) can be given as the product of the object's mass (m) and its initial velocity ($v_{initial}$):

$$\mathbf{P_{initial}} = \mathbf{m} \times \mathbf{v_{initial}} \qquad \textbf{[5]}$$

Then, using the equation above, you can calculate the change in momentum (dp) and update the momentum (p_{final}) by adding the change in momentum to the initial momentum:

$$\mathbf{dp} = \mathbf{F} \times \mathbf{dt} \qquad \textbf{[6]}$$

$$\mathbf{p_{final}} = \mathbf{p_{initial}} + \mathbf{dp} \qquad \textbf{[7]}$$

Once you have the updated momentum, you can calculate the updated velocity by dividing the final momentum by the mass of the object:

$$\mathbf{v_{final}} = \frac{\mathbf{p_{final}}}{\mathbf{m}} \qquad \textbf{[8]}$$

This process allows you to update the momentum and velocity of an object based on the force acting on it over a certain time interval.

3. POSITION CALCULATION

To update the position of an object based on its momentum, you can use the concept of impulse. Impulse is defined as the change in momentum of an object and is equal to the force acting on the object multiplied by the time interval over which the force acts. Mathematically, it can be expressed as:

$$\textbf{Impulse } (\mathbf{J}) = \mathbf{F} \times \mathbf{\Delta t} \qquad \textbf{[9]}$$

Where:

J is the impulse,

F is the force acting on the object, and

Δt is the time interval over which the force acts.

By rearranging this equation, we can solve for the force:

$$\mathbf{F} = \frac{\mathbf{J}}{\Delta \mathbf{t}} \qquad [10]$$

Now, let's consider the momentum (p) of an object. The momentum is the product of the mass (m) and the velocity (v) of the object:

$$\mathbf{p} = \mathbf{m} \times \mathbf{v} \qquad [11]$$

If we want to update the position of the object, we need to consider the change in momentum (Δp). The change in momentum is given by:

$$\Delta \mathbf{p} = \mathbf{p}_{\mathbf{final}} - \mathbf{p}_{\mathbf{initial}} \qquad [12]$$

To update the position, we use the concept of average velocity. The average velocity (v_{avg}) of an object is defined as the change in position (Δx) divided by the time interval (Δt):

$$\mathbf{v}_{\mathbf{avg}} = \frac{\Delta \mathbf{x}}{\Delta \mathbf{t}} \qquad [13]$$

Rearranging this equation, we can solve for the change in position:

$$\Delta \mathbf{x} = \mathbf{v}_{\mathbf{avg}} \times \Delta \mathbf{t} \qquad [14]$$

Now, let's substitute the momentum equation into the equation for average velocity:

$$\mathbf{v}_{\mathbf{avg}} = \frac{\Delta \mathbf{p}}{(\mathbf{m} \times \Delta \mathbf{t})} \qquad [15]$$

Substituting this back into the equation for change in position, we get:

$$\Delta \mathbf{x} = \left(\frac{\Delta \mathbf{p}}{(\mathbf{m})}\right) \times \Delta \mathbf{t} \qquad [16]$$

This equation tells us that the change in position (Δx) of an object is equal to the change in momentum (Δp) divided by the mass (m) of the object.

To update the position, you can add this change in position to the initial position ($x_{initial}$) of the object:

$$\mathbf{x}_{\mathbf{final}} = \mathbf{x}_{\mathbf{initial}} + \Delta \mathbf{x} \qquad [17]$$

By performing these calculations, you can update the position of an object based on its momentum.

4. TIME CALCULATION

Updating the time based on a change in time is a straightforward process. When you have a known initial time and want to update it by a certain change in time, you can simply add the change in time to the initial time. Mathematically, it can be expressed as:

$$\mathbf{t}_{\mathbf{final}} = \mathbf{t}_{\mathbf{initial}} + \Delta \mathbf{t} \qquad [18]$$

Where:

t_{final} is the updated or final time,

$t_{initial}$ is the initial time, and

Δt is the change in time.

By adding the change in time to the initial time, you obtain the updated or final time value.

For example, let's say you have an initial time of $t_{initial}$ = 10 seconds, and you want to update the time by a change of Δt = 5 seconds. Applying the formula, you would have:

$$\mathbf{t}_{\mathbf{final}} = \mathbf{10\ seconds} + \mathbf{5\ seconds} = \mathbf{15\ seconds}$$

Therefore, the updated time would be **t_{final} = 15 seconds.**

This method is commonly used to update or calculate the time based on a given initial time and a known change in time.

5. ANGLE CALCULATION

To calculate an angle between two vectors using the dot product, you can use the inverse cosine function ($\cos^{-1}$). The dot product of two vectors is related to the cosine of the angle between them. The formula to calculate the dot product is:

$$\mathbf{A} \cdot \mathbf{B} = (|\mathbf{A}||\mathbf{B}|)\cos(\boldsymbol{\theta}) \qquad \textbf{[19]}$$

Where:

A and B are vectors,

|A| and |B| are the magnitudes (lengths) of vectors A and B, respectively,

θ is the angle between the two vectors.

Rearranging the equation, we have:

$$\cos(\boldsymbol{\theta}) = \frac{(\mathbf{A} \cdot \mathbf{B})}{(|\mathbf{A}||\mathbf{B}|)} \qquad \textbf{[20]}$$

To calculate the angle θ, you can use the inverse cosine function:

$$\boldsymbol{\theta} = \boldsymbol{cos}^{-1}\left(\frac{(\boldsymbol{A} \cdot \boldsymbol{B})}{(|\boldsymbol{A}||\boldsymbol{B}|)}\right) \qquad \textbf{[21]}$$

In this equation, (A · B) represents the dot product of vectors A and B, and |A| and |B| are the magnitudes (norms) of the vectors.

Using this formula, you can calculate the angle between two vectors based on their dot product and magnitudes. The resulting angle will be in radians. If you prefer the angle in degrees, you can convert it using the appropriate conversion factor (180°/π radians).

6. CALCULATION OF VELOCITY IN A TWO BODY SYSTEM:

To calculate the velocity in a two-body system, we need to know the masses of the two bodies and the distance between them. Additionally, we need to assume that there are no external forces acting on the system.

Let's denote the masses of the two bodies as m1 and m2, where m1 is the mass of the first body and m2 is the mass of the second body. The distance between them can be represented as r.

In a two-body system, the velocity of each body can be calculated using the concept of gravitational force. According to Newton's law of universal gravitation, the gravitational force between two bodies is given by:

$$\mathbf{F} = \mathbf{G} * \frac{(\mathbf{m_1} \times \mathbf{m_2})}{\mathbf{r^2}}$$

where G is the gravitational constant (approximately 6.674×10^{-11} N m^2 / kg^2).

Now, let's assume that the two bodies are moving in circular orbits around their common center of mass. In this case, the gravitational force provides the necessary centripetal force to keep each body in its orbit.

The centripetal force can be expressed as:

$$\mathbf{F} = \frac{(\mathbf{m_1} \times \mathbf{v_1}^2)}{\mathbf{r}} \qquad \textbf{[22]}$$

where v1 is the velocity of the first body.

Setting the gravitational force equal to the centripetal force, we can solve for the velocity:

$$\mathbf{G} \times \left(\frac{(\mathbf{m_1} \times \mathbf{m_2})}{\mathbf{r^2}}\right) = \frac{(\mathbf{m_1} \times \mathbf{v_1}^2)}{\mathbf{r}}$$

Simplifying the equation, we get:

$$\mathbf{v1} = \sqrt{\frac{(\mathbf{G} \times \mathbf{m_2})}{\mathbf{r}}} \qquad \textbf{[23]}$$

Similarly, for the second body:

$$v2 = \sqrt{\frac{(G \times m_1)}{r}} \qquad [24]$$

These equations give us the velocities v1 and v2 for the two bodies in the two-body system.

7. CALCULATION OF TIME PERIOD:

To derive the time period of an object in circular orbit based on the radius of the orbit, we can start with the centripetal force acting on the object.

In circular motion, the centripetal force is provided by the gravitational force between the object and the central body. According to Newton's law of universal gravitation, the gravitational force between two objects is given by:

$$F = G \times \frac{(m1 \times m2)}{r^2}$$

where F is the gravitational force, G is the gravitational constant, m1 and m2 are the masses of the two objects, and r is the distance between them.

For an object in circular orbit, the gravitational force provides the necessary centripetal force to keep the object in its orbit. The centripetal force is given by:

$$F = \frac{(m_1 \times {v_1}^2)}{r}$$

where m is the mass of the object and v is its velocity.

Setting the gravitational force equal to the centripetal force, we have:

$$G \times \left(\frac{(m_1 \times m_2)}{r^2}\right) = \frac{(m_1 \times {v_1}^2)}{r}$$

Simplifying the equation, we get:

$$v^2 = \frac{(G \times (m_1 + m_2))}{r} \qquad [25]$$

Now, we can consider the time period T, which is the time taken by the object to complete one full orbit. The time period is related to the velocity and the circumference of the orbit.

The circumference of the circular orbit is given by:

$$C = 2 \times \pi \times r \qquad [26]$$

The velocity v can be expressed as:

$$v = \frac{C}{T} \qquad [27]$$

Substituting the value of C and rearranging the equation, we get:

$$v^2 = \frac{(4 \times \pi^2 \times r^2)}{T^2} \qquad [28]$$

Equating this expression with the previous expression for v^2, we have:

$$\frac{(G \times (m_1 + m_2))}{r} = \frac{(4 \times \pi^2 \times r^2)}{T^2}$$

Simplifying further, we find:

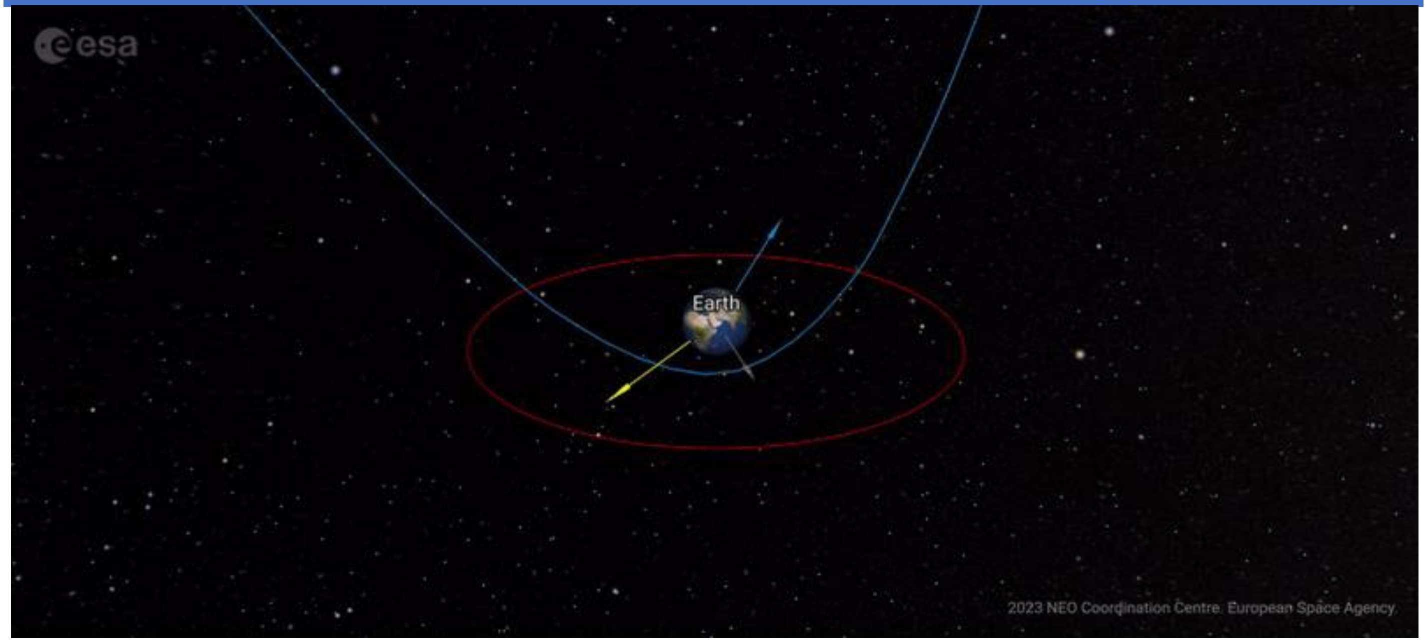
$$T^2 = \frac{(4 \times \pi^2 \times r^3)}{(G \times (m_1 + m_2)}$$

Taking the square root of both sides, we finally obtain the expression for the time period T in terms of the radius of the orbit r:

$$T = 2 \times \pi \times \sqrt{\frac{(r^3)}{(G \times (m_1 + m_2))}} \qquad [29]$$

This equation gives us the time period of an object in circular orbit as a function of the radius of the orbit and the masses of the objects involved.

• VISUALIZING THE NEAR-EARTH ASTEROID & CALCULATING DEFLECTION ANGLE



In this paper we going to visualize the movement of near-earth asteroid and calculate its deflection angle. As mentioned in [4] in the early hours of 27 January 2023, a tiny asteroid will safely fly by Earth 100 times closer than the moon, 10 times closer than satellites in the geostationary ring. Asteroid 2023 BU is three to eight metres in diameter, and as it comes just 3600 km from Earth's surface its path will be dramatically altered by our planet's gravity.2023 BU was discovered about a week ago. Although it doesn't seem like much warning, the advance detection of this very small – and safe – asteroid, shows just how much detection technologies are improving. The size and shape of the asteroid shown is not to scale. Stimulating asteroids near Earth can have several important implications and potential benefits.

Here are some reasons why stimulating asteroids near Earth is considered important:

1. **Scientific Research**
   Studying asteroids provides valuable insights into the early solar system and the formation of planets. By stimulating and studying asteroids near Earth, scientists can gather data on their composition, structure, and other characteristics. This research can enhance our understanding of the universe and the processes that shaped our own planet.
2. **Planetary Defense**
   Some asteroids pose a potential threat to Earth if their orbits bring them close to our planet. By studying and tracking these near-Earth asteroids (NEAs), we can identify potentially hazardous objects and develop strategies for mitigating the risk of a collision. Stimulating these asteroids allows us to gather more accurate data about their trajectories and properties, aiding in the development of effective asteroid deflection and mitigation techniques.
3. **Resource Exploration**
   Asteroids are rich in valuable resources such as metals, water, and organic compounds. By stimulating and exploring near-Earth asteroids, we can assess their resource potential and determine if they could serve as future mining targets. Utilizing asteroid resources could reduce the need for Earth-based mining and provide valuable resources for space exploration and colonization.

4. **Human Space Exploration**

   Studying and stimulating asteroids near Earth can contribute to the advancement of human space exploration. Asteroids can serve as potential destinations for future crewed missions, providing opportunities for scientific research, testing technologies, and even serving as stepping stones for deeper space exploration. Understanding asteroids and their properties is crucial for planning and executing safe and successful manned missions.

5. **Commercial Opportunities**

   The exploration and utilization of near-Earth asteroids can also create economic opportunities. Companies and organizations may develop asteroid mining operations or offer asteroid-based services, such as resource extraction, space manufacturing, or scientific research. This emerging sector could drive technological advancements, foster innovation, and potentially contribute to the space economy.

   Overall, stimulating asteroids near Earth offers scientific, planetary defense, resource exploration, human space exploration, and commercial opportunities. It expands our knowledge of the solar system, helps protect our planet from potential asteroid impacts, and paves the way for future space exploration and utilization.

1. PROGRAMING ALGORITHM

Break the program into small time steps

1) Find the position vector r (position vector between asteroid and earth).
2) Calculate the gravitational force F_G using r.

$$\mathbf{F_G = G * \frac{(m1 * m2)}{r^2}}$$

3) Calculate/Update momentum using F_G.

$$\mathbf{p_{final} = p_{initial} + F_G \times dt}$$

4) Find the new position using the updated momentum.

$$\mathbf{x_{final} = x_{initial} + \left(\frac{\Delta p}{(m)}\right) \times \Delta t}$$

5) Update the time.
6) Repeat the steps from 1 to 5 for a particular time period.
7) Calculate the final velocity of asteroid using mass and momentum.

$$\mathbf{V_{final} = \frac{P_{final}}{m}}$$

8) Calculate the deflection angle using dot product.

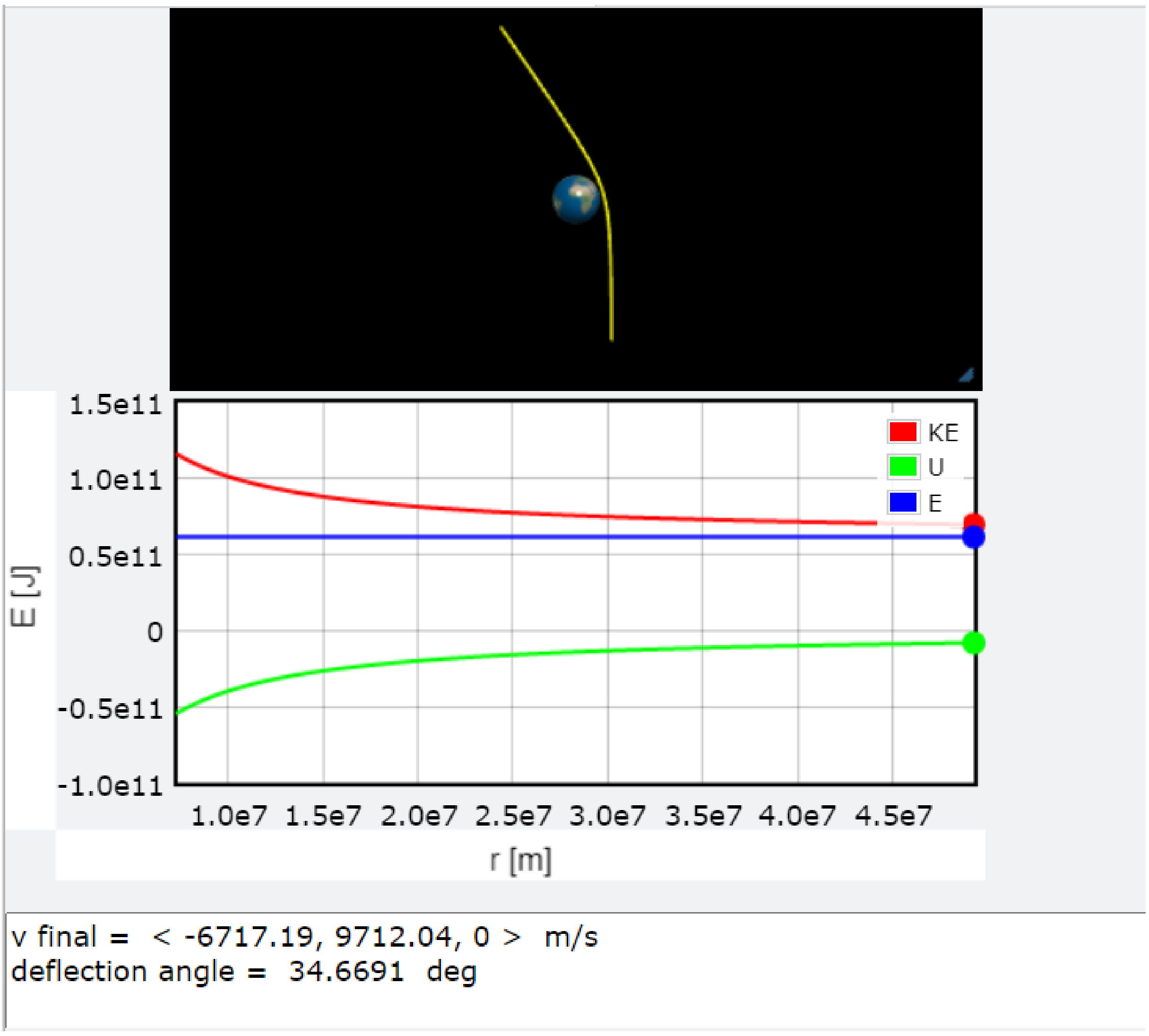
$$\boldsymbol{\theta} = \mathbf{cos^{-1}\left(\frac{(A \cdot B)}{(|A|\ |B|)}\right)}$$

2. PYTHON CODE

```
from visual import*
#Constants used in the program
G=6.67e-11
ME=5.97e24
RE=6.37e6
RQ=RE+2.95e6
v0=12e3
mQ=1000
#making graphs
g1 = graph(xtitle="r [m]", ytitle="E [J]", width=500, height=250)
fK = gcurve(color=color.red, label="KE", dot=True)
fU = gcurve(color=color.green, label="U", dot=True)
fE = gcurve(color=color.blue, label = "E", dot=True)
#earth model
earth=sphere(pos=vector(0,0,0),radius=RE, material=materials.earth)
#asteroid model
QG=sphere(pos=vector(RQ,-4*RQ,0),radius=RE/10,color=color.yellow,make_trail=True)
earth.m=ME
QG.m=mQ
#Assinging initial momentum to the asteroid
QG.p=QG.m*vector(0,v0,0)
t=0
#Break the program into small time steps
dt=1
#Repeat the steps from 1 to 5 for a particular time period
while t<70000:
    rate(10000)
#Finding the position vector r (position vector between asteroid and earth)
    r=QG.pos-earth.pos
#Calculating the gravitational force FG using r
    F=-G*earth.m*QG.m*norm(r)/mag(r)**2
#Calculating/Updating momentum using FG
    QG.p=QG.p + F*dt
#Find the new position using the updated momentum
    QG.pos=QG.pos+QG.p*dt/QG.m
#Calculating energies for graph
     K = mag(QG.p)**2/(2*QG.m)  #mag(QG.p)**2/(2*QG.m)
     U = -(G*ME*QG.m)/mag(r)
```

```
E = K + U
#Plotting energy values on graph
fK.plot(pos=(mag(r),K))
fU.plot(pos=(mag(r),U))
fE.plot(pos=(mag(r),E))
#Updating the time
t=t+dt
#Calculating the final velocity of asteroid using mass and momentum
print("v final = ",QG.p/QG.m," m/s")
vQG0=vector(0,v0,0)
#Calculating the angle of deflection
theta=acos(dot(vQG0,QG.p/QG.m)/(mag(vQG0)*mag(QG.p/QG.m)))
print("deflection angle = ",theta*180/pi," deg")
```

3. OUTPUT



```
v final =  < -6717.19, 9712.04, 0 >  m/s
deflection angle =  34.6691  deg
```

4. CODE EXPLAINED WITH SYNTAX

1) Physical Constants used in the program

a) Gravitational constant: Program statement 2

The gravitational constant, frequently represented by the letter "G," is a fundamental physical constant that can be found in both Albert Einstein's general relativity theory and Sir Isaac Newton's law of universal gravitation. It serves as an indicator of how strong the gravitational pull is between two objects. Approximately **6.67430×10^{-11}m^3/kg s^2** is the gravitational constant's value.

It's important to remember that the gravitational constant is a fundamental constant of nature, and that experiments and measurements have been used to determine the exact value of this constant. It plays a crucial role in understanding the behaviour of gravitational forces on both small and large scales systems.

Assigning value of gravitational constant:

Real value of G=**6.67430×10^{-11}m^3/kg**

Python syntax: G=6.67e-11

G is the variable assigned to represent Gravitational constant

b) Mass of Earth: Program statement 3

The mass of the Earth is approximately **5.972×10^{24} kg.** This value represents the total amount of matter contained within the Earth, including its core, mantle, and crust. The mass of the Earth is a fundamental property that determines its gravitational pull on objects around it, including the moon and other satellites that orbit the planet.

Assigning value of Mass of earth

Real value: Mass of earth=5.972×10^{24} kg

Python syntax: ME=5.97e24

ME is a variable that represents mass of earth

c) Radius of Earth: Program statement 4

It is important to note that the Earth is not a perfect sphere but rather an oblate spheroid, meaning its shape is slightly flattened at the poles and bulging at the equator due to its rotation. The equatorial radius is about 21 kilometers (13 miles) larger than the polar radius, which contributes to the Earth's oblate shape.The mean radius of the Earth is approximately **6,371 kilometers** (3,959 miles). This value represents the average distance from the center of the Earth to its surface.

Assigning value of Radius of earth

Real value: Radius of earth=6.371×10^6 m

Python syntax: RE=6.37e6

RE is a variable that represents Radius of earth

d) Position of asteroid: Program statement 5

The positions of asteroids can vary greatly over time as they move along their orbits around the Sun. But for the stimulation we have to assign an initial position of the asteroid.

Assigning initial position of asteroid

Position of asteroid (RQ) = (RE) radius of earth + 2950000 meters

Python syntax: RQ=RE+2.95e6

RQ is a variable that represents the initial position of asteroid, the direction of the position will be assigned later.

e) **Initial Velocity of Asteroid: Program statement 6**

The velocity of an asteroid can vary depending on its orbit, distance from the Sun, and other factors. We are here assigning a random initial value to the velocity of asteroid.

Assigning initial velocity of asteroid

$V_{initial}$=12000m/s

Python syntax: v0=12e3

v0 is a variable that represents the initial velocity of asteroid

f) **Mass of Asteroid: Program statement 7**

Asteroids can vary widely in terms of their mass, ranging from very small rocky fragments to significantly larger bodies. The mass of most asteroids falls within a wide range, from a few kilograms to potentially several hundred billion kilograms. The smallest asteroids, known as meteoroids, can be as small as a few grams or even smaller. On the other end of the spectrum, the largest known asteroid, Ceres, which is classified as a dwarf planet, has a mass of approximately **9.39×10^{20} kilograms**. It's important to note that the mass of an asteroid is a key factor in determining its gravitational influence, structural characteristics, and behaviour in space. Precise measurements of the masses of individual asteroids are often challenging and rely on various methods such as gravitational interactions, spacecraft observations, and radar imaging. Here we will be assigning a mass of 1000 kg to the asteroid.

Assigning initial velocity of asteroid

$M_{asteroid}$=1000 kilograms

Python syntax: mQ=1000

mQ is a variable that represents the mass of asteroid

2) **Making Graphs:**

In the GlowScript programming environment, you can create graphs and plots using the `graph`, `gcurve`, and `plot` objects. These objects allow you to visualize data in a 2D graph.

1. **`graph` object:** It represents the canvas on which you can draw your graph. You can create a graph object using the following syntax:

   Python Syntax:graph_object = graph(title='Graph Title', xtitle='X-axis Label', ytitle='Y-axis Label')

   The `title`, `xtitle`, and `ytitle` arguments are optional and can be used to set the graph's title, x-axis label, and y-axis label, respectively.

2. **`gcurve` object:** It represents a curve on the graph. You can create a `gcurve` object and add points to it using the `plot` method. Here's an example:

   Python Syntax: curve_object = gcurve(color=color.red) #Create a red curve

   ```
   # Add points to the curve
   curve_object.plot(x_value1, y_value1)
   curve_object.plot(x_value2, y_value2)
   ```

   You can specify the color of the curve using the `color` parameter e.g. `color.red`, `color.blue` etc.

3. **`plot` function:** Alternatively, you can use the `plot` function to create and plot data directly on the graph. Here's an example:

```
graph_object = graph(title='Graph Title', xtitle='X-axis Label', ytitle='Y-axis Label')
# Plot data using the plot function
plot(x_data, y_data, color=color.green)
```

   The `x_data` and `y_data` arguments should be lists or arrays containing the x and y values, respectively. You can also specify the color of the plot using the `color` parameter.

Here's a complete example that demonstrates creating a graph and plotting data:

```
from vpython import *
# Create the graph
graph_object = graph(title='Graph Example', xtitle='X-axis', ytitle='Y-axis')
# Create a curve and plot some points
curve_object = gcurve(color=color.red)
curve_object.plot(0, 0)
curve_object.plot(1, 1)
curve_object.plot(2, 4)
# Create another plot using the plot function
x_data = [0, 1, 2, 3, 4]
y_data = [0, 1, 4, 9, 16]
plot(x_data, y_data, color=color.blue)
```

This code will create a graph titled "Graph Example" with the x-axis labeled "X-axis" and the y-axis labeled "Y-axis". It will also create a red curve and plot the points (0, 0), (1, 1), and (2, 4). Additionally, a blue plot will be created using the `plot` function with the provided data.

Program statement 8,9,10,11

Python Syntax: g1 = graph(xtitle="r [m]", ytitle="E [J]", width=500, height=250)

Make a graph with distance (r) on x-axis and Total energy on y-axis

```
fK = gcurve(color=color.red, label="KE", dot=True)
fU = gcurve(color=color.green, label="U", dot=True)
fE = gcurve(color=color.blue, label = "E", dot=True)
```

Marking curve on 2D-plane for Kinetic energy , Potential energy and Total energy

With red, green, blue colours respectively.

Program statement 25,26,27

Python Syntax: #Calculating energies for graph

```
K = mag(QG.p)**2/(2*QG.m)  #mag(QG.p)**2/(2*QG.m)
U = -(G*ME*QG.m)/mag(r)
E = K + U
```

K is the variable that represents Kinetic Energy

U is the variable that represents Potential Energy

E is the variable that represents Total Energy

Program statement 28,29,30

Python Syntax: #Plotting energy values on graph

```
fK.plot(pos=(mag(r),K)) #plotting Kinetic energy
fU.plot(pos=(mag(r),U)) #plotting Potential energy
fE.plot(pos=(mag(r),E)) #plotting Total energy
```

Plotting points on the respective curves on the graph

3) Sphere function:

Here is an example of how to make a sphere:

ball = sphere (pos= vector (1,2,1), radius=0.5)

This produces a sphere centered at location (1,2,1) with radius = 0.5, with the current foreground color.

The sphere object has the following attributes and default values, similar to cylinder: pos vector(0,0,0), axis vector(1,0,0), color vector(1,1,1) which is color.white, red (1), green (1), blue (1), opacity (1), shininess (0.6), emissive (False), **texture**, and up vector(0,1,0). As with cylinders, up and axis attributes affect the orientation of the sphere but have only a subtle effect on appearance unless a non-smooth texture is specified or the cross section is oval. The magnitude of the axis attribute is irrelevant. Additional sphere attributes:

radius. The radius of the sphere, default = 1.

size Default is vector(2,2,2). Instead of specifying the radius, you can set size=vector(length ,height ,width), which means that the cross section of the sphere can be elliptical, making it like the ellipsoid object. Unlike other objects, changing size doesn't change axis, changing axis doesn't change size.

canvas By default, an object such as a sphere will be displayed in the most recently created 3D canvas, which will be the default canvas named "scene" unless you create a canvas yourself.

Note that the pos attribute for cylinder, arrow, cone, and pyramid corresponds to one end of the object, whereas for a sphere it corresponds to the center of the object.

If you include make_trail=True when you create the object, a trail will be left behind the object as you move it. For related options, see **Leaving a Trail**.

Making 3d model of earth and asteroid: Program statement 12,13

Python syntax:

earth=sphere(pos=vector(0,0,0), radius=RE ,material=materials.earth)

Making a sphere at origin with radius of earth and giving it earth's texture using materials.earth

Python syntax:

```
QG=sphere(pos=vector(RQ,-4*RQ,0),radius=RE/10,color=color.yellow,make_trail=True)
```

Making a sphere at (RQ,-4*RQ,0) in 3d space with radius 1/10 of earth and giving it yellow colour and switching on trail property

4) Calculating initial momentum of asteroid

Momentum is a fundamental concept in physics that represents the quantity of motion possessed by an object. It is defined as the product of an object's mass and its velocity. The momentum of an object is a vector quantity, meaning it has both magnitude and direction.

Mathematically, momentum (p) is given by the formula: p = m * v

where:

- p is the momentum vector,

- m is the mass of the object,

- v is the velocity vector of the object.

The momentum vector points in the same direction as the velocity vector. The magnitude of the momentum vector is equal to the product of the mass and the magnitude of the velocity.

Momentum is a conserved quantity in isolated systems, meaning the total momentum of a system remains constant unless acted upon by external forces. This conservation principle is known as the law of conservation of momentum and is particularly useful in analyzing collisions and interactions between objects.

a. Vector function in glowscript

In GlowScript, the `vector()` function is a built-in function that creates and returns a vector object. Vectors are fundamental data types in GlowScript used to represent quantities with both magnitude and direction, such as positions, velocities, forces, and more.

The `vector()` function can be used in several ways

1. Creating a vector with specified components

```
v = vector(x, y, z)
```

Here, `x`, `y`, and `z` are the components of the vector, representing its coordinates in 3D space.

2. Creating a vector from another vector

```
v2 = vector(v1)
```

This creates a new vector `v2` with the same components as `v1`. It can be useful for making copies or performing vector operations.

3. Creating a vector from a list or tuple

```
v = vector([x, y, z])  # or vector((x, y, z))
```

This syntax allows you to create a vector by passing a list or tuple containing the components.

Once you have created a vector, you can perform various operations on it, such as addition, subtraction, scalar multiplication, dot product, cross product, and more.

Here's an example that demonstrates the usage of the **`vector()`** function and some vector operations in GlowScript:

```
from visual import*

# Creating vectors

v1 = vector(1, 2, 3)

v2 = vector(4, 5, 6)

# Addition and subtraction
```

```
v_sum = v1 + v2
v_diff = v1 - v2
# Scalar multiplication
v_scaled = 2 * v1
# Dot product
dot_product = v1.dot(v2)
# Cross product
cross_product = v1.cross(v2)
# Printing the results
print(v_sum)  # Output: <5, 7, 9>
print(v_diff)  # Output: <-3, -3, -3>
print(v_scaled)  # Output: <2, 4, 6>
print(dot_product)  # Output: 32
print(cross_product)  # Output: <-3, 6, -3>
```

In the example above, `v1` and `v2` are two vectors created using the `vector()` function. The code then performs addition, subtraction, scalar multiplication, dot product, and cross product operations on these vectors, producing the respective results.

Assigning initial momentum to the asteroid: Program Statement 16

$P_{ASTEROID} = M_{ASTEROID} \times V_{ASTEROID}$

Python syntax : QG.p=QG.m*vector(0,v0,0)

QG.p is the variable that represents momentum of asteroid,

QG.m represents the variable that represents the mass of the asteroid,

vector(0,v0,0) assigning direction to the velocity of asteroid in 3d plane

5) Find the position vector r (position vector between asteroid and earth)

The `pos` function is used to set or retrieve the position of an object in a 3D scene. It is commonly used to manipulate the position of 3D objects such as spheres, boxes, or other geometric shapes.

The `pos` function can be used in two different ways:

- **EXAMPLE:**

1. Setting the position:

To set the position of an object, you would use the `pos` function with an assignment operator (`=`) followed by the desired position coordinates. The position is specified as a vector with three components: `pos(x, y, z)’. For example, to set the position of an object called `myObject` to the coordinates (1, 2, 3), you would write:

Python syntax : myObject.pos = vector(1, 2, 3)

2. Retrieving the position:

To retrieve the current position of an object, you can simply use the `pos` function without the assignment operator. For example, to retrieve the position of `myObject`, you would write:

Python syntax: objectPosition = myObject.pos

The variable `objectPosition` will then store the vector representing the current position of `myObject`.

a) Finding the position vector between asteroid and earth: Program Statement 21

To calculate the position vector between two points in three-dimensional space, you can subtract the coordinates of one point from the coordinates of the other point. The result will be a vector representing the displacement or position between the two points. Let's assume you have two points, Point A and Point B, with coordinates (x1, y1, z1) and (x2, y2, z2) respectively. The position vector (P) between Point A and Point B is calculated as:

P = B - A

This means subtracting the coordinates of Point A from the coordinates of Point B component-wise. The resulting vector will have components (x2 - x1, y2 - y1, z2 - z1).

Position vector between asteroid and earth = Position vector of asteroid – Position vector of earth

Python syntax : r=QG.pos-earth.pos

r is a variable that represents the Position vector between asteroid and earth

6) Calculate the gravitational force F_G using r: Program Statement 22

The universal law of gravitation, formulated by Sir Isaac Newton, describes the gravitational force between two objects. In vector form, the law can be expressed as:

$$\mathbf{F} = -\mathbf{G} \times \frac{(\mathbf{m_1} \times \mathbf{m_2})}{\mathbf{r^2}} \hat{\mathbf{r}}$$

where:

- F is the gravitational force vector between the two objects,

- G is the gravitational constant,

- m_1 and m_2 are the masses of the two objects,

- r is the separation vector between the two objects,

- $\hat{r}$ is the unit vector in the direction of r.

$$\textbf{Force calculation: } \mathbf{F} = -\mathbf{G} \times \frac{(\mathbf{m_1} \times \mathbf{m_2})}{\mathbf{r^2}} \hat{\mathbf{r}}$$

Universal law of Gravitation

Python syntax : F=-G*earth.m*QG.m*norm(r)/mag(r)**2

F is the variable that represents the gravitational force between earth and asteroid

earth.m , QG.m represents mass of earth and asteroid

norm(r) represents the unit vector of r and mag(r) represents the magnitude of r

7) Calculate/Update using Momentum: Program Statement 23

When updating momentum using force, you typically apply the principles of classical mechanics. The momentum of an object is defined as the product of its mass and velocity. The force acting on an object causes a change in its momentum according to Newton's second law, which states that the force is equal to the rate of change of momentum.

Mathematically, the relationship between force, momentum, and time is given by the equation:

$$\mathbf{F} = \frac{\mathbf{dp}}{\mathbf{dt}}$$

where F represents the force, p is the momentum, and t is the time.

To update momentum using force, you need to consider the time interval over which the force is applied. If the force is constant, you can use the following equation to update momentum:

$$\mathbf{p}_{\mathbf{new}} = \mathbf{p}_{\mathbf{initial}} + \mathbf{F} \times \mathbf{dt}$$

where p_{new} is the updated momentum, $p_{initial}$ is the initial momentum, F is the force, and dt is the time interval over which the force is applied.

In practice, when dealing with continuous forces or forces that vary over time, you may need to integrate the force over small time intervals to obtain an accurate update for momentum. This process involves dividing the total time interval into smaller increments and updating the momentum incrementally.

It's important to note that the above equations assume a constant mass. If the mass of the object changes during the force application, you'll need to consider the appropriate equations for mass variation as well.

Keep in mind that this explanation is based on classical mechanics and assumes a simplified scenario. In more complex situations or when considering relativistic effects, additional equations and considerations may be required.

$\mathbf{p_{new} = p_{initial} + F \times dt}$

Python syntax : QG.p=QG.p + F*dt

QG.p and QG.m are the variable that represent the momentum and mass of asteroid

F and dt are the variable that represents the gravitational force and small change in time

8) Finding the new position using the updated momentum: Program statement 24

To update the position of an object based on its momentum, you can use the concept of impulse. Impulse is defined as the change in momentum of an object and is equal to the force acting on the object multiplied by the time interval over which the force acts. Mathematically, it can be expressed as:

$$\mathbf{\Delta x = \left(\frac{\Delta p}{(m)}\right) \times \Delta t}$$

This equation tells us that the change in position (Δx) of an object is equal to the change in momentum (Δp) divided by the mass (m) of the object.

To update the position, you can add this change in position to the initial position ($x_{initial}$) of the object:

$$\mathbf{x_{final} = x_{initial} + \Delta x}$$

By performing these calculations, you can update the position of an object based on its momentum.

$\mathbf{x_{final} = x_{initial} + \left(\frac{\Delta p}{(m)}\right) \times \Delta t}$

Python syntax : QG.pos=QG.pos+QG.p*dt/QG.m

QG.p,QG.pos and QG.m are the variable that represent the momentum, position and mass of asteroid

dt is the variable that represents the small change in time

9) Updating the time: Programming statement 31

Time Update formula $\mathbf{t_{new} = t_{initial} + dt}$

Python syntax : $t = t + dt$

t is the variable that represents time

dt is the variable that represents the small change in time

10) Repeat the steps from 3 to 7 for a particular time period: **Program statement 19**

WHILE LOOP: A while loop is a control flow structure that allows you to repeatedly execute a block of code as long as a specified condition is true. The loop will continue until the condition evaluates to false. Here's a breakdown of the key components and execution process of a while loop:

1.**Condition:**The condition is an expression that determines whether the loop should continue or not. It is evaluated before each iteration of the loop. If the condition is true, the code block inside the loop is executed. If the condition is false, the loop is terminated, and the program continues with the next line of code after the loop.

2. **Code block:** The code block is a set of statements that are executed within the loop if the condition is true. It can consist of any valid Python code, including assignments, function calls, conditional statements, and more. Indentation is crucial in Python to define the code block. Typically, four spaces or a tab is used to indent the statements consistently.

3. Iteration: The loop iterates or repeats until the condition becomes false. After executing the code block, the program goes back to the condition and checks whether it is still true. If the condition is true, the process is repeated, executing the code block again. This iteration continues until the condition becomes false.

4. Updating the condition: It's essential to ensure that the condition within the while loop will eventually become false. Otherwise, you may end up with an infinite loop that continues indefinitely. Within the loop, you usually need to update the variables or expressions used in the condition to ensure that it will eventually evaluate to false.

Here's an example to illustrate the execution of a while loop:

```
count = 0
while count < 5:
   print("Count:", count)
   count += 1
print("Loop finished.")
```

In this example, the variable `count` is initialized to 0. The condition `count < 5` is evaluated, and since it is true, the code block inside the loop is executed. It prints the current value of `count` and then increments it by 1 using the `count += 1` statement. After the code block executes, the program goes back to the condition, checks if it is still true, and repeats the process. This continues until `count` reaches 5. At that point, the condition `count < 5` evaluates to false, and the loop terminates. The program then proceeds to the next line after the loop and prints "Loop finished."

While loops are useful when you want to repeat a block of code based on a condition that may change during the execution of the loop. They provide flexibility and control over how many times the code is executed, allowing you to solve a wide range of problems.

Python syntax : while t<7000

The loop will be executed until the value of t is less than 7000

11) Calculating the final velocity of asteroid using mass and momentum: **Program statement 32**

Mathematically, momentum (p) is given by the formula: $p = m \times v$

Python syntax : print("v final = ",QG.p/QG.m," m/s")

QG.p/QG.m corresponds to $V_{final} = P_{final} / m$

12) Calculating the deflection angle using dot product: **Program statement 33,34**

Mathematically, calculation of angle using dot product is given by the formula:

$$\boldsymbol{\theta} = \mathbf{cos}^{-1}\left(\frac{(\mathbf{A}\ \cdot\ \mathbf{B})}{(|\mathbf{A}|\ |\mathbf{B}|)}\right)$$

Python syntax: vQG0=vector(0,v0,0)

vQG0 corresponds to the variable that denotes velocity of asteroid in a particular direction, vector(0,v0,0) assigning direction to the velocity of asteroid in 3d plane

Python syntax:theta=acos(dot(vQG0,QG.p/QG.m)/(mag(vQG0)*mag(QG.p/QG.m)))

theta corresponds to the variable that denotes angle of deflection ,QG.p and QG.m are the variable that represent the momentum and mass of asteroid, dot(vQG0,QG.p/QG.m) **corresponds to the dot product between vQG0 and** QG.p/QG.m (**final velocity of asteroid),** mag(QG.p/QG.m) **corresponds to the magnitude of final velocity of asteroid.**

• REPLICATING DART: DOUBLE ASTEROID REDIRECTION TEST

▪ Introduction:

Planetary defence encompasses all the capabilities needed to detect and warn of potential asteroid or comet impacts with Earth, and then either prevent them or mitigate their possible effects. Near-Earth objects (NEOs) are asteroids and comets that orbit the Sun like the planets, but with orbits that bring them into or through a zone between approximately 91 million and 121 million miles (195 million kilometres) from the Sun, meaning that they can pass within about 30 million miles (50 million kilometres) of Earth's orbit. Planetary defence is "applied planetary science" to address the NEO impact hazard.

▪ Double Asteroid Redirection Test Mission: [5]

The Johns Hopkins Applied Physics Laboratory (APL) in Laurel, Maryland developed and oversaw the Double Asteroid Redirection Test (DART), which is the first human effort to alter an asteroid's motion in space by purposefully colliding a spacecraft with it. The target asteroid for DART is not a threat to Earth, but it is the ideal test case to determine whether the kinetic impactor technique, a method of asteroid deflection, would be a practical way to defend our planet in the event that an asteroid on a collision course with Earth were to be discovered in the future. On September 26, 2022, at 7:14 p.m. ET, DART is planned to collide with its target asteroid Dimorphos, a satellite of the asteroid Didymos.

▪ DART Impactor [5]

DART is a budget-friendly spacecraft. The spacecraft's primary structure is a box with approximate measurements of 1.2 x 1.3 x 1.3 metres (3.9 x 4.3 x 4.3 feet), from which other structures extend to produce measurements of approximately 1.8 x 1.9 x 2.6 metres (5.9 x 6.2 x 8.5 feet) in width, length, and height. The spacecraft has two enormous solar arrays, each measuring 8.5 metres (27.9 feet) in length when fully extended. DART travelled at a speed of roughly 6.1 kilometres per second (3.8 miles per second) as it crashed into Dimorphos. At launch, the DART spacecraft weighed about 1,345 pounds (610 kilogrammes), and at impact, it weighed about 1280 pounds (580 kilogrammes). DART carries both xenon (about 130 pounds, or 60 kilogrammes) to power the ion propulsion technology demonstration engine and hydrazine propellant (about 110 pounds, or 50 kilogrammes) for spacecraft manoeuvres and attitude control.

Key spacecraft details include: [5]

- **Solar Electric Propulsion:** The spacecraft is powered by solar panels, providing the necessary electricity for propulsion and onboard systems.
- **Autonomous Navigation:** DART utilizes advanced onboard sensors, cameras, and an imaging system for autonomous navigation and targeting.
- **Kinetic Impactor:** The spacecraft is equipped with a high-velocity impactor, referred to as the DART impactor, which will be released before the final approach to the asteroid. The impactor is a solid mass weighing approximately 500 kg and will collide with Didymoon at a velocity of about 6 km/s.
- **Communication Systems:** DART communicates with Earth using high-gain antennas and the Deep Space Network, relaying data, images, and telemetry.
- **CubeSat:** DART will deploy a small CubeSat called LICIA (Light Italian CubeSat for Imaging of Asteroids) developed by the Italian Space Agency (ASI). LICIA will capture images of the impact event and provide additional data to complement DART's observations.

- **Didymos—The Ideal Target for DART's Mission** [6]

The Greek word Didymos, which translates to "twin" and explains why the mission's name contains the word "double," refers to the binary asteroid system Didymos. Didymos is the ideal candidate for humanity's first planetary defence experiment, even though the planet is not in danger of being struck by it and is therefore not in danger from it [7]. Two asteroids make up the system: Didymos, a larger asteroid with a diameter of 760 metres (0.47 miles), and Dimorphos, a moon-like asteroid with a diameter of 150 metres (492 feet), which orbits Didymos. The two asteroids' centres are separated by 1.2 kilometres (0.74 miles). Dimorphos orbital period around Didymos was 11 hours, 55 minutes prior to DART's kinetic impact. Dimorphos orbit around Didymos was shortened by 33 minutes as a result of the nearly head-on collision with the small asteroid moonlet caused by the DART spacecraft.

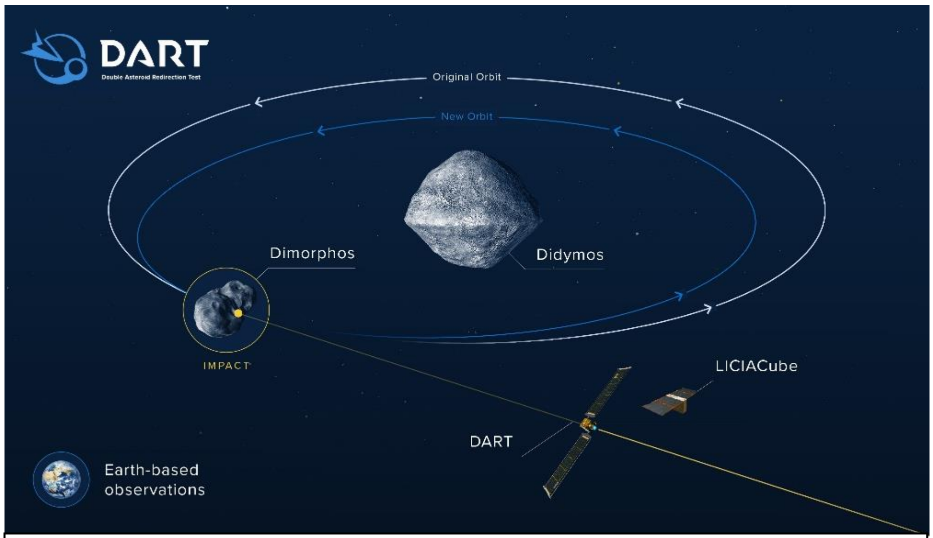


Illustration of how DART's impact altered the orbit of Dimorphos about Didymos. Telescopes on Earth are used to measure the change in the orbit of Dimorphos to evaluate the effectiveness of the DART impact.

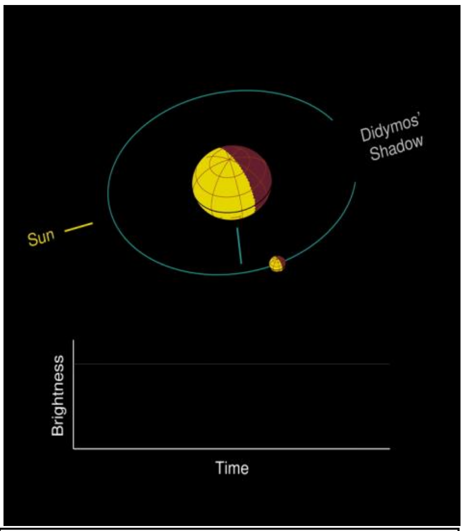
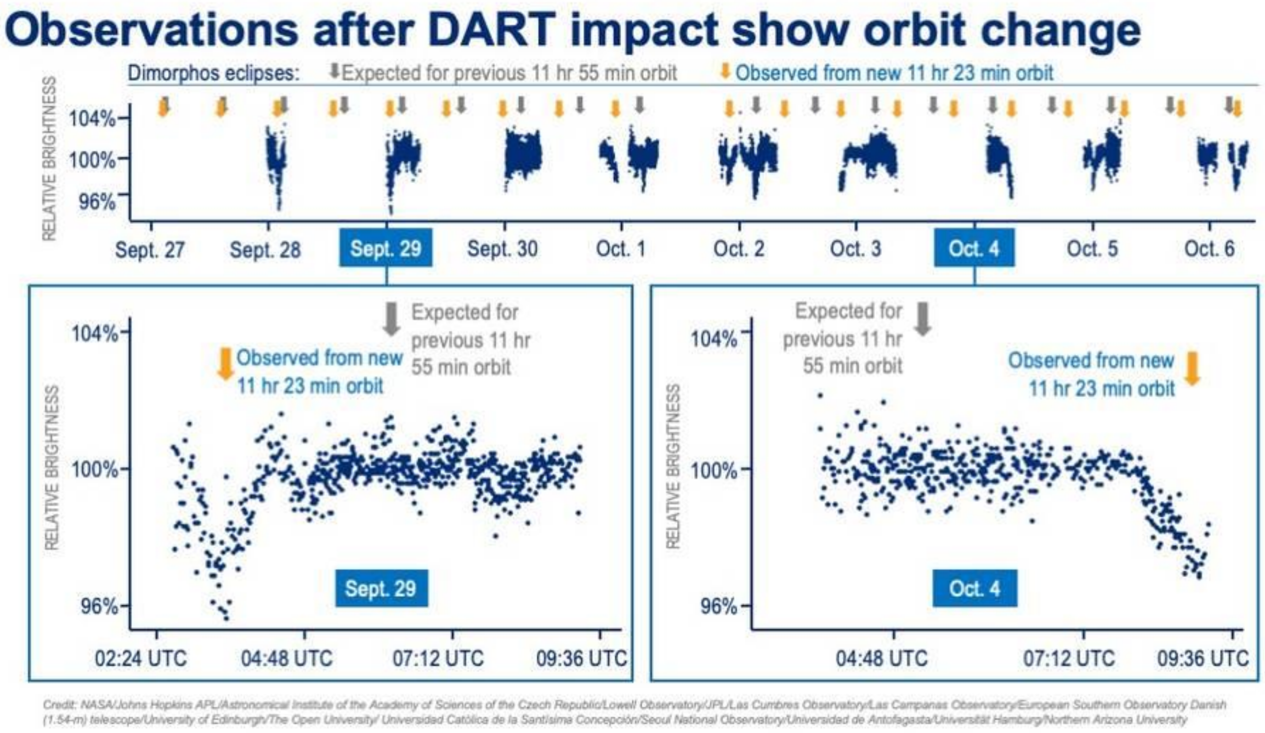
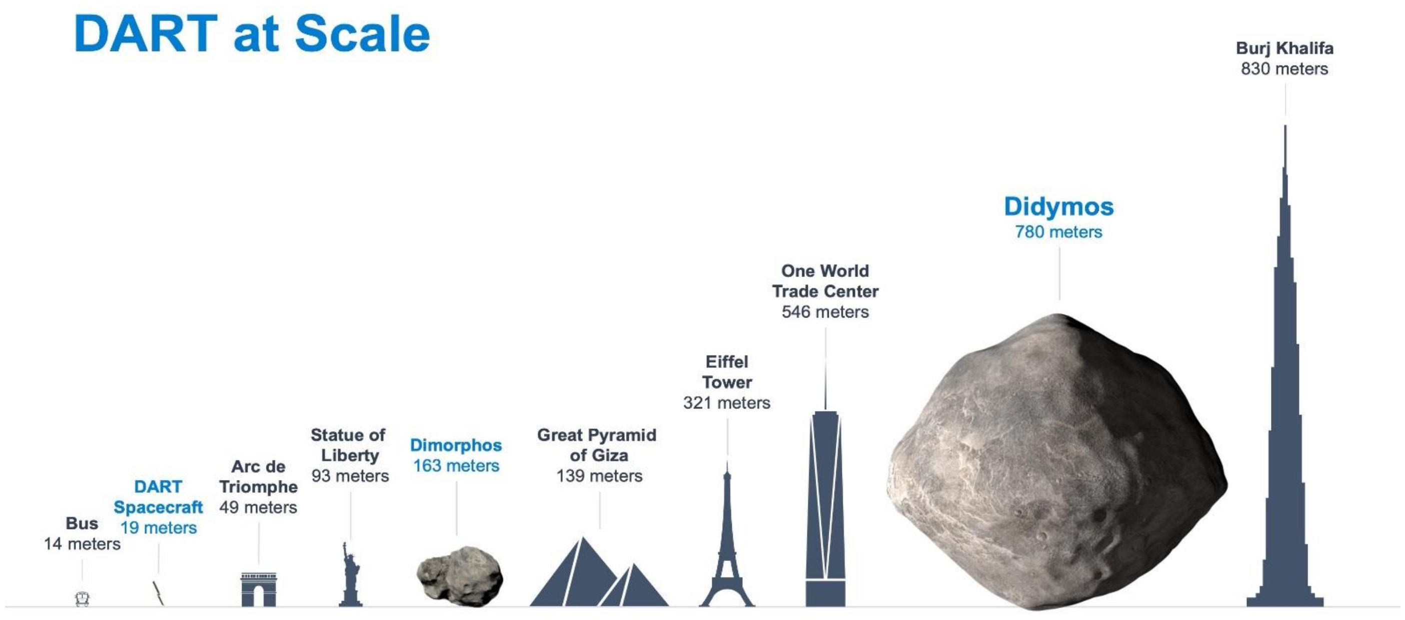
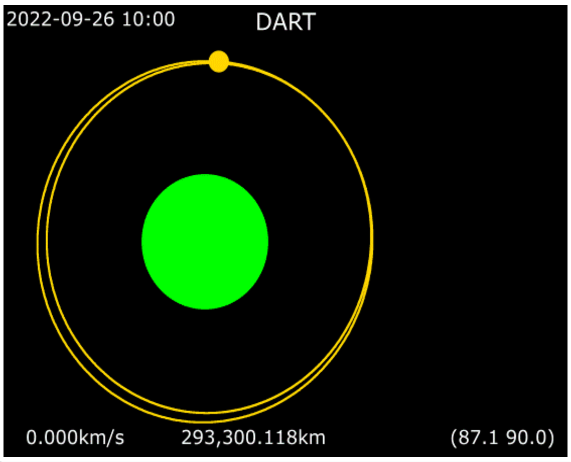


Illustration of Dimorphos orbit time period.





As seen from Earth, the Didymos system is an eclipsing binary, which means that as Dimorphos orbits the larger asteroid, Didymos passes in front of and behind it. In order to determine Dimorphos' orbit, telescopes on Earth can measure the regular variation in brightness of the combined Didymos system. To allow for the best possible telescopic observations, the DART impact was timed for September 2022, when the distance between Earth and Didymos was at its shortest. At the time of the DART impact, Didymos was still about 11 million kilometres (7 million miles) from Earth, but telescopes all over the world have contributed to the global international observing campaign to ascertain the impact's effects.



Animation of DART around Didymos - Impact on Dimorphos
🟪 DART · 🟩 Didymos · 🟨 Dimorphos

The DART demonstration is a careful and effective design. The DART demonstration is a careful and effective design. According to current predictions, Didymo's orbit does not cross that of Earth at any point, and the energy impulse that DART sent to Dimorphos was small and did not disturb the asteroid. At the time of its kinetic collision with Dimorphos, the DART spacecraft weighed about 580 kilogrammes (1280 pounds). Dimorphos mass has not been directly measured, but it is estimated to be about 5 billion kgs using assumptions about the asteroid's density and size. The DART demonstration is a careful and effective design. According to current predictions, Didymo's orbit does not cross that of Earth at any point, and the energy impulse that DART sent to Dimorphos was small and did not disturb the asteroid. At the time of its kinetic collision with Dimorphos, the DART spacecraft weighed about 580 kilogrammes (1280 pounds). Dimorphos mass has not been directly measured, but it is estimated to be about 5 billion kgs using assumptions about the asteroid's density and size. Furthermore, Dimorphos' orbit was altered by DART's kinetic impact in order to move it a little bit closer to Didymos. Should an asteroid ever be discovered, the DART mission serves as a demonstration of the ability to respond to a potential threat from an asteroid impact. DART is a spacecraft created to crash into an asteroid as a technological experiment. The asteroid DART's target DOES NOT pose a risk to Earth. Should an Earth-threatening asteroid be found in the future, this asteroid system is the ideal testing ground to determine whether purposefully crashing a spacecraft into an asteroid is an effective way to change its course. Only about 42% of those asteroids have been discovered as of March 2023, despite the fact that no known asteroid larger than 140 metres in size has a significant chance of hitting Earth for the next 100 years.

1) PROGRAMING ALGORITHM

Break the program into small time steps

1. Find the position vector r (position vector between Didymos and Dimorphos).
2. Calculate the gravitational force F_G using r.

$$\mathbf{F_G = G * \frac{(m1 * m2)}{r^2}}$$

3. Calculate/Update momentum of both bodies using F_G.

$$\mathbf{p_{final} = p_{initial} + F_G \times dt}$$

4. Find the new position for both bodies and the spacecraft using the updated momentum.

$$\mathbf{x_{final} = x_{initial} + \left(\frac{\Delta p}{(m)}\right) \times \Delta t}$$

5. Apply the condition for change in momentum of the moon Dimorphos only after collision.

$$\mathbf{p_{final} = p_{initial} + (F_G - F_{Collision}) \times dt}$$

6. Update the time.
7. Repeat the steps from 1 to 5 for a particular time period.
8. Calculate the deflection angle using dot product.

$$\boldsymbol{\theta} = \mathbf{cos^{-1}\left(\frac{(A \cdot B)}{(|A|\ |B|)}\right)}$$

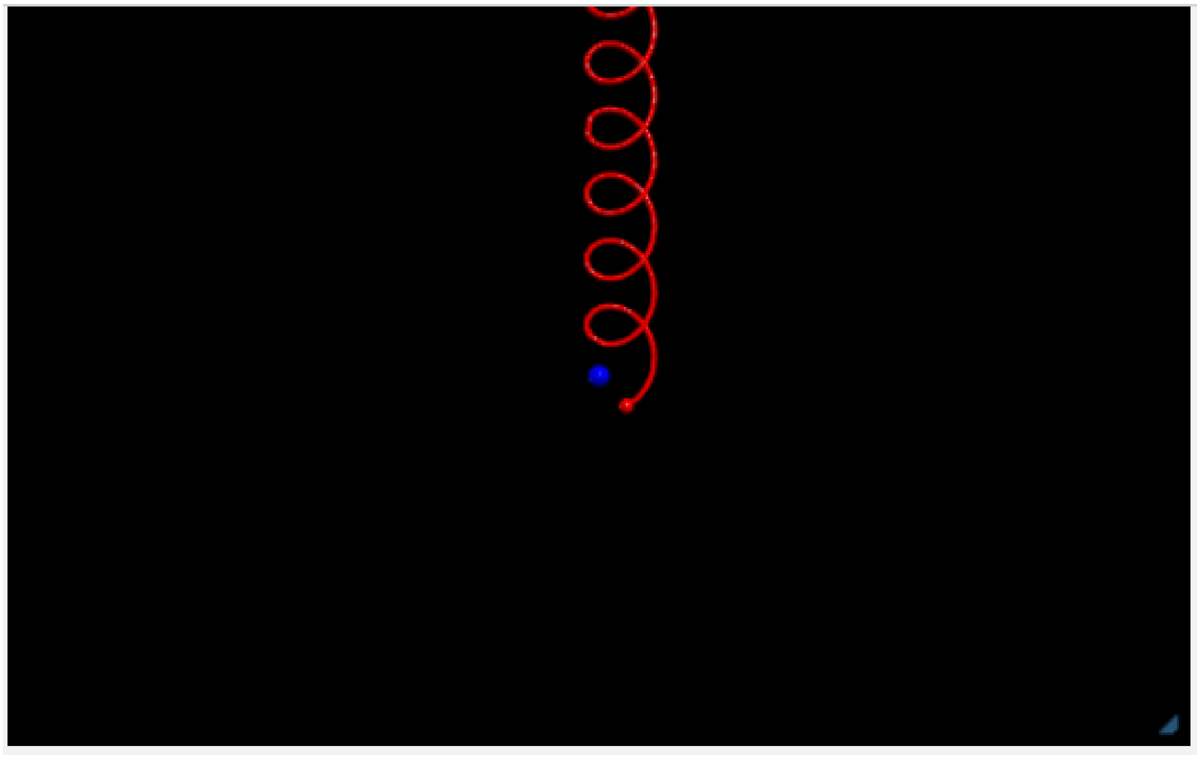
9. Calculate the initial and final orbital time period of Dimorphos using the given formula

$$\mathbf{T = 2 \times \boldsymbol{\pi} \times \sqrt{\frac{(r^3)}{(G \times (m_1 + m_2))}}}$$

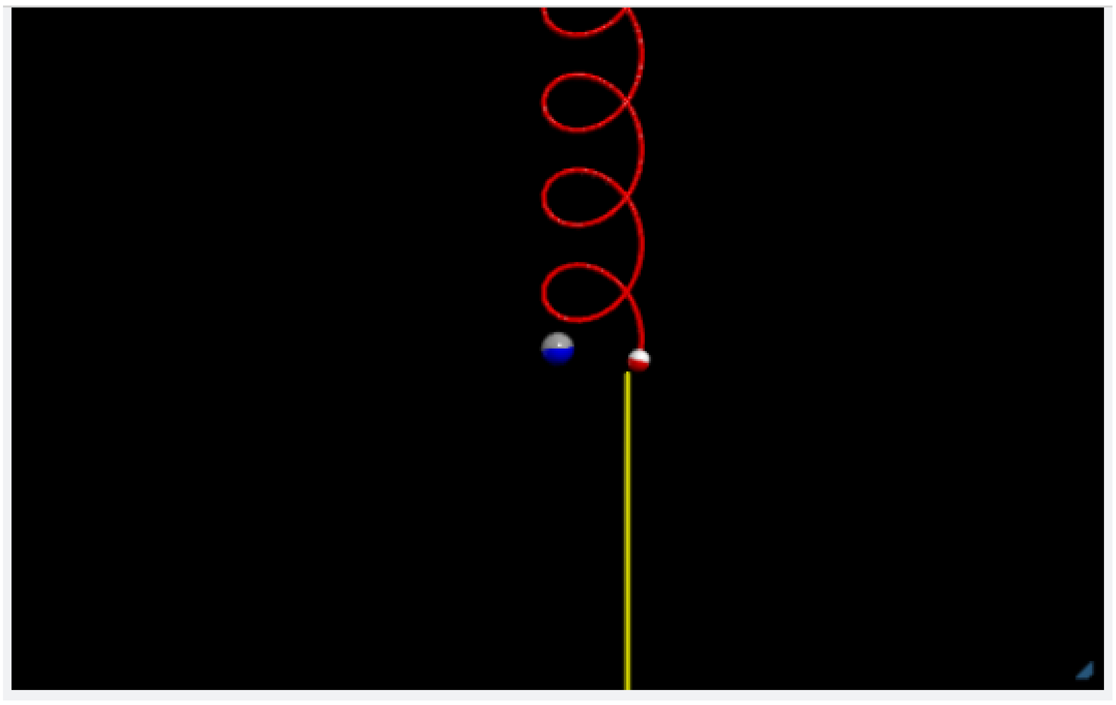
2) PYTHON SYNTAX:

```
from vpython import*
G=6.67e-11
ME=5.27e10
RE=780
MM=5e9
RM=160
REM=1.2e3
didymos = sphere(pos=vector(0,0,0),radius=0.5*RE, texture=textures.rough)
moon=sphere(pos=didymos.pos+REM*vector(1,0,0),radius=RM, make_trail=True)
didymos1 = sphere(pos=vector(0,0,0),radius=0.5*RE, color=color.blue)
moon1=sphere(pos=didymos.pos+REM*vector(1,0,0),radius=RM,color=color.red, make_trail=True)
v0 = sqrt(G*ME/REM)
v1 = sqrt(G*MM/REM)
moon.p = MM*vector(0,-v0,0)
didymos.p =ME*vector(0,-v1,0)
v2 = v0-0.99915e-1 #change in velocity after collision
l=0
moon1.p = MM*vector(0,-v0,0)
didymos1.p =ME*vector(0,-v1,0)
ro=RM/2
dart=sphere(pos=didymos.pos+vector(1000,-100*RE,0),radius=ro,color=color.yellow, make_trail=True)
dart.m =610
vo0 =0.0225
theta = -10*pi/180
dart.p = dart.m*vo0*vector(0,cos(theta),0)
t = 0
dt = 200
while t<4.0e6:
    rate(3000)
    scene.camera.follow(didymos)
        #path asteroid will follow without collision
    rem = moon1.pos- didymos1.pos
    Fem = -G*MM*ME*norm(rem)/mag(rem)**2
    moon1.p = moon1.p + Fem*dt
    didymos1.p = didymos1.p -Fem*dt
    moon1.pos = moon1.pos + moon1.p*dt/MM
    didymos1.pos = didymos1.pos + didymos1.p*dt/ME
    r = dart.pos - moon.pos
    #path the asteroid will follow after collision
if mag(r)>(RM+ro):
    rem = moon.pos- didymos.pos
    Fem = -G*MM*ME*norm(rem)/mag(rem)**2
    moon.p = moon.p + Fem*dt
    didymos.p = didymos.p -Fem*dt
    moon.pos = moon.pos + moon.p*dt/MM
    didymos.pos = didymos.pos + didymos.p*dt/ME
    dart.pos = dart.pos + dart.p*dt/dart.m
    t=t+dt
if mag(r)<=(RM+ro):
    if (l==0):
        moon.p = MM*vector(0,-v2,0)
        l=1
    dart.visible=False
    rem = moon.pos-didymos.pos
    Fem = -G*MM*ME*norm(rem)/mag(rem)**2
      #update momentum
    moon.p=moon.p+(Fem)*dt
    didymos.p =didymos.p-(Fem)*dt
      #update position
    moon.pos=moon.pos+moon.p*dt/MM
    didymos.pos = didymos.pos + didymos.p*dt/ME
    t=t+dt
    vGQ0=vector(0,-v1,0)
    dp=didymos.p/ME
    theta=acos(dot(vGQ0,dp)/(mag(vGQ0)*mag(dp)))
    print("deflection angle = ",theta*180/pi," deg")
    Ti=2*3.14*sqrt((1200**3)/(G*ME))
    Ti=(Ti/60)
    print("Before collision Time period = ",Ti,"MINS")
    Tf=2*3.14*sqrt((mag(rem)**3)/(G*ME))
    Tf=(Tf/60)
    Tf1=Ti-Tf
    print("After collision Time period = ",Tf,"MINS")
    print("DIFFERENCE = ",Tf1,"MINS")
```

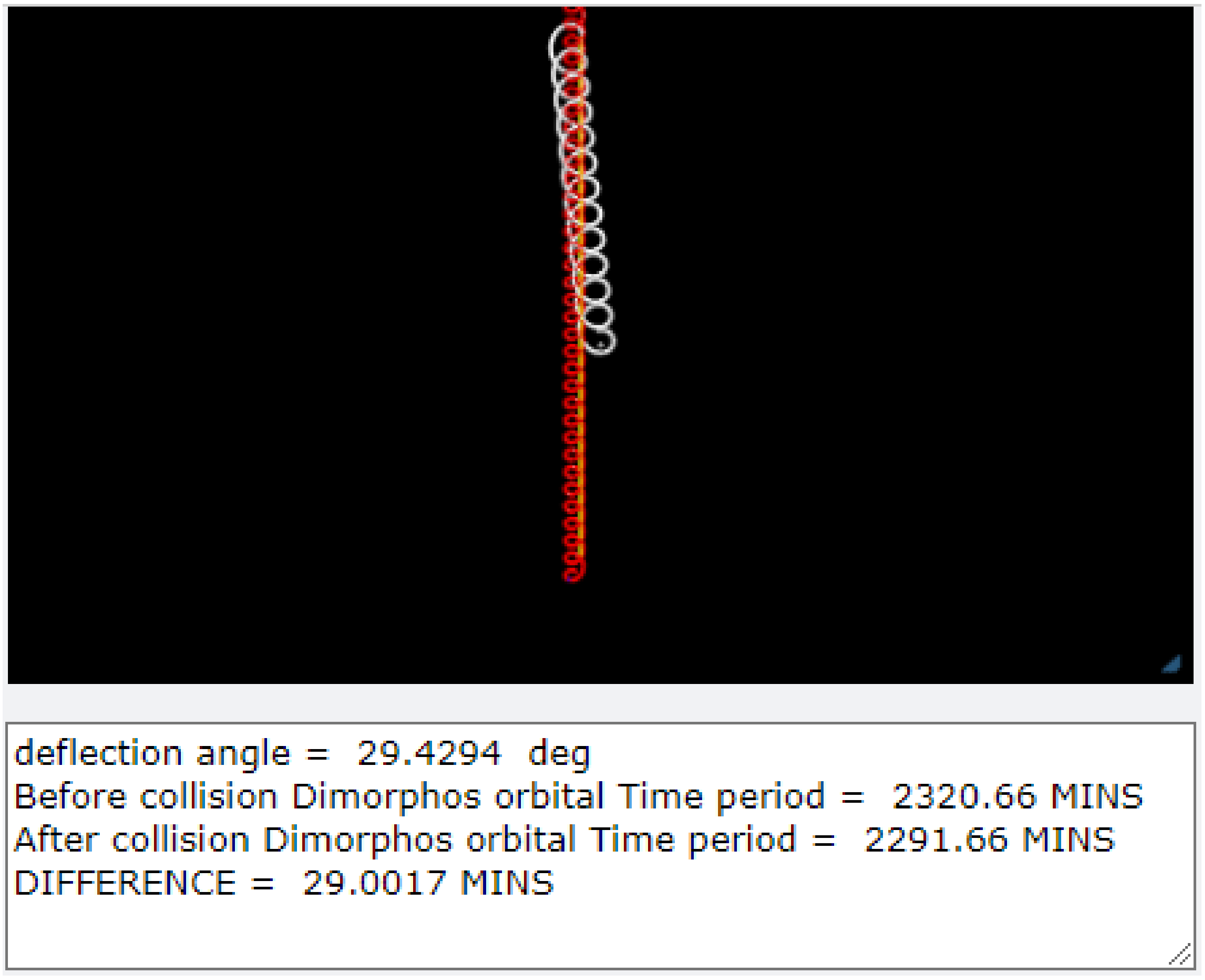
5. OUTPUT



Straight path of Didymos(Blue ball) and Dimorphos(Red ball) system in space. Red spiral denotes the trail/path of Dimorphos around Didymos as the system is moving forward.



Time of collision of spacecraft with Dimorphos, yellow line indicates the trail/Path of spacecraft.



```
deflection angle =  29.4294  deg
Before collision Dimorphos orbital Time period =  2320.66 MINS
After collision Dimorphos orbital Time period =  2291.66 MINS
DIFFERENCE =  29.0017 MINS
```

Change in path of the system after the collision with the spacecraft is denoted with the white spiral, while the red spiral denotes the path which the asteroid system should have followed if there was no collision.

6. CODE EXPLAINED WITH SYNTAX

1. Physical Constants used in the program

a) **Gravitational constant:** **Program statement 2**

Real value of G=$6.67430\times10^{-11}m^3/kg$

Python syntax: G=6.67e-11

G is the variable assigned to represent Gravitational constant

b) **Mass of Didymos:**The mass of the Didymos system is estimated at 528 billion kg

Assigning value of Mass of Didymos: Program statement 3

Real value: Mass of Didymos=5.27×10^{10} **kg**

Python syntax: ME=5.27e10

ME is a variable that represents mass of Didymos

c) **Radius of Didymos**

Didymos, the larger asteroid of the binary pair (also called Didymos A) is about a half mile (780 meters) in diameter

Assigning value of Radius of Didymos: Program statement 4

Real value: Radius of Didymos=780 m

Python syntax: RE=780

RE is a variable that represents Radius of Didymos

d) **Mass of Dimorphos:** The mass of Dimorphos is estimated as roughly 5 billion kilograms

Assigning value of Mass of Dimorphos: Program statement 5

Real value: Mass of Didymos=5×10^{9} **kg**

Python syntax: MM=5e9

MM is a variable that represents mass of Diphormos

e) **Radius of Dimorphos: Program statement 6**

The moonlet, Dimorphos (Didymos B), is about 525 feet (160 meters) in diameter.

Assigning value of Radius of Dimorphos:

Real value: Radius of Dimorphos =160 m

Python syntax: RE=780

RM is a variable that represents Radius of Diphormos

f) **Distance between Didymos and Diphormos: Program statement 7**

The separation between the centres of the two asteroids is 1.2 kilometres (0.74 miles)

Real value: Distance between two asteroids =1200 m

REM is a variable that represents distance between the two asteroids

Python syntax: REM=1.2e3

g) **Mass of space craft: Program statement 24**

Real value: Mass of Spacecraft =610 kg

dart.m is a variable that represents Mass of Spacecraft

Python syntax: dart.m =610

2. Sphere function: Program statements 8,9,10,11,21

Making 3d model of earth and asteroid

Python syntax: didymos = sphere(pos=vector(0,0,0),radius=0.3*RE, texture=textures.rough)

#didymos model

Making a sphere at origin with 0.3 times radius of didymos with a rough.

Pythonsyntax:moon=sphere(pos=didymos.pos+REM*vector(1,0,0),radius=RM,make_trail=True)

#diphormos model

Making a sphere at (REM,0,0) in 3d space with radius of Diphormos and switching on trail property

Python syntax: didymos1 = sphere(pos=vector(0,0,0),radius=0.3*RE, color=color.blue))

#didymos model that will show the path of asteroid without collision

Making a sphere at origin with 0.3 times radius of didymos with blue colour

Python syntax:
moon1=sphere(pos=didymos.pos+REM*vector(1,0,0),radius=RM,color=color.red,make_trail=True)

#diphormos model that will show the path of asteroid without collision

Making a sphere at (REM,0,0) in 3d space with radius of Diphormos with red colour and switching on trail property

Python syntax:
dart=sphere(pos=didymos.pos+vector(1000,100*RE,0),radius=ro,color=color.yellow, make_trail=True)

#Dart spacecraft model

Making a sphere at (1000,-100*RE,0) in 3d space with radius of 0.5 times Diphormos with yellow colour and switching on trail property

3. **Velocity: Program statements 12,13,23,16**

The velocity of an asteroid can vary depending on its orbit, distance from the Sun, and other factors. We are here assigning a using the formula initial value to the velocity of asteroids using **equation 23** and **equation 24**.

Assigning initial velocity of asteroids:

Python syntax: v0 = sqrt(G*ME/REM)

v0 is a variable that represents the initial velocity of Diphormos

Python syntax: v1 = sqrt(G*MM/REM)

v1 is a variable that represents the initial velocity of Didymos

Initial velocity of the spacecraft was assigned a random value

Python syntax: vo0 =0.0225

vo0 is a variable that represents the initial velocity of Spacecraft

Python syntax: v2 = v0-0.99915e-1 #change in velocity after collision

V2 is a variable that represents the change in velocity of dimorphos after collision

4. **Calculating initial momentum: Program statements 14,15,18,19,25,24**

Vector function in glowscript

In GlowScript, the `vector()` function is a built-in function that creates and returns a vector object. Vectors are fundamental data types in GlowScript used to represent quantities with both magnitude and direction, such as positions, velocities, forces, and more. Here we will be calculating momentum using equation 2.

Assigning initial momentum:

($P_{ASTEROID} = M_{ASTEROID} \times V_{ASTEROID}$)

Python syntax :
```
moon.p = MM*vector(0,-v0,0)
didymos.p =ME*vector(0,-v1,0)
moon1.p = MM*vector(0,-v0,0)
didymos1.p =ME*vector(0,-v1,0)
```

Initial momentum to the spacecraft was assigned by

$P_{SPACECRAFT} = M_{SPACECRAFT} \times V_{SPACECRAFT}$

Python syntax :
```
dart.p = dart.m*vo0*vector(0,cos(theta),0)
theta = -10*pi/180
```
(Angle)

moon.p , didymos.p , didymos1.p , mon1.p , dart.p are the variablea that represents the momentum of the corresponding models.

5. **Find the position vector r (position vector between asteroid and earth): Program statements 31,39,52**

Finding the position vector r (position vector between Didymos and Dimorphos).

Position vector between didymo and diphormos = Position vector of didymos – Position vector of diphormos

Python syntax : rem = moon1.pos- didymos1.pos

rem = moon.pos- didymos.pos

rem is a variable that represents the Position vector between asteroid and earth

1. **Calculate the gravitational force F_G using r using equation 1): Program statements 32,40,53**

   Universal law of Gravitation

   Python syntax : Fem = -G*MM*ME*norm(rem)/mag(rem)**2

   Fem is the variable that represents the gravitational force between didymos and diphormos

   norm(rem) represents the unit vector of rem and mag(rem) represents the magnitude of rem

2. **Calculate/Update using Momentum: Program statements 33,34,41,54,42,55**

   $$\mathbf{p}_{\text{new}} = \mathbf{p}_{\text{initial}} + \mathbf{F} \times \mathbf{dt}$$

   Python syntax : moon1.p = moon1.p + Fem*dt

   didymos1.p = didymos1.p -Fem*dt

   moon.p = moon.p + Fem*dt

   didymos.p = didymos.p -Fem*dt

   Fem and dt are the variable that represents the gravitational force and small change in time

3. **Finding the new position using the updated momentum: Program statements 35,36,43,56,44,57,45**

   $$\mathbf{x}_{\text{final}} = \mathbf{x}_{\text{initial}} + \left(\frac{\Delta \mathbf{p}}{(\mathbf{m})}\right) \times \Delta\mathbf{t}$$

   Python syntax : moon1.pos = moon1.pos + moon1.p*dt/MM

   didymos1.pos = didymos1.pos + didymos1.p*dt/ME

   moon.pos = moon.pos + moon.p*dt/MM

   didymos.pos = didymos.pos + didymos.p*dt/ME

   dart.pos = dart.pos + dart.p*dt/dart.m

   moon.pos , didymos.pos , didymos1.pos , mon1.pos , dart.pos are the variables that represents the position of the corresponding models.

   dt is the variable that represents the small change in time

4. **Updating the time: Program statement 68**

   $$\mathbf{t}_{\text{new}} = \mathbf{t}_{\text{initial}} + \mathbf{dt}$$

   Python syntax : t = t + dt

   t is the variable that represents time

   dt is the variable that represents the small change in time

5. **Repeat the statement from 32 to 60 for a particular time period: Program statement 28**

The loop will be executed until the value of t is less than 4×10^6

Python syntax : while t<4.0e6

6. **Calculating the deflection angle using dot product: Program statement 59,60,61**

   Mathematically, calculation of angle using dot product is given by the **equation 21**

   Python syntax: vGQ0=vector(0,-v1,0) #direction of velocity of Didymos asteroid without collision

   dp=didymos.p/ME #direction of velocity of Didymos asteroid with collision

   theta=acos(dot(vGQ0,dp)/(mag(vGQ0)*mag(dp)))#calculationofdeflection angle

7. **Calculating the orbital time period of Diphormos before and after collision: Program statement 63,64,66,67,68**

   Here we will be using the **equation 29** to calculate the orbital time period

   Python syntax: Ti=2*3.14*sqrt((1200**3)/(G*ME))

   #orbital time period of Diphormos before collision

   Ti=(Ti/60)

   #converting seconds to minutes

   Tf=2*3.14*sqrt((mag(rem)**3)/(G*ME)))

   #orbital time period of Diphormos after collision

   Tf=(Tf/60)

   #converting seconds to minutes

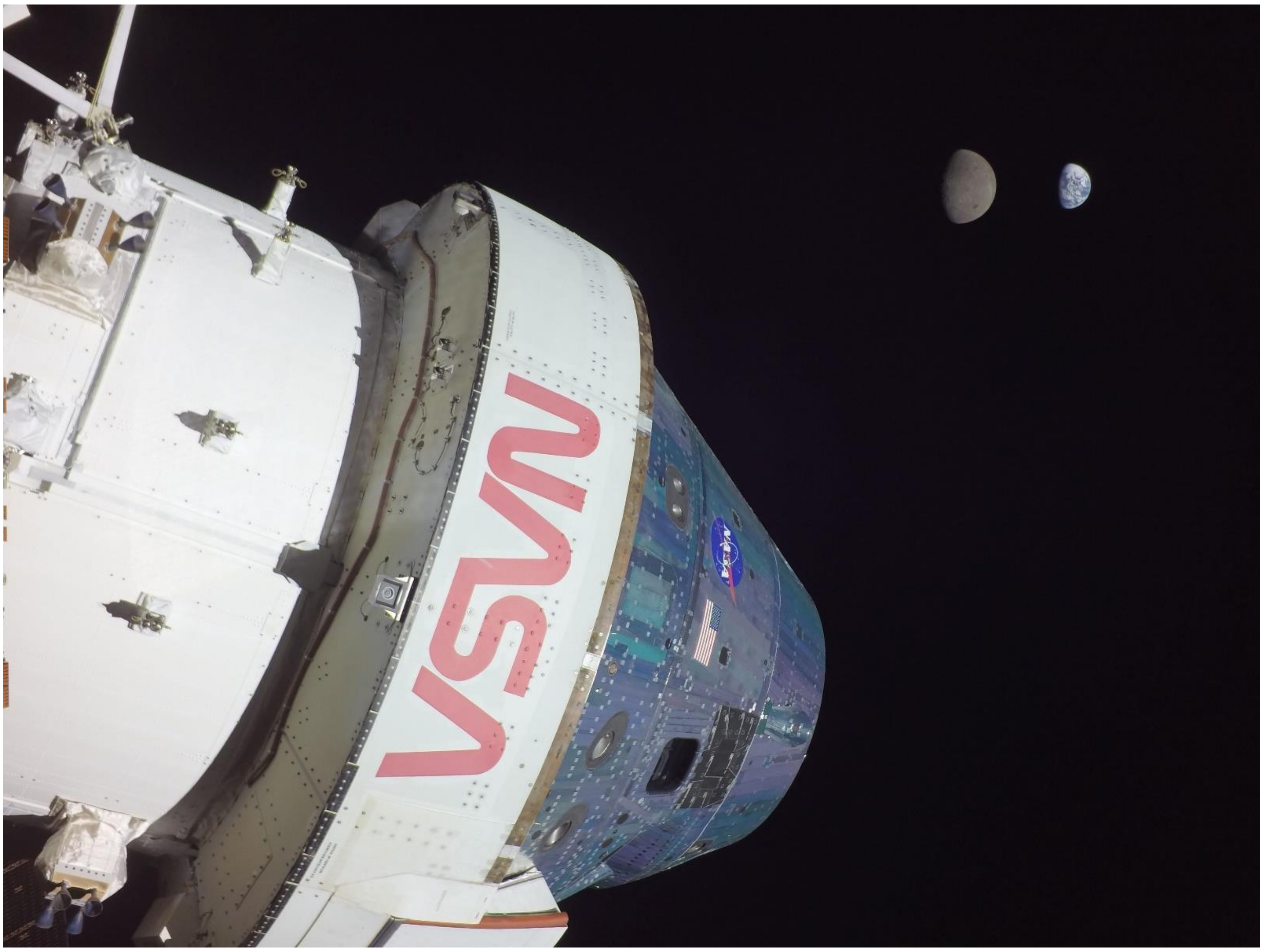
   Tf1=Ti-Tf

   #difference between initial and final time period

• REPLICATING ARTEMIS 1 (EXPLORATION MISSION-1 (EM-1))

Introduction:

Artemis 1 represents a monumental leap towards returning humans to the Moon and beyond. Through its pioneering technology, scientific objectives, and global collaboration, this program sets the stage for a new era of human presence in space. As Artemis 1 ushers in a sustainable lunar infrastructure, it also propels us closer to unlocking the mysteries of our universe and shaping the destiny of humanity as a multiplanetary species. With every launch, every scientific discovery, and every step taken on alien soil, Artemis 1 embodies the spirit of exploration, reminding us that the stars are not out of reach, but rather a beckoning call to push the boundaries of what we thought was possible. Artemis I is the first integrated test of NASA’s deep space exploration systems: the Orion spacecraft, Space Launch System (SLS) rocket and the ground systems at the agency’s Kennedy Space Centre in Florida. The first in a series of increasingly complex missions, Artemis I is an uncrewed flight test that will provide a foundation for human deep space exploration and demonstrate our commitment and capability to return humans to the Moon and extend beyond.



Artemis 1 looking back at the Moon and Earth at its maximum distance from Earth on November 28, 2022

During this flight, Orion will launch atop the most powerful rocket in the world and fly farther than any spacecraft built for humans has ever flown. Over the course of the mission, it will travel 280,000 miles (450,000 kilometers) from Earth and 40,000 miles (64,000 kilometers) beyond the far side of the Moon.

Orion will stay in space longer than any human spacecraft has without docking to a space station and return home faster and hotter than ever before.

This first Artemis mission will demonstrate the performance of both Orion and the SLS rocket and test our capabilities to orbit the Moon and return to Earth. The flight will pave the way for future missions to the lunar vicinity, including landing the first woman and first person of color on the surface of the Moon.

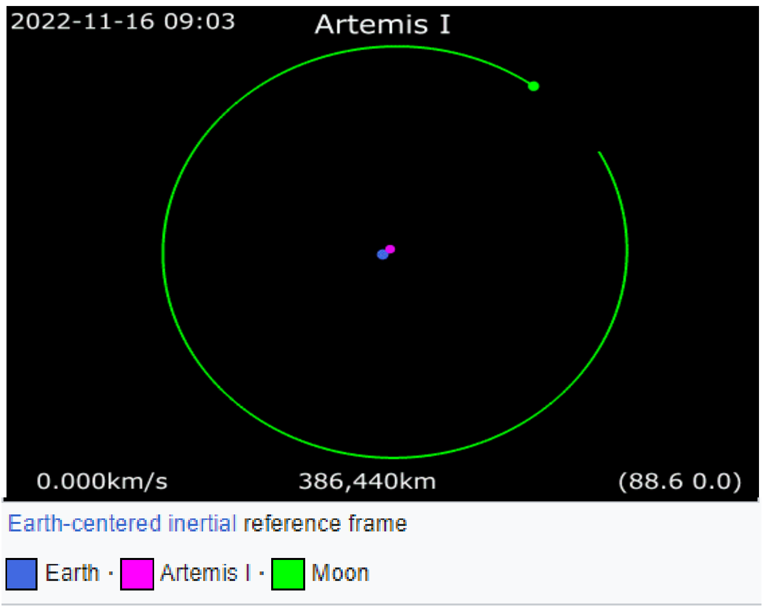
With Artemis I, NASA sets the stage for human exploration into deep space, where astronauts will build and begin testing the systems near the Moon needed for lunar surface missions and exploration to other destinations farther from Earth, including Mars. With Artemis, NASA will collaborate with industry and international partners to establish long-term exploration for the first time.

Mission Profile:

Artemis 1 embarks on an extraordinary journey, pushing the boundaries of human exploration and laying the foundation for future lunar missions. Let's explore the stages of this remarkable odyssey:

Overview: [8] [9]

- Launch site: Launch Pad 39B at NASA's Kennedy Space Center in Florida
- Launch date: Nov. 16, 2022
- Launch time: 1:47 a.m. EST
- Mission Duration: 25 days, 10 hours, 53 minutes
- Destination: distant retrograde orbit around the Moon
- Total mission miles: approximately 1.4 million miles (2.3 million kilometers)
- Targeted splashdown site: Pacific Ocean, off the coast of San Diego
- Return speed: Up to 25,000 mph (40,000 kph)
- Splashdown: Dec. 11, 2022

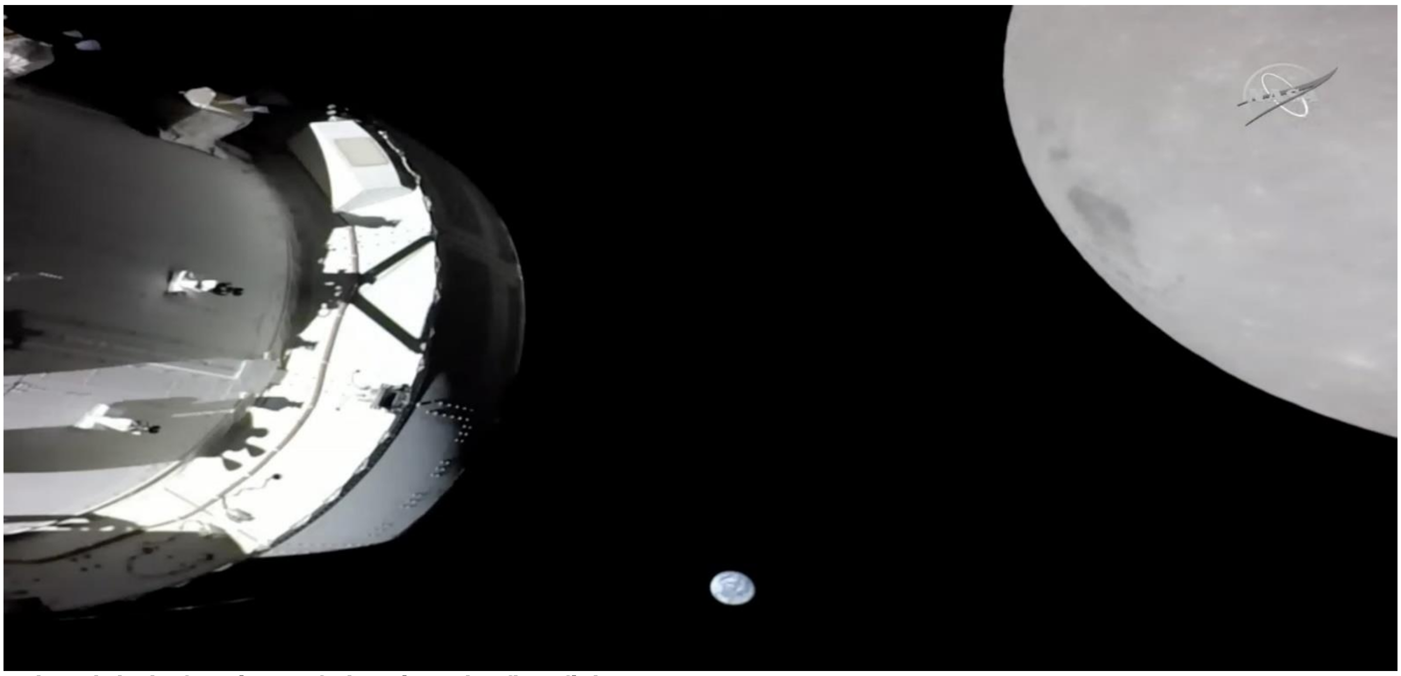

2022-11-16 09:03
Artemis I
0.000km/s
386,440km
(88.6 0.0)
Earth-centered inertial reference frame
🟦 Earth · 🟪 Artemis I · 🟩 Moon

Mission profile animation

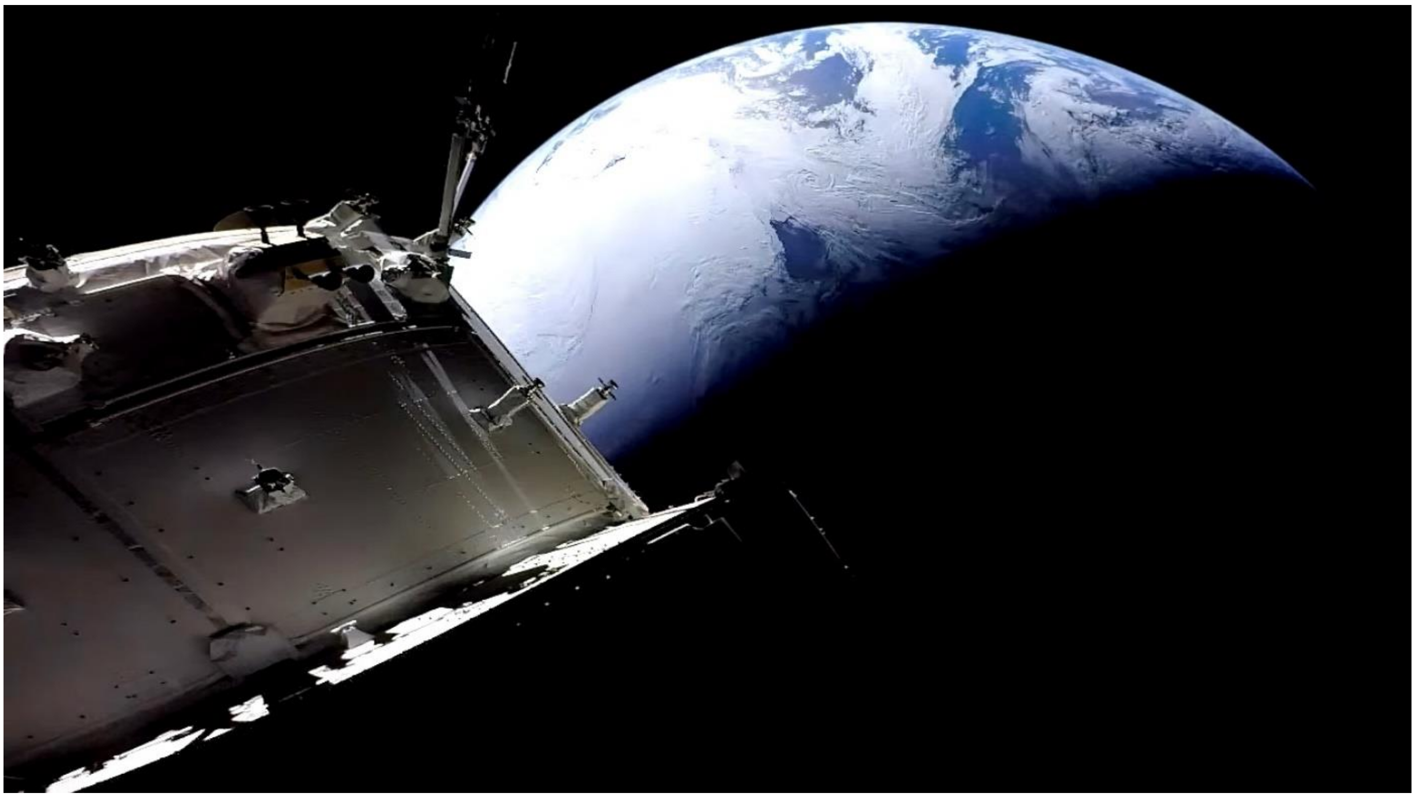
1. **Launch and Ascent:** The journey of Artemis 1 begins with the awe-inspiring launch of the Space Launch System (SLS). The SLS, equipped with its powerful engines and solid rocket boosters, propels the Orion spacecraft into space with an incredible display of raw power. The launch marks the culmination of years of planning, engineering, and anticipation, as the spacecraft soars beyond the confines of Earth's atmosphere, carrying the hopes and dreams of a new era of space exploration.
2. **Earth Departure:** After reaching space, the Orion spacecraft separates from the spent rocket stages and sets its sights on the Moon. The spacecraft fires its engines to initiate the Earth departure maneuver, propelling it on a trajectory away from our planet and towards the lunar realm. This crucial maneuver sets the stage for the awe-inspiring journey ahead.
3. **Translunar Injection**: Once on its way, the Orion spacecraft undergoes a critical maneuver known as translunar injection. This maneuver involves firing the spacecraft's engines to increase its velocity and adjust its trajectory, ensuring a precise course towards the Moon. The propulsion systems propel the

spacecraft onward, as humanity takes another significant step towards returning to our celestial neighbour.

4. **Lunar Flyby:** As Artemis 1 ventures deeper into space, it approaches the Moon for a mesmerizing lunar flyby. While not intending to land on the lunar surface, this stage of the mission provides an unparalleled opportunity to observe and study our nearest celestial companion up close. The spacecraft navigates precise trajectory that takes it in close proximity to the Moon, allowing astronauts aboard the Orion spacecraft to witness the awe-inspiring lunar landscape firsthand.


Orion right before loss of signal on the first flyby

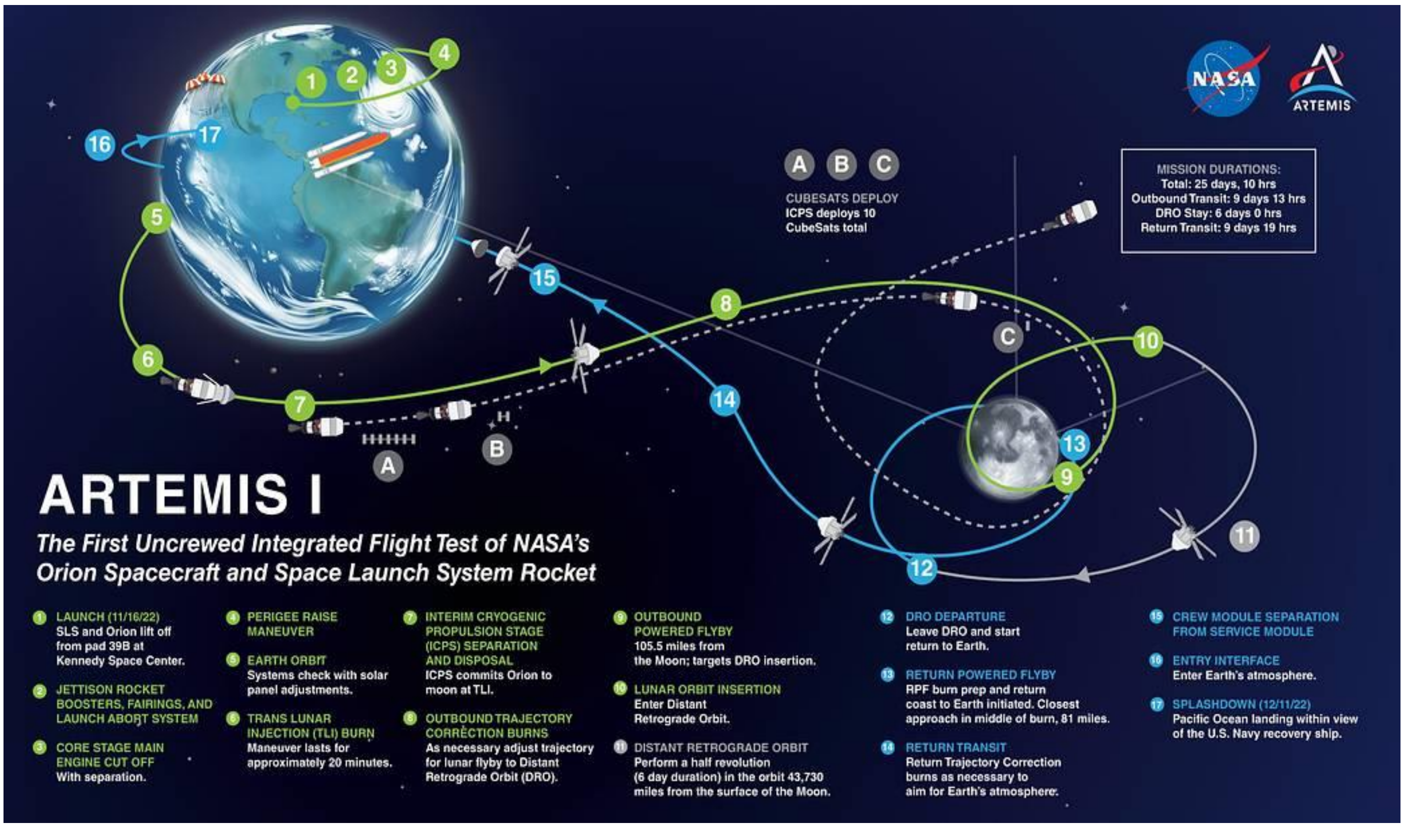
Lunar Return and Re-entry: Having completed its lunar flyby, Artemis 1 initiates the return journey to Earth. The propulsion systems of the Orion spacecraft are engaged once again, this time to gradually


Artemis 1 images the Earth before re-entry

slow down the vehicle's speed, allowing it to re-enter Earth's atmosphere safely. The spacecraft faces the intense heat of re-entry, protected by its advanced heat shield, which withstands the extreme temperatures generated during atmospheric re-entry.

6. **Splashdown and Recovery:** After enduring the rigors of re-entry, the Orion spacecraft gracefully descends through the Earth's atmosphere, aided by parachutes that slow its descent. The spacecraft ultimately makes a controlled splashdown in the designated recovery zone, typically in the ocean. Recovery teams swiftly move in to retrieve the crew module and safely bring the astronauts back to solid ground, marking the triumphant conclusion of the Artemis 1 journey.

Orion descending down to the Pacific Ocean



ARTEMIS I

The First Uncrewed Integrated Flight Test of NASA's Orion Spacecraft and Space Launch System Rocket

A B C CUBESATS DEPLOY
ICPS deploys 10 CubeSats total

MISSION DURATIONS:
Total: 25 days, 10 hrs
Outbound Transit: 9 days 13 hrs
DRO Stay: 6 days 0 hrs
Return Transit: 9 days 19 hrs

1. LAUNCH (11/16/22)
SLS and Orion lift off from pad 39B at Kennedy Space Center.
2. JETTISON ROCKET BOOSTERS, FAIRINGS, AND LAUNCH ABORT SYSTEM
3. CORE STAGE MAIN ENGINE CUT OFF
With separation.
4. PERIGEE RAISE MANEUVER
5. EARTH ORBIT
Systems check with solar panel adjustments.
6. TRANS LUNAR INJECTION (TLI) BURN
Maneuver lasts for approximately 20 minutes.
7. INTERIM CRYOGENIC PROPULSION STAGE (ICPS) SEPARATION AND DISPOSAL
ICPS commits Orion to moon at TLI.
8. OUTBOUND TRAJECTORY CORRECTION BURNS
As necessary adjust trajectory for lunar flyby to Distant Retrograde Orbit (DRO).
9. OUTBOUND POWERED FLYBY
105.5 miles from the Moon; targets DRO insertion.
10. LUNAR ORBIT INSERTION
Enter Distant Retrograde Orbit.
11. DISTANT RETROGRADE ORBIT
Perform a half revolution (6 day duration) in the orbit 43,730 miles from the surface of the Moon.
12. DRO DEPARTURE
Leave DRO and start return to Earth.
13. RETURN POWERED FLYBY
RPF burn prep and return coast to Earth initiated. Closest approach in middle of burn, 81 miles.
14. RETURN TRANSIT
Return Trajectory Correction burns as necessary to aim for Earth's atmosphere.
15. CREW MODULE SEPARATION FROM SERVICE MODULE
16. ENTRY INTERFACE
Enter Earth's atmosphere.
17. SPLASHDOWN (12/11/22)
Pacific Ocean landing within view of the U.S. Navy recovery ship.

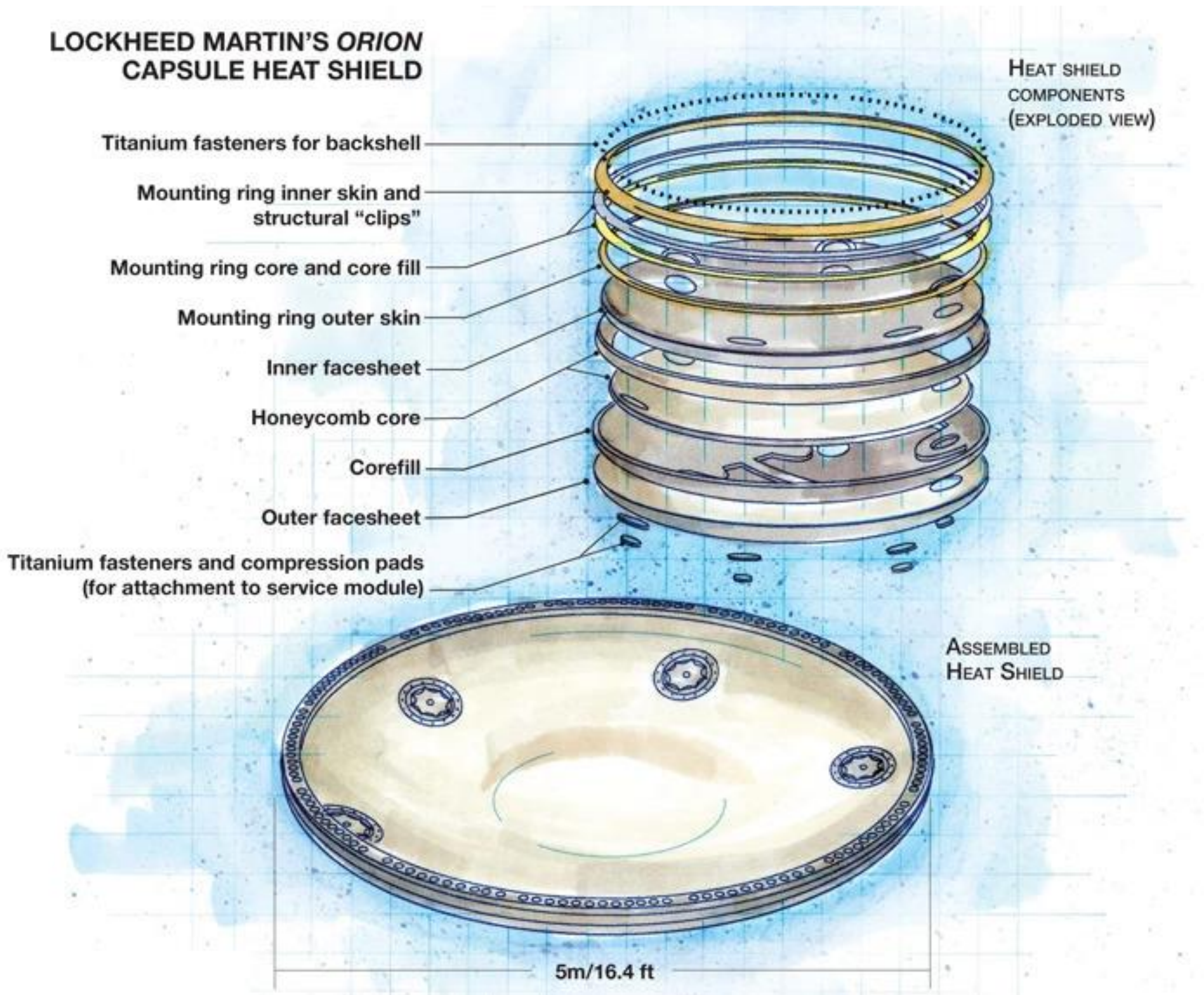
Summary of the Artemis 1 mission [10]

Throughout its incredible journey, Artemis 1 will make the way for future lunar exploration missions. It serves as a critical testbed for the Space Launch System, validating its performance and capabilities, while also proving the readiness of the Orion spacecraft for crewed missions. Artemis 1's achievements bring humanity one step closer to establishing a sustainable presence on the Moon, unlocking the potential for scientific discovery, resource utilization, and the eventual expansion of human presence further into our solar system.

Objectives:

Artemis 1 has several primary objectives that serve as building blocks for future lunar missions:

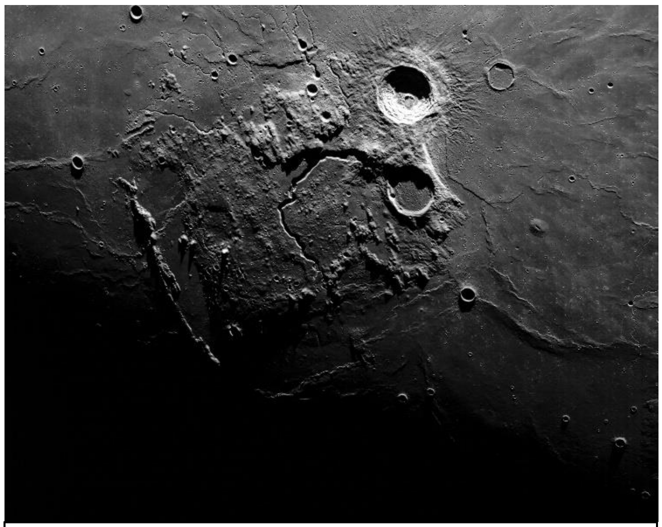
Demonstrate Orion's heat shield can withstand the high speed and high heat conditions when returning through Earth's atmosphere from lunar velocities


LOCKHEED MARTIN'S *ORION* CAPSULE HEAT SHIELD
HEAT SHIELD COMPONENTS (EXPLODED VIEW)
Titanium fasteners for backshell
Mounting ring inner skin and structural "clips"
Mounting ring core and core fill
Mounting ring outer skin
Inner facesheet
Honeycomb core
Corefill
Outer facesheet
Titanium fasteners and compression pads (for attachment to service module)
ASSEMBLED HEAT SHIELD
5m/16.4 ft

When Orion returns from the Moon, it will be traveling nearly 25,000 mph (40,000 kph) and experience temperatures up to 5,000 degrees Fahrenheit (2,800 degrees Celsius) as it enters Earth's atmosphere, much faster and hotter than a return from low-Earth orbit. While the heat shield has undergone extensive testing on Earth and was demonstrated on Exploration Flight Test-1 in 2014, no aerodynamic or aerothermal test facility can recreate the conditions the heat shield will experience returning at lunar return speeds. Validating heat shield performance is required before crews fly in Orion.

Demonstrate operations and facilities during all mission phases

From launch countdown through recovery of Orion from the Pacific Ocean at the end of its mission, Artemis I provide an opportunity to test many aspects of NASA's launch facilities and ground-based infrastructure, SLS operations, including separation events during ascent, Orion operations in space, and recovery procedures. During the flight, engineers will verify systems such as the spacecraft's communications, propulsion, and navigation systems. Operating Orion in space will give engineers further confidence the spacecraft can tolerate the extreme thermal environment of deep space and successfully pass through the Van Allen Radiation Belt, that Orion's main engine and solar array wings work as designed, and the flight operations teams can successfully manage and execute the mission, as well as demonstrate the performance of support systems for NASA facilities needed during the flight.


Rilles and craters near lunar terminator captured by orion

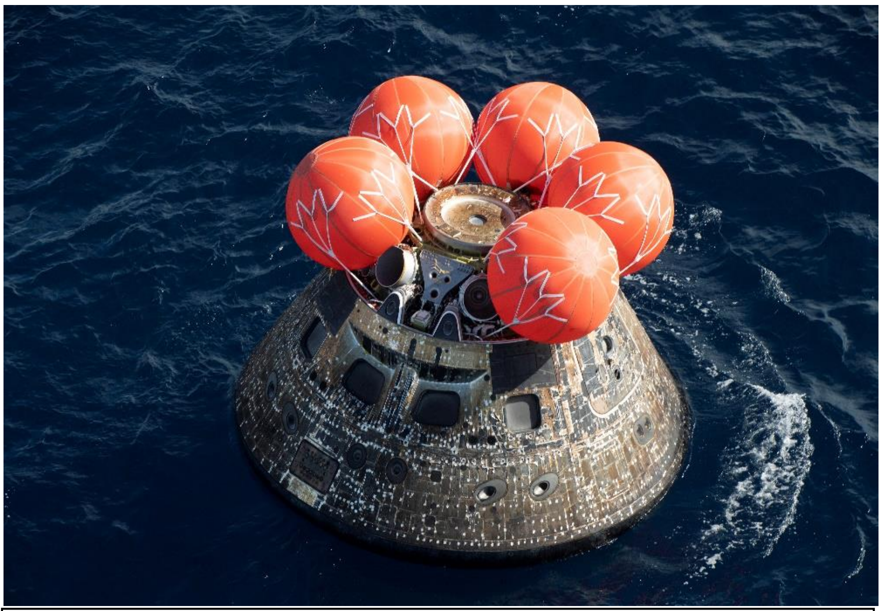
Close up picture from Orion near closest approach on the first flyby

Environmental Data Collection:

Orion is equipped with a range of instruments and sensors to collect data about the space environment it encounters during the mission. This includes measurements of radiation levels, magnetic fields, micrometeoroid impacts, and other environmental factors. By gathering this data, scientists can better understand the challenges and risks associated with long-duration space travel, enabling the development of improved shielding and protective measures for future astronauts.

Retrieve Orion after splashdown

While engineers will receive data throughout the flight, retrieving the crew module after splashdown will provide information to engineers to inform future missions. Once returned to Kennedy after the mission, technicians will conduct detailed inspections of Orion, retrieve data recorded on board during the flight, reuse components such as avionics systems, and retrieve information from payloads. It will also allow NASA to demonstrate its recovery techniques and procedures, which are critical to the safe return of future crews.



Orion off the coast of Baja California shortly after splashdown

Life Support Systems:

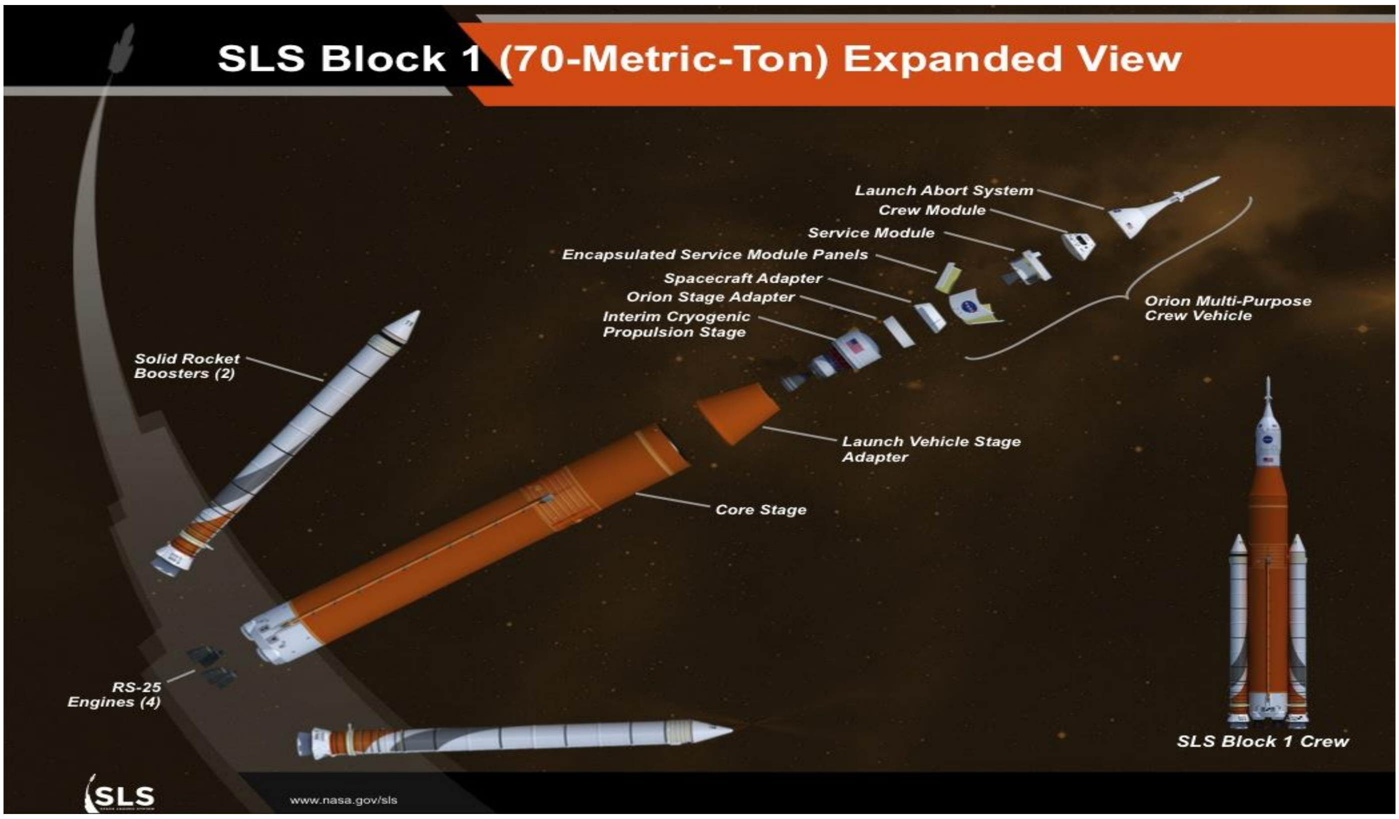
Orion's life support systems play a crucial role in sustaining the health and well-being of astronauts during deep space missions. Artemis 1 provides an opportunity to gather data on the performance and reliability of these systems, including air revitalization, temperature regulation, waste management, and water and food supplies. This data helps refine and improve life support technologies for longer-duration missions, laying the groundwork for sustainable lunar habitation and future exploration of other destinations.

Lunar flyby illustration

Accomplish additional flight test objectives

A number of additional objectives will demonstrate other capabilities and aspects of the rocket, spacecraft, integrated systems, and recovery plans. Some of these flight test objectives include certifying Orion’s optical navigation system, deploying the 10 CubeSats riding inside the Orion stage adapter, operating the technology and biology payloads onboard Orion, and collecting imagery throughout the mission.

Components: The Artemis 1 program consists of several key components that work in harmony to accomplish its objectives. [11]


SLS Block 1 (70-Metric-Ton) Expanded View
Launch Abort System
Crew Module
Service Module
Encapsulated Service Module Panels
Spacecraft Adapter
Orion Stage Adapter
Interim Cryogenic Propulsion Stage
Orion Multi-Purpose Crew Vehicle
Solid Rocket Boosters (2)
Launch Vehicle Stage Adapter
Core Stage
RS-25 Engines (4)
SLS Block 1 Crew
SLS
www.nasa.gov/sls

1. **Space Launch System (SLS):** The SLS, with its powerful engines and solid rocket boosters, serves as the backbone of Artemis 1. It provides the necessary thrust to launch the Orion spacecraft and other payloads into deep space.
2. **Orion Spacecraft:** Designed to support crewed missions, Orion serves as the primary vehicle for human exploration beyond Earth. , Orion will weigh over 20 tonnes in total. It will transport astronauts safely to the Moon and back, providing life support, radiation shielding, and advanced systems for navigation and communication.
3. **Exploration Ground Systems (EGS):** EGS encompasses the infrastructure, facilities, and support systems needed for Artemis 1 and future missions. This includes the launch complex, vehicle assembly, and launch operations, ensuring the successful integration and launch of the SLS and Orion.

Broader Implications:

Artemis 1 holds significant implications for both scientific discovery and human exploration beyond Earth:

1. **Lunar Science:** The Artemis missions, including Artemis 1, will contribute to advancing our understanding of the Moon's geology, resources, and potential for sustained human presence. Data collected during this program will pave the way for future research, such as establishing a lunar outpost or utilizing lunar resources for further space exploration.
2. **International Collaboration:** Artemis 1 represents a collaborative effort between NASA and its international partners, including the European Space Agency (ESA), Canadian Space Agency (CSA), and others. This global cooperation fosters shared knowledge, resources, and expertise, bringing nations together to achieve common goals in space exploration.
3. **Inspiration and Education:** Just as the Apollo program inspired generations of scientists, engineers, and dreamers, Artemis 1 has the potential to ignite a new wave of enthusiasm for space exploration.

Through public engagement, educational initiatives, and increased access to space, Artemis inspires the next generation to push boundaries and pursue careers in science, technology, engineering, and mathematics (STEM).
The Artemis I mission holds immense significance for the future of space exploration. By successfully demonstrating the capabilities of the SLS and Orion, Artemis I will provide essential data and insights that will inform subsequent missions. It will validate the engineering designs, systems, and technologies necessary for sustainable lunar exploration, setting the stage for crewed missions to the Moon in the future.

Furthermore, Artemis I will pave the way for Artemis II, the first crewed mission of the Artemis program. The knowledge gained from the uncrewed test flight will enable NASA to refine its systems, procedures, and safety protocols for human spaceflight. Artemis I represents a crucial step toward establishing a sustainable human presence on the Moon, fostering scientific discoveries, and enabling future exploration of Mars and beyond.

3) PROGRAMING ALGORITHM

1. Give Orion's momentum direction towards moon.

Break the program into small time steps

2. Calculate the force values for a particular distance and update the positions of moon, earth and Orion.

3. Deflect Orion after a particular distance to get into moon's orbit.

4. Continue calculating the force values for a particular distance while Orion is in the moon's orbit and update the positions of moon, earth and Orion.

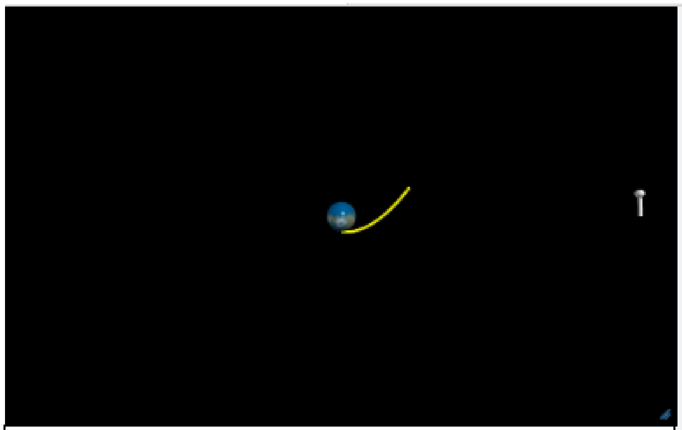
5. Deflect the Orion after a particular time period to get out of moon's orbit and direct it towards earth.

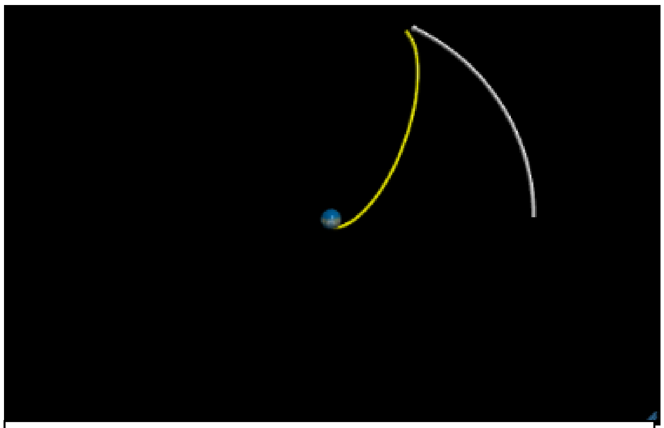
6. Continue calculating the force values while Orion is re-entering earth and update the positions of moon, earth and Orion.

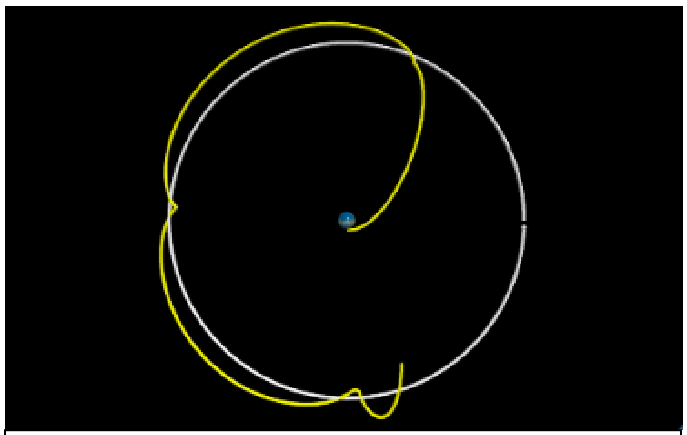
Python Syntax:

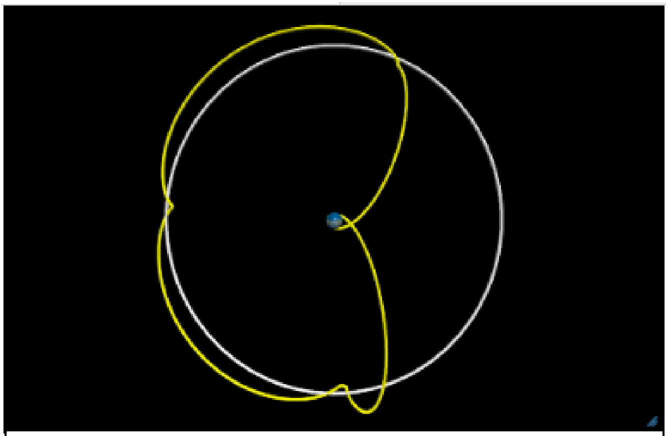
```
from vpython import *
G=6.67e-11
ME=5.972e24
RE=6.371e6
MM=7.348e22
RM=1.7371e6
REM=384400e3
earth=sphere(pos=vector(0,0,0),radius=3*RE,texture=textures.earth)
moon = sphere(pos=earth.pos+REM*vector(1,0,0),radius=3*RM, make_trail=True)
v0 = sqrt(G*ME/REM)
moon.p = MM*vector(0,v0,0)
earth.p = -moon.p
orion = sphere(pos=earth.pos+vector(0,-3.1*RE,0), radius=3*RM/2, color=color.yellow, make_trail=True)
orion.m = 1e4
vo0 =6.2e3
theta = -10*pi/180
orion.p = orion.m*vo0*vector(cos(theta),sin(theta),0)
t = 0
dt = 100
dl=0
sl=0
def force():
    rem = moon.pos- earth.pos
    reo = orion.pos - earth.pos
    rmo = orion.pos - moon.pos
    Fem = -G*MM*ME*norm(rem)/mag(rem)**2
    Feo = -G*ME*orion.m*norm(reo)/mag(reo)**2
    Fmo = -G*MM*orion.m*norm(rmo)/mag(rmo)**2
    moon.p = moon.p + Fem*dt
    earth.p = earth.p -Fem*dt
    orion.p = orion.p + (Feo + Fmo)*dt
    moon.pos = moon.pos + moon.p*dt/MM
    earth.pos = earth.pos + earth.p*dt/ME
    orion.pos = orion.pos + orion.p*dt/orion.m
def deflect(vo0,x,y):
    theta = x*pi/180
    orion.p = orion.m*vo0*vector(y,sin(theta),cos(theta))
while t<1e9:
    rate(4000)
    if mag(moon.pos-orion.pos)>1e7 and el==0:
        force()
    if mag(moon.pos-orion.pos)<1e7 and el==0:
        if dl==0:
            deflect(36,10,-3.5)
        dl=dl+1
    if dl==1:
        force()
    if t>1.85e6:
        el=1
        if sl==0:
    deflect(500,-90,0)
    sl=sl+1
        if sl==1:
            force()
            if t>2.228e6:
                break
    t = t + dt
```

OUTPUT:


Launch directed to moon


Entering moon’s orbit


Leaving moon’s orbit


Re-entering Earth

CODE EXPLAINED WITH SYNTAX

1. Physical Constants used in the program

a) Gravitational constant

The gravitational constant, frequently represented by the letter "G," is a fundamental physical constant that can be found in both Albert Einstein's general relativity theory and Sir Isaac Newton's law of universal gravitation. It serves as an indicator of how strong the gravitational pull is between two objects. Approximately **$6.67430\times10^{-11} m^3/kg\ s^2$** is the gravitational constant's value.

It's important to remember that the gravitational constant is a fundamental constant of nature, and that experiments and measurements have been used to determine the exact value of this constant. It plays a crucial role in understanding the behaviour of gravitational forces on both small and large scales systems.

Assigning value of gravitational constant: Program Statement 2

Real value of G=$6.67430\times10^{-11} m^3/kg$

Python syntax: G=6.67e-11

G is the variable assigned to represent Gravitational constant

b) Mass of Earth: Program Statement 3

The mass of the Earth is approximately **5.972×10^{24} kg.** This value represents the total amount of matter contained within the Earth, including its core, mantle, and crust. The mass of the Earth is a fundamental property that determines its gravitational pull on objects around it, including the moon and other satellites that orbit the planet.

Assigning value of Mass of earth:

Real value: Mass of earth=5.972×10^{24} kg

Python syntax: ME=5.97e24

ME is a variable that represents mass of earth

c) Radius of Earth: Program Statement 4

It is important to note that the Earth is not a perfect sphere but rather an oblate spheroid, meaning its shape is slightly flattened at the poles and bulging at the equator due to its rotation. The equatorial radius is about 21 kilometers (13 miles) larger than the polar radius, which contributes to the Earth's oblate shape.The mean radius of the Earth is approximately **6,371 kilometers** (3,959 miles). This value represents the average distance from the center of the Earth to its surface.

Assigning value of Radius of earth:

Real value: Radius of earth=6.371×10^{6} m

Python syntax: RE=6.37e6

RE is a variable that represents Radius of earth

c) **Mass of Moon:** **Program Statement 5**

The Moon, Earth's natural satellite, has a relatively small mass compared to our planet. With a mass of approximately 7.35 x 10^22 kilograms, the Moon is about 1/80th the mass of Earth. Despite its smaller size, the Moon's gravitational pull still affects our planet and plays a crucial role in phenomena such as ocean tides. The Moon's mass and its proximity to Earth have made it an object of fascination for scientists and a target for human exploration and space missions.

Assigning value of Mass of moon:

Real value: Mass of moon=7.35 x 10^22 kilograms

Python syntax: MM=7.348e22

MM is a variable that represents mass of moon

d) **Radius of moon:** **Program Statement 6**

The Moon, Earth's natural satellite, has a radius of approximately 1,737 kilometers. It is the fifth-largest moon in the Solar System and has a relatively small radius compared to other celestial bodies. Despite its smaller size, the Moon plays a significant role in Earth's tides, and its gravitational pull influences various phenomena on our planet. The Moon's radius is an essential factor in understanding its physical characteristics and its impact on Earth's environment.

Assigning value of Radius of earth:

Real value: Radius of earth=1.737× 10^6 m

RM is a variable that represents Radius of moon

Python syntax: RM=1.7371e6

e) **Distance between Earth and Moon:** **Program Statement 7**

The average distance between Earth and the Moon is approximately 384,400 kilometers (238,900 miles). This distance can vary slightly due to the Moon's elliptical orbit around the Earth. The Moon is Earth's only natural satellite and it takes about 1.3 seconds for light to travel from the Moon to Earth, making it the closest celestial body to our planet.

Assigning value of distance between Earth and Moon

Real value: Distance between moon and earth=384,400000

Python syntax: REM=384400e3

REM is a variable that represents distance between the two asteroids

f) **Mass of space craft:** **Program Statement 14**

Python syntax: orion.m =1e3

orion.m is a variable that represents Mass of Spacecraft

1. **Sphere function: Program Statement 8,9,13**

   Making 3d model of earth and asteroid

   Python syntax: #earth model

   earth=sphere(pos=vector(0,0,0), radius=3*RE,texture=textures.earth)

   Making a sphere at origin with 3 times radius of earth with a texture of earth

   Python syntax:#moon model

   moon = sphere(pos=earth.pos+REM*vector(1,0,0),radius=3*RM, make_trail=True))

   Making a sphere at (REM,0,0) in 3d space with radius of 3 times radius of moon and switching on trail property

   Python syntax:#orion model

   orion=sphere(pos=earth.pos+vector(0,-3.1*RE,0),radius=3*RM/2,color=color.yellow, make_trail=True)

   Making a sphere at (0,-3.1*RE,0) in 3d space with radius of 2.5 times moon with yellow colour and switching on trail property

2. **Initial Velocity : Program Statement 10,15**

   The initial velocity of the spacecraft was assigned to be 6200 m/s

   Assigning initial velocity of asteroids:

   Python syntax: v0 = sqrt(G*ME/REM)

   v0 is a variable that represents the initial velocity of moon

   Python syntax: vo0 =6.2e3

   vo0 is a variable that represents the initial velocity of Spacecraft

3. **Calculating initial momentum**

   Vector function in glowscript: Program Statement 11,12,17,16

   In GlowScript, the `vector()` function is a built-in function that creates and returns a vector object. Vectors are fundamental data types in GlowScript used to represent quantities with both magnitude and direction, such as positions, velocities, forces, and more. Here we will be calculating momentum using equation 2.

   Assigning initial momentum:

   $P_{ASTEROID} = M_{ASTEROID} \times V_{ASTEROID}$

   Python syntax : moon.p = MM*vector(0,v0,0)

   earth.p = -moon.p

   moon.p and earth.p are the variables that represents the momentum of moon and earth

Initial momentum to the spacecraft was assigned by

$P_{SPACECRAFT} = M_{SPACECRAFT} \times V_{SPACECRAFT}$

Python syntax : orion.p=orion.m*vo0*vector(cos(theta),sin(theta),0)

theta = -10*pi/180 #Angle

orion.p is a variable that represents the momentum of orion space craft

- **Making Force Function:**

1. **Defining Force calculation function: Program Statement 22**

   Python syntax : def force():

2. **Find the position vector r (position vector between two bodies): Program Statement 23,24,25**

   Position vector between two bodies = Position vector of 1st body – Position vector of 2nd body

   Python syntax : rem = moon.pos- earth.pos

   reo = orion.pos - earth.pos

   rmo = orion.pos - moon.pos

   rem is a variable that represents the Position vector between moon and earth

   reo is a variable that represents the Position vector between orion and earth

   rmo is a variable that represents the Position vector between moon and orion

3. **Calculate the gravitational force F_G using r using equation 1: Program Statement 26,27,28**

   Universal law of Gravitation

   Python syntax : Fem = -G*MM*ME*norm(rem)/mag(rem)**2

   Feo = -G*ME*orion.m*norm(reo)/mag(reo)**2

   Fmo = -G*MM*orion.m*norm(rmo)/mag(rmo)**2

   Fem is the variable that represents the gravitational force between earth and moon

   Feo is the variable that represents the gravitational force between earth and orion

   Fmo is the variable that represents the gravitational force between moonand orion

   norm(rem), norm(reo) , norm(rmo) represents the unit vector of rem ,reo,rmo and mag(rem), mag(reo) , mag(rmo) represents the magnitude of rem , reo , rmo

4. **Calculate/Update using Momentum: Program Statement 29,30,31**

   $\mathbf{p}_{new} = \mathbf{p}_{initial} + \mathbf{F} \times \mathbf{dt}$

   Python syntax : moon.p = moon.p + Fem*dt

   earth.p = earth.p -Fem*dt

   . orion.p = orion.p + (Feo + Fmo)*dt

   Fem,Fmo and dt are the variable that represents the gravitational force and small change in time

5. **Finding the new position using the updated momentum: Program Statement 32,33,34**

   $(\mathbf{x}_{\mathbf{final}} = \mathbf{x}_{\mathbf{initial}} + \left(\frac{\Delta \mathbf{p}}{(\mathbf{m})}\right) \times \Delta \mathbf{t})$

   Python syntax : moon.pos = moon.pos + moon.p*dt/MM
   earth.pos = earth.pos + earth.p*dt/ME
   orion.pos = orion.pos + orion.p*dt/orion.m

   moon.pos is the variable that represents the the position of moon

   earth.pos is the variable that represents the the position of earth

   orion.pos is the variable that represents the the position of orion

   dt is the variable that represents the small change in time

- **Making a function that will be used for spacecraft entry and exit to the moon orbit**

1. **Defining Deflection calculation function: Program statement 35**

   Python syntax : def deflect(vo0,x,y):
2. **Calculating deflection angle: Program statement 36**

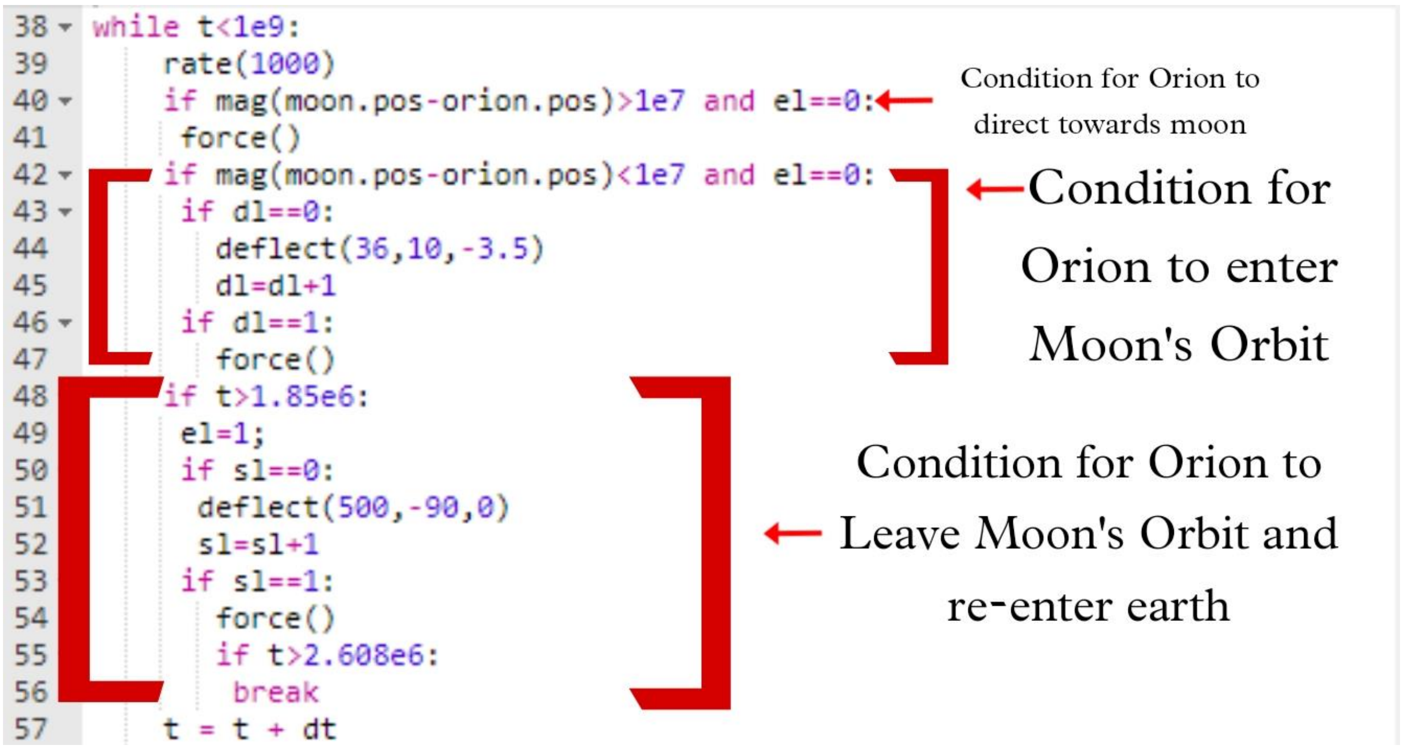
   Python syntax : theta = x*pi/180
3. **Combining the deflection angle with momentum to change Orion's path: Program statement 37**

   Python syntax : theta = x*pi/180

- **Loop: To repeat specific action, while loop has been used : Program statement 38**

  Python syntax : while t<1.0e9:

  The loop will be executed until the value of t is less than 1×10^9 seconds



```
38  while t<1e9:
39      rate(1000)
40      if mag(moon.pos-orion.pos)>1e7 and el==0:
41       force()
42      if mag(moon.pos-orion.pos)<1e7 and el==0:
43       if dl==0:
44         deflect(36,10,-3.5)
45         dl=dl+1
46       if dl==1:
47         force()
48      if t>1.85e6:
49       el=1;
50       if sl==0:
51        deflect(500,-90,0)
52        sl=sl+1
53       if sl==1:
54         force()
55         if t>2.608e6:
56          break
57      t = t + dt
```

Condition for Orion to direct towards moon
Condition for Orion to enter Moon's Orbit
Condition for Orion to Leave Moon's Orbit and re-enter earth

a) **Updating the time: Program statement 57**

$t_{new} = t_{initial} + dt$

Python syntax : t = t + dt

t is the variable that represents time

dt is the variable that represents the small change in time

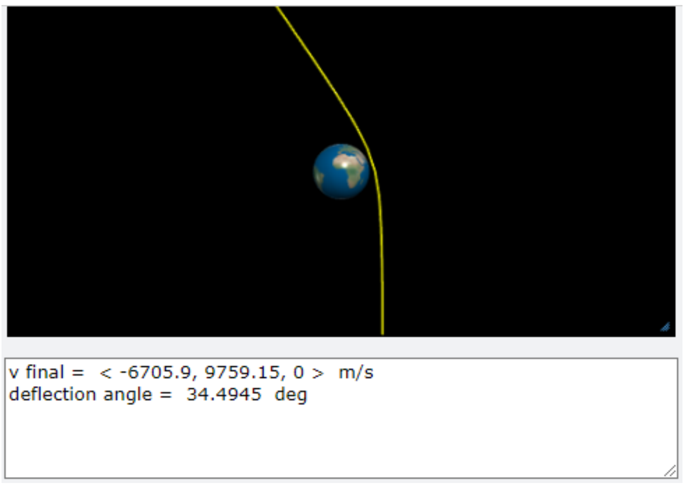
RESULTS AND DISCUSSION

1. VISUALIZING THE NEAR-EARTH ASTEROID & CALCULATING DEFLECTION ANGLE:

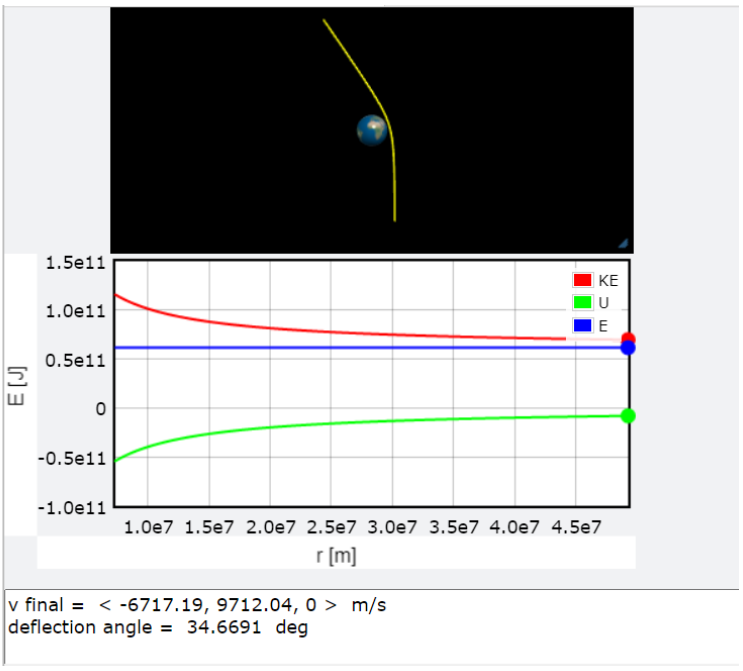
In this program the stimulation of a near earth asteroid has been done ,basic idea and codes were borrowed from a tutorial provided by Allain Rehtt [3] , the report consist of detail study of code and also analysis of the physics concept in brief, also further extended the program by adding the graphs that shows the change in energies of the asteroid as it passes nearby earth.

Final Output of reference: [3]

The output shows the trajectory of a near earth asteroid, where the yellow line represents the path followed by the asteroid and the blue ball at the centre represents earth. This program also calculates the final velocity of asteroid as well as the deflection angle.



```
v final =  < -6705.9, 9759.15, 0 >  m/s
deflection angle =  34.4945  deg
```



```
v final =  < -6717.19, 9712.04, 0 >  m/s
deflection angle =  34.6691  deg
```

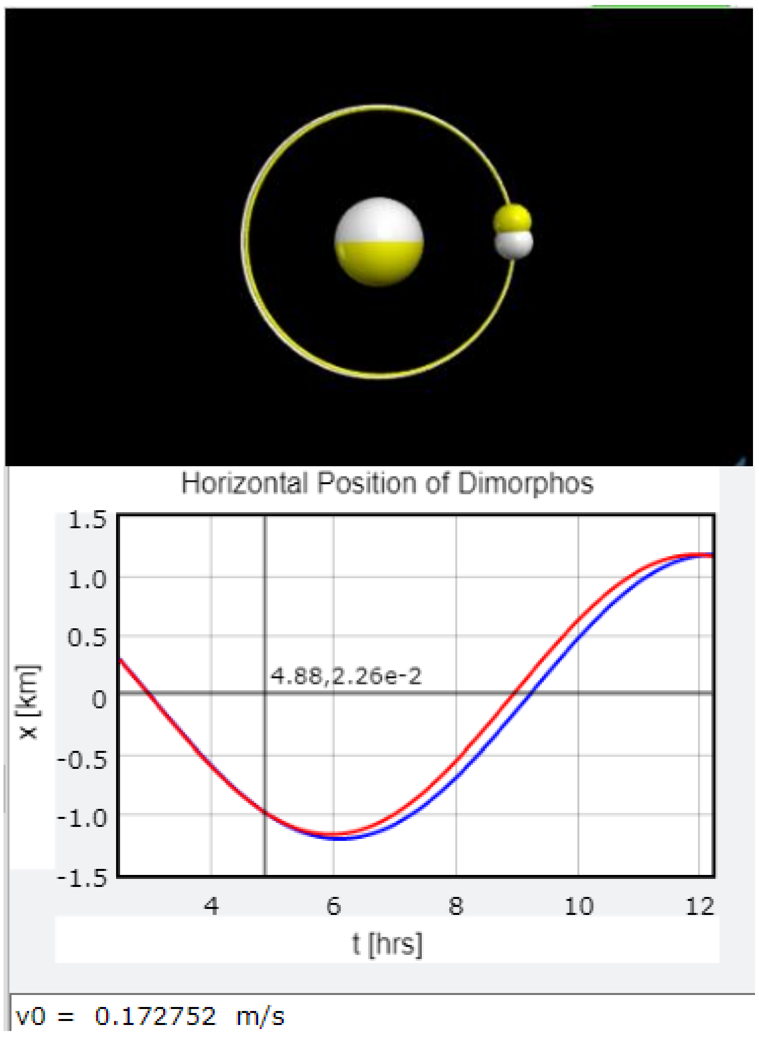
Modified Output with Energy graph:

Now further addition of energy diagram has been made to this program where calculation of Kinetic energy, Potential energy and Total Energy has been added and plotted in graph which will represent the change in energy of the asteroid as the it passes near-by earth. Red line on graph represents kinetic energy, blue line on graph represents total energy and green line on the graph represents potential energy.

2. REPLICATING DART: DOUBLE ASTEROID REDIRECTION TEST:

In this program the stimulation of Dart mission has been done, the basic idea and codes were borrowed from Allain Rhett [1] where the detail study of code and analyzed the physics concept in brief. From the tutorial provided by Allain Rhett [1] , the output consist of stimulation that shows the change in orbit of the dimorphos and an graph that also portrays the same and final velocity of Dimorphos has also been calculated. In this report the program has been further extended where a moving twin asteroid system has been stimulated to which a dart space craft collides and two final paths has been shown where one represents the path that asteroid will take without collision and the other it will take with collision , deflection angle of the asteroid and change in time period of dimorphos has also been calculated.

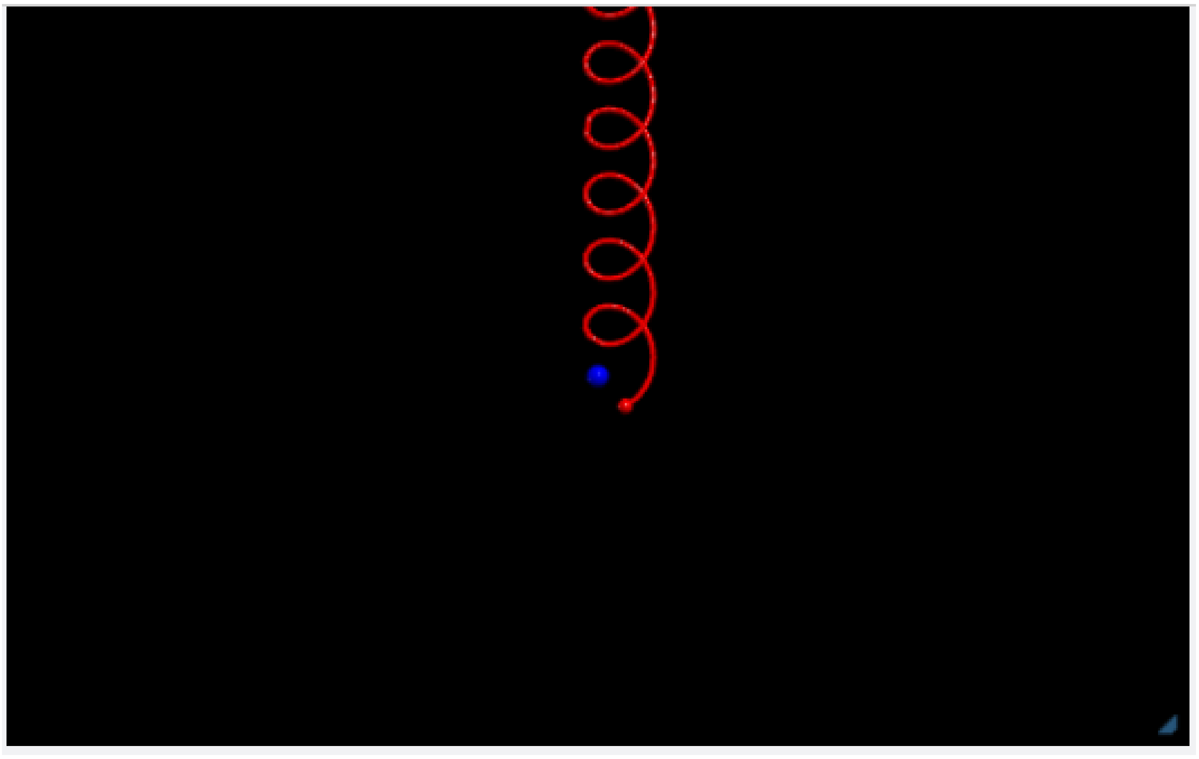
Final Output of reference: [1]



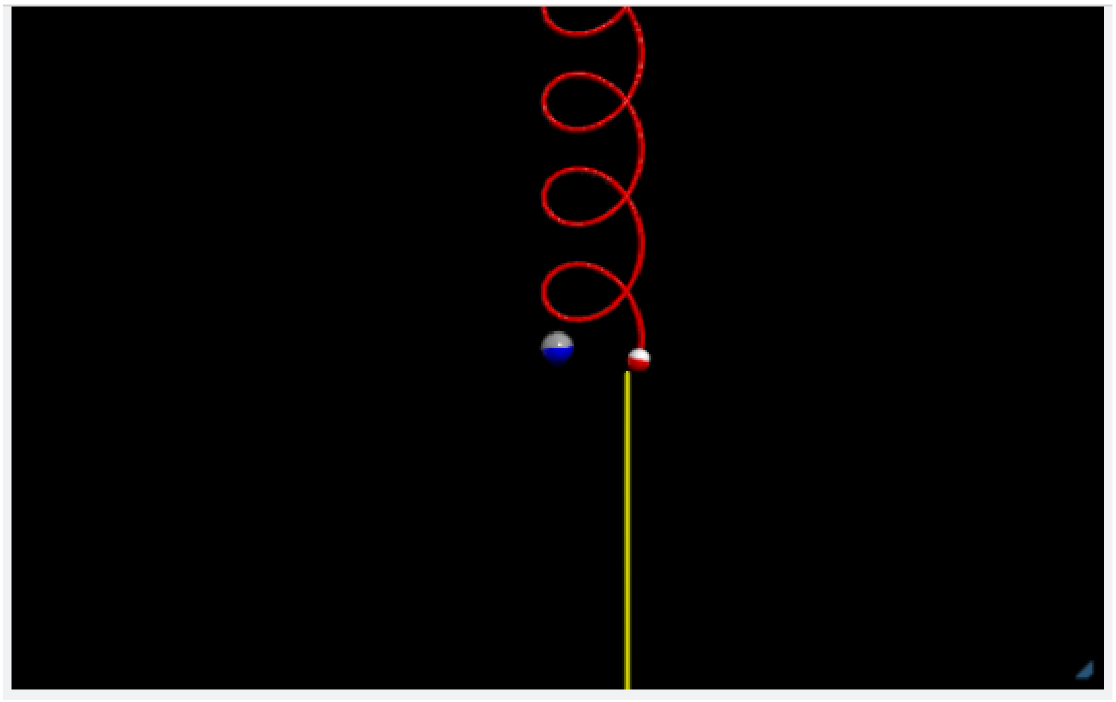
```
v0 =  0.172752  m/s
```

The output shows two systems together , the white one is the asteroid system without collision where as the yellow one represents the asteroid system with collision ,a minor change in the orbit of dimorphos is observed. The represents the position of dimorphos at specific time, it also explains the change in orbit of dimorphos because of collision as the red line represents the path of dimorphos without collision and the blue line represents the path of dimorphos with collision. Finally the program calculates the final velocity of Dimorphos.

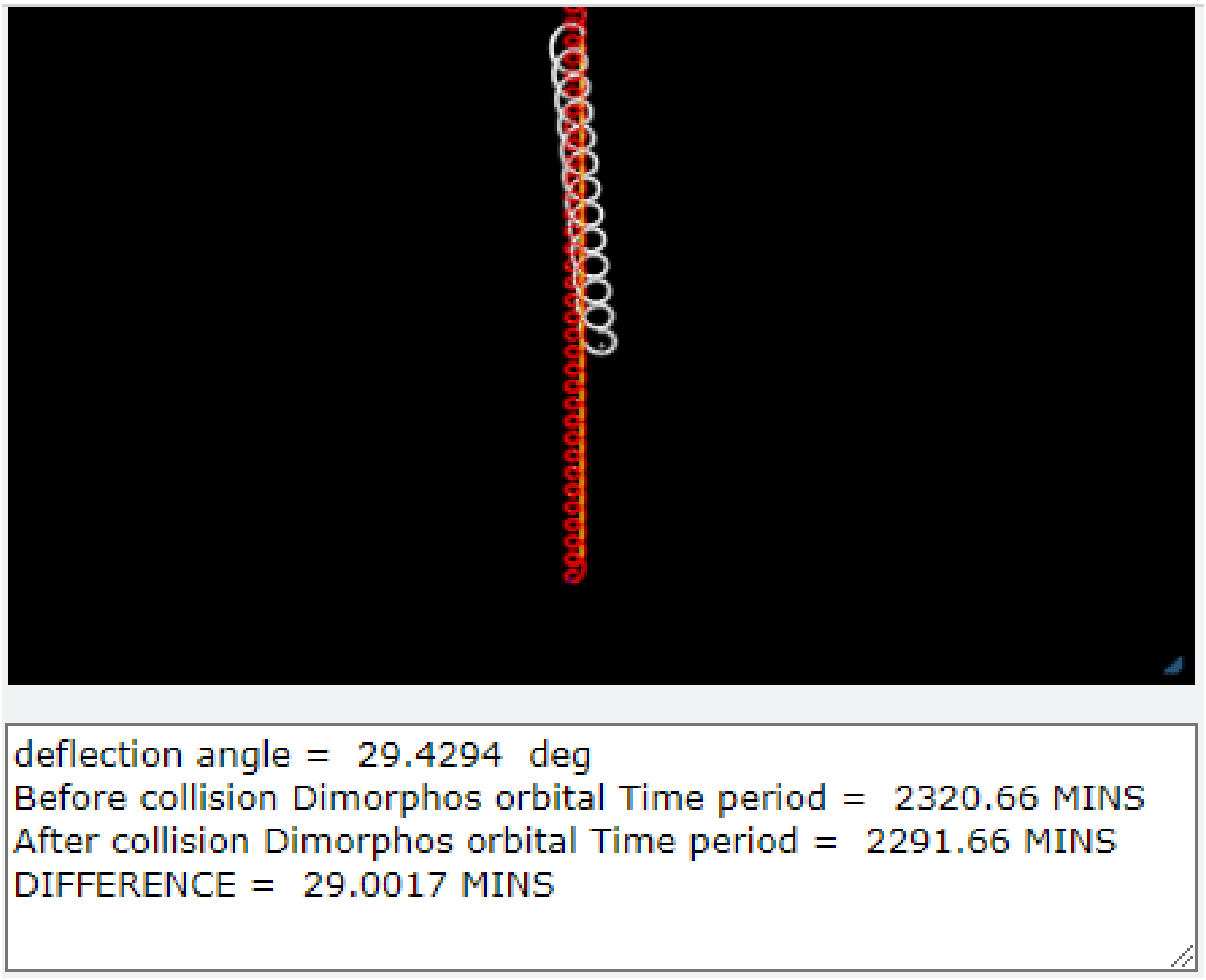
6. Modified output:



Straight path of Didymos(Blue ball) and Dimorphos(Red ball) system in space. Red spiral denotes the trail/path of Dimorphos around Didymos as the system is moving forward.



Time of collision of spacecraft with Dimorphos, yellow line indicates the trail/Path of Dart spacecraft.



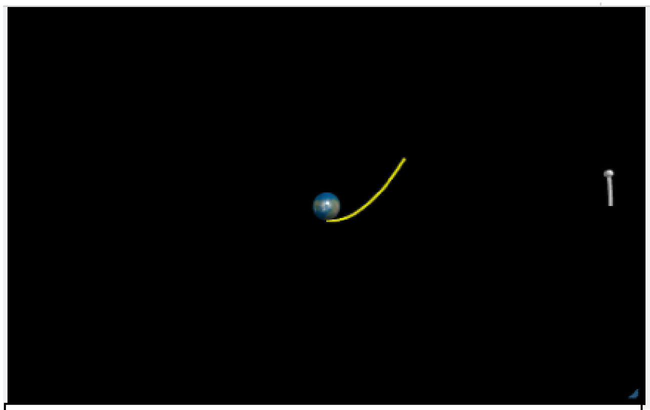
```
deflection angle =  29.4294  deg
Before collision Dimorphos orbital Time period =  2320.66 MINS
After collision Dimorphos orbital Time period =  2291.66 MINS
DIFFERENCE =  29.0017 MINS
```

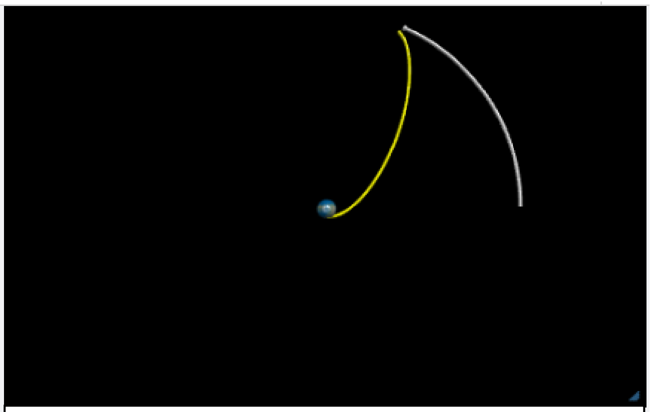
Change in path of the system after the collision with the spacecraft is denoted with the white spiral, while the red spiral denotes the path which the asteroid system should have followed if there was no collision. Change in path of the asteroid system was observed in the stimulation. Calculation of deflection angle was added which will show how much the asteroid sytem has deflected from its real path after collision , calculation of dimorphos orbital time period was also added where the time period before and after collision was calculated and finally the change in time period was aslo displayed.

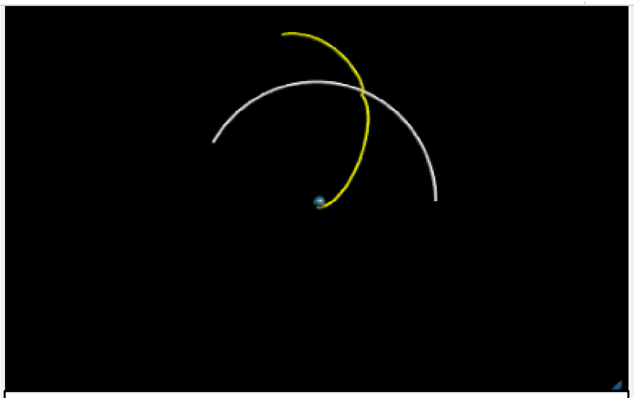
3. REPLICATING ARTEMIS 1 (EXPLORATION MISSION-1 (EM-1)):

In this program the stimulation of Artemis-1 mission has been done, the basic idea and codes were borrowed from Allain Rhett [2]where the detail study of code and analyzed the physics concept in brief. From the tutorial provided by Allain Rhett [2] , the output consisted of Orion spacecraft accurately directed towards moon from earth. In this report the program has been further extended where Orion spacecraft is able to enter moon's orbit (lunar fly by) and leave moon's orbit and re-enter earth. To summarize the updated program were able to completely replicate Orion's journey in Artemis-1 mission.

Output of reference: [2]

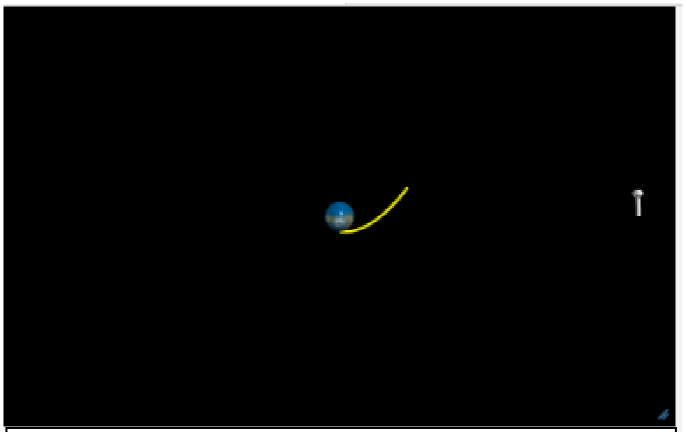

1.Launch directed to moon

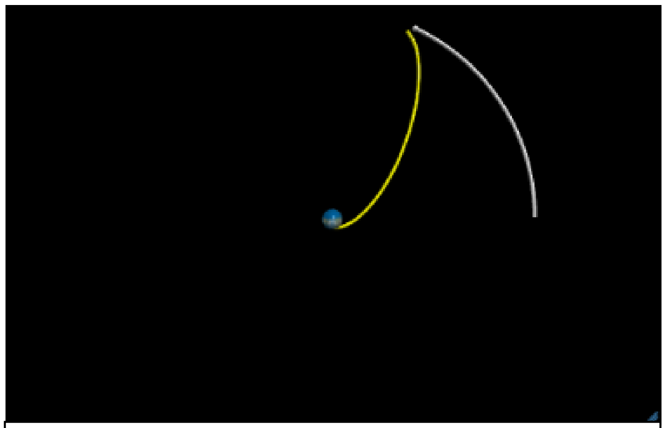

2.Reaching moon's orbit

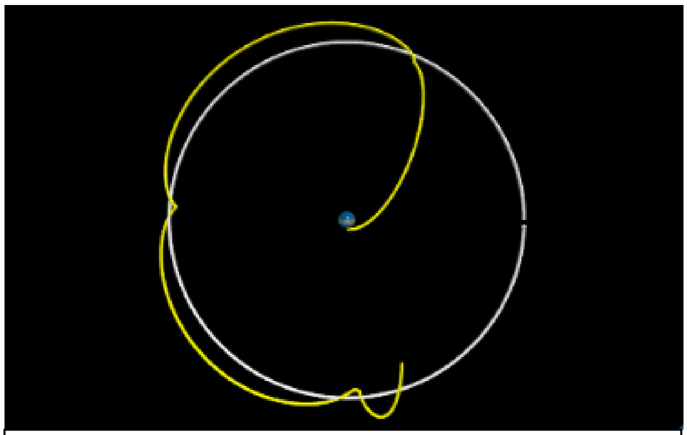

3.Leaving moon's orbit

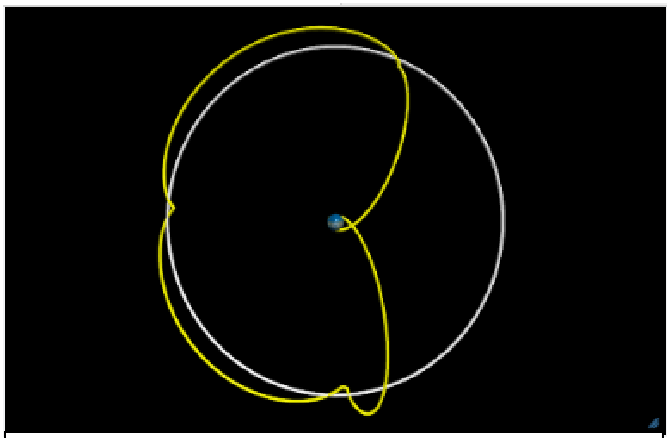
The output shows the trajectory of Orion spacecraft accurately launched to moon, where the yellow line represents the path followed by the Orion spacecraft, white line represents the path followed by moon around earth and the blue ball at the centre represents earth. The program is able to stimulate Orion's path and direct it towards moon. Here the 1st photo represents the launch of Orion towards moon from earth , the 2nd photo represents Orion reaching moon's orbit and 3rd photo represents Orion leaving moon's orbit away from earth.

Modified output:


1.Launch directed to moon


2.Entering moon's orbit


3.Leaving moon's orbit


4.Re-entering Earth

Further extending the program the successful stimulation of Orion's path that it took in the Artemis-1 mission was achieved. The output shows the trajectory of Orion spacecraft accurately launched to moon, where the yellow line represents the path followed by the Orion spacecraft, white line represents the path followed by moon around earth and the blue ball at the centre represents earth. The program is able to stimulate Orion's path and direct it towards moon. Here the 1st photo represents the launch of Orion towards moon from earth, the 2nd photo represents Orion reaching moon's orbit, 3rd photo represents Orion leaving moon's orbit towards earth and 4th photo represents Orion re-entering earth's atmosphere i.e. the complete 4 stages of ARTEMIS-1 mission.

• REFERENCES

[1] R. Allain, “Modeling the DART-Dimorphos Collision in Python,” [Online]. Available: https://youtu.be/1-Ha0s08x_c.

[2] R. Allain, “Modeling the motion of Artemis 1 with Python and Physics,” [Online]. Available: https://youtu.be/tL1Il6If19c.

[3] R. Allain, “Visualizing the Near Earth Asteroid 2020 QG with Python,” [Online]. Available: https://youtu.be/LeUIrZDlD_4.

[4] E. S. Agency, “Safe but very close approach of small asteroid 2023 BU,” [Online]. Available: https://www.esa.int/ESA_Multimedia/Images/2023/01/Safe_but_very_close_approach_of_small_asteroid_2023_BU.

[5] “DART: MISSION INDEX,” [Online]. Available: https://dart.jhuapl.edu/Mission/index.php.

[6] NASA, “Didymos & Dimorphos,” [Online]. Available: https://solarsystem.nasa.gov/asteroids-comets-and-meteors/asteroids/didymos/in-depth/.

[7] NASA, “The Science Behind NASA's First Attempt at Redirecting an Asteroid,” [Online]. Available: https://www.jpl.nasa.gov/edu/news/2022/9/22/the-science-behind-nasas-first-attempt-at-redirecting-an-asteroid/#:~:text=In%20a%20successful%20attempt%20to,no%20threat%20to%20our%20planet..

[8] NASA, “ARTEMIS I PRESS KIT,” [Online]. Available: https://www.nasa.gov/specials/artemis-i-press-kit/.

[9] NASA, “NASA Space Science Data Coordinaated Archive,” [Online]. Available: https://nssdc.gsfc.nasa.gov/nmc/spacecraft/display.action?id=2021-110A.

[10] NASA, “ARTEMIS 1 MAP,” [Online]. Available: https://www.nasa.gov/image-feature/artemis-i-map.

[11] NASA, “Space Launch System Block 1 Crew Vehicle Expanded View,” [Online]. Available: https://www.nasa.gov/exploration/systems/sls/multimedia/gallery/sls_config_70t.html.

[12] “Vpython-Documentation,” [Online]. Available: https://www.glowscript.org/docs/VPythonDocs/index.html.

[13] NASA, “NASA, SpaceX Launch DART: First Planetary Defense Test Mission,” [Online]. Available: https://blogs.nasa.gov/dart/.

[14] NASA, “NASA’s DART Mission Hits Asteroid in First-Ever Planetary Defense Tes,” [Online]. Available: https://www.nasa.gov/press-release/nasa-s-dart-mission-hits-asteroid-in-first-ever-planetary-defense-test.

[15] N. STEM, “NASA’s DART Mission: Learning to Defend Earth from Asteroid Impacts,” [Online]. Available: https://youtu.be/DIN6XJOUV4g.

[16] NASA, “ARTEMIS 1 - Resource reel (Photos took by orion),” [Online]. Available: https://images.nasa.gov/search?page=1&media=image&yearStart=1920&yearEnd=2023&keywords=Artemis%20I%20Resource%20Reel.

WHAT I LEARNED FROM THE INTERSHIP

- How to read and analyse and give presentation on a particular topic.
- Helped me to know about the multidisciplinary approaches that is absent in the academic field.
- Studied about python
- Get to know about the importance of Python in Physics
- Studied Python glow script where I got to know about stimulations
- Studied the importance of algorithm development and understood how algorithm can be developed
- Studied how to use physics concept in coding to stimulate natural phenomena
- Studied about different Space missions in detail and understood the physics involved behind it