

Developing a custom flight simulation for a mission from Earth to Triton

Martin Rupp

SCIENTIFIC AND COMPUTER DEVELOPMENT SCD LTD

[Developing a custom flight simulation for a mission from Earth to Triton](#)

[Introduction](#)

[Scope of the simulation](#)

[Reminder about gravity-assisted propulsion principles](#)

[General principle](#)

[N-body problem](#)

[Obtaining NASA data for planets' orbits](#)

[Numerical computation of planet orbits](#)

[Heliocentric coordinate reference](#)

[Solar Ecliptic Coordinate System \(SE\)](#)

[Heliographic Inertial Coordinate System \(HGI\)](#)

[Heliographic \(rotating\) Coordinate System \(HG\)](#)

[Trajectory computation](#)

[Lambert Solver](#)

[Choice of the graphical engine](#)

[Development of the simulation](#)

[Preparation of the mission](#)

[The first stage of the mission](#)

[Second stage of the mission](#)

[Conclusion](#)

[REFERENCES](#)

Introduction

What are the techniques used to simulate orbital flight? How do NASA and other space agencies compute the parameters of a flight and how do they simulate space missions? There are several orbital simulation software which are in open access. For example, the General Mission Analysis Tool ([GMAT](#)).

In this small article, we want to show the steps for the development of a basic orbital flight simulator that we will write in C#. GMAT is a very complete tool but it doesn't have parameters for Triton and developing a plug-in for GMAT isn't straightforward.

Scope of the simulation

In this scenario, we simulate sending a CubeSat from Earth to Triton. As a first step, a heavy launcher sends the mothership to Neptune, using gravity-assisted propulsion. The second step consists of the mother ship detaching the CubeSat as she is injected into Neptune's orbit, computing the orbit transfer, and sending the Cubesat into Triton's orbit. The goal of the mission is to observe Triton from its orbit.

Reminder about gravity-assisted propulsion principles

General principle

In the same a boat can navigate, without being self-propelled, in a river, just by following the flow of the water stream, a spaceship can navigate in space without being self-propelled, just by following the flow of the gravitation stream. In both cases, this is a force field that acts on the ship and moves it through space. Self-propulsion is also required, in both cases, at the start and end of the journey. For the boat, it's required to leave the dock when the trip starts by using engines and, reversely, to leave the river flow when approaching the destination port. For the ship, engines are needed for leaving Earth (or any dock in general) during a boost phase and, at the end of the trip, to leave the gravitation field to be injected into the orbit of the destination.

One of the main differences between navigation in a water body and space is that, in space, apart from Gravitation, there are generally no matter, no viscous forces that will slow down the ship.

The ship's mass is infinitesimal in space compared to the cosmic bodies around her so it can 'freely' jump from one orbit to another, 'playing' with the different planets or moons trajectories and the corresponding gravitation fields.

In the context of the solar system, which is the context of the simulation, the dominant gravitation field is the Sun's. Most planets orbit around the sun, roughly in the same two-dimensional plane, except Neptune which orbits in a slightly inclined plane.

Therefore, a spaceship would primarily rely on the sun's attraction to navigate from one planet to another, computing adequately the trajectory to coincide with the trajectory of the target planet at the rendezvous point. Indeed, the trajectories of the solar system planets are rather almost 100% predictable and not subject to fluctuational changes. The moons of the solar system may be potentially much less predictable, at least some of them.

N-body problem

The problem of determining the trajectory of a spaceship when considering gravity-assisted navigation is related to the n-body problem.

If we name S the spaceship then, its motion according to a celestial body (planet or moon) C of the solar system is ruled by the 2-body problem. The 2-body problem can be broken down into two distinct 1-body problems by considering the barycentre G of the ponderated points $(S, m(S))(C, m(C))$.

In the case of a spaceship, it usually $m(S) \ll m(C)$ G coincides concretely with C .

It is well known that in such a case, trajectories shall be conic: either elliptic, parabolic, or hyperbolic.

Using Newton's principles, the equation of a spaceship in motion (eq1) is :

$$\ddot{\vec{r}} = \mathcal{G}m(C)/|\vec{r}|^2 \vec{u}_r$$

\mathcal{G} is Newton's constant

\vec{r} is the position vector of the spaceship in the referential of C

$$\vec{u}_r = \frac{1}{|\vec{r}|} \vec{r}$$

This equation is the basis of all the subsequent computations for the parameters of the trajectories.

Obtaining NASA data for planets' orbits

Numerical computation of planet orbits

The orbit of each planet in the solar system can be computed using Kepler's formula. However, in the context of that simulation, we preferred using existing data from NASA. The *Horizons System*, maintained by the JPL, contains a comprehensive database of coordinates of celestial objects (past, present, and future).

[Heliocentric Trajectories for Selected Spacecraft, Planets, and Comets \(nasa.gov\)](https://ssd.jpl.nasa.gov/horizons/)

Heliocentric coordinate reference

The data that we choose to use are heliocentric coordinate systems. Heliocentric simply means that the referential of the trajectory is the referential created by the sun with the following axes:

Solar Ecliptic Coordinate System (SE)

The SE is a heliocentric coordinate system with the Z-axis normal to and northward from the ecliptic plane. The ecliptic plane is defined as the imaginary plane containing the Earth's orbit around the sun. The X-axis extends toward the first point of Aries (Vernal Equinox, i.e. to the Sun from Earth on the first day of Spring). The Vernal equinox is defined as one of the two moments in the year when the Sun is exactly above the Equator and day and night are of equal length, the other one occurring in Autumn. At these points the ecliptic plane and the celestial equator intersect. The Y-axis completes the right-handed set.

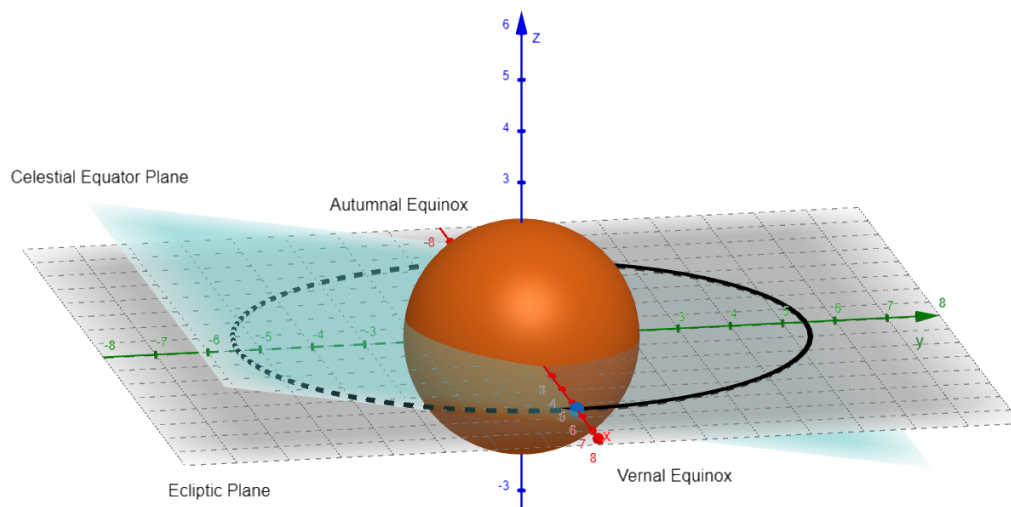


Illustration 1: SE coordinate system

The position of a point in space is defined by the following coordinates:

- RAD_AU (radius)
- SE_LAT (ecliptic latitude)
- SE_LONG (ecliptic longitude)

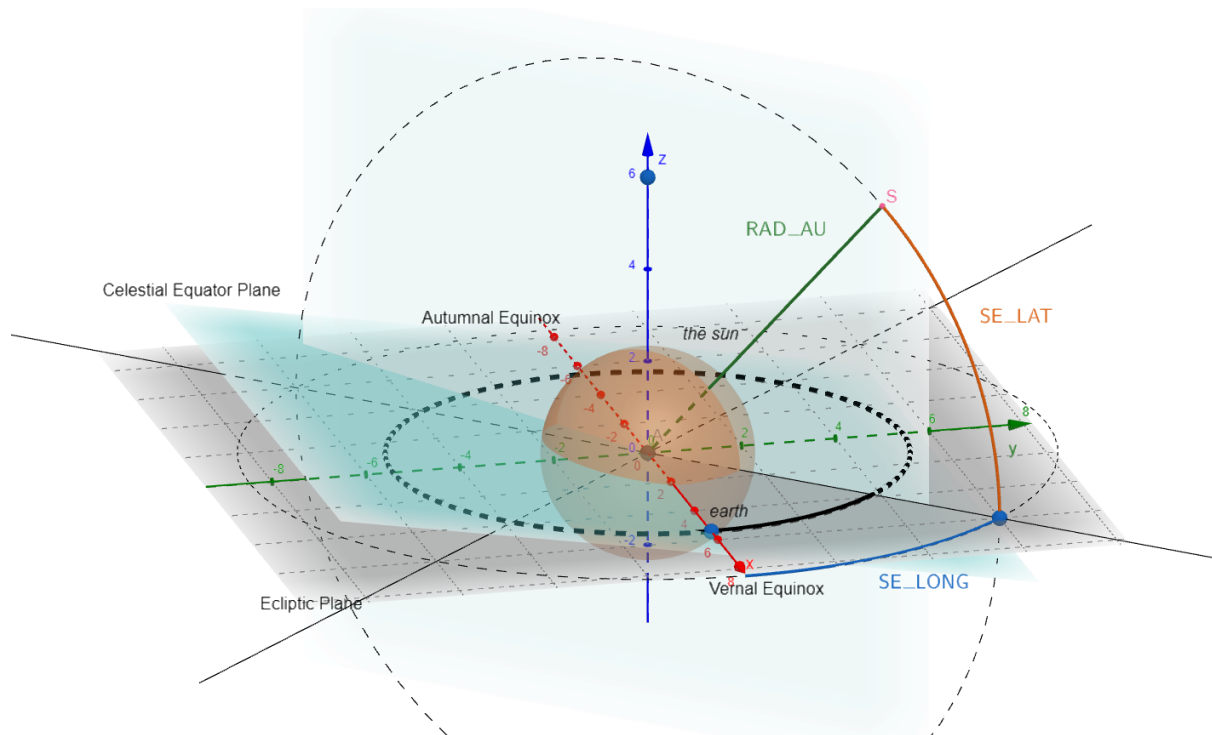


Illustration 2: SE coordinate system

Heliographic Inertial Coordinate System (HGI)

The HGI coordinates are Sun-centred and inertially fixed concerning an X-axis directed along the intersection line of the ecliptic and solar equatorial planes, and defines zero of the longitude.

Heliographic (rotating) Coordinate System (HG)

HG differs from HGI only in the sense that the zero of the longitude, HG_LONG is fixed on the Sun and (by convention) rotates at the fixed period of 25.38 days

Trajectory computation

Lambert Solver

Lambert's problem is the determination of an orbit from two position vectors and the time of flight. This consists of finding solutions $t \rightarrow r(t)$ of eq1 (defined earlier) with boundary conditions r_1, r_2, T .

- $r(t)$ is a solution of eq1
- $r(t_1) = r_1, r(t_2) = r_2$
- $t_2 - t_1 = T$

This problem is not trivial and there are no exact solutions for it. Any algorithm that provides a method to find numerical solutions for it is called a **Lambert's solver**.

Such solvers are routinely used by space agencies to determine the trajectories of spaceships for interplanetary missions. They vary in efficiency and precision.

In our simulator, we use a Lambert Solver based on the Matlab code available in [1], Appendix D.12.

The Lambert Solver always determines two possible trajectories: one *prograde* and the other *retrograde*.

Choice of the graphical engine

There are a lot of graphical engines that can be used for the simulation, for example, Unreal Engine or Unity3D. We preferred something more simple and we chose a simple but efficient graphical engine using GDI and available as a winform component: [Editor3D](#). Editor3D and Graph3D are available as a 3D surface plot control and can be [downloaded from Code Project](#).

This component doesn't use hardware acceleration code therefore the number of polygons that are displayed will determine the drawing speed.

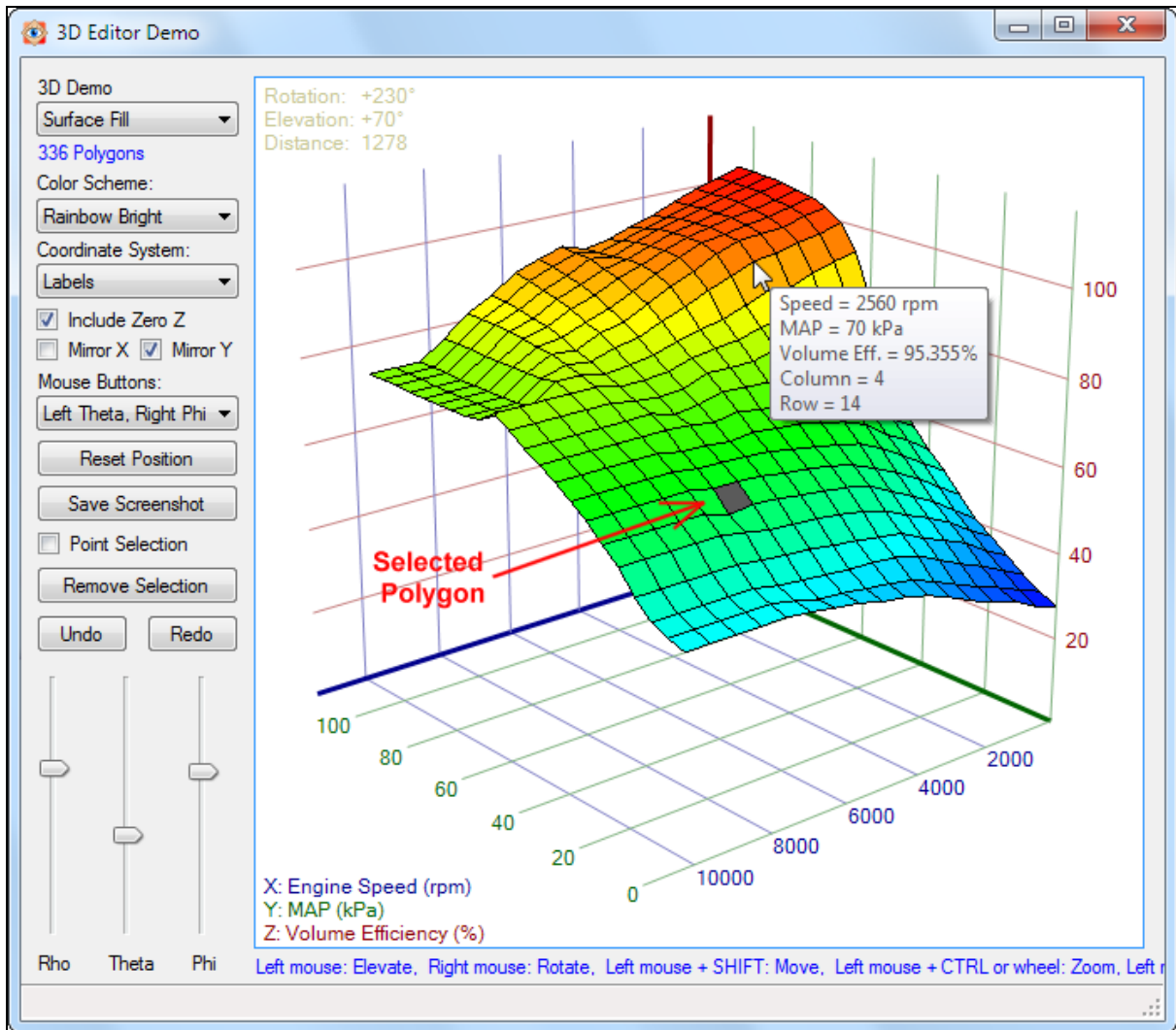


Illustration 3: Graph3D graphical engine component

This component has enough functionalities for our small simulation project. We will not be able to use texture or complex 3D effects but we will be able to render the orbits of the planets and moons of the solar system in the chosen referential and plot the trajectory of the mother ship as well as the trajectory of the satellite. The component is available as an open-source component for C# and the .NET platform, which means that we can eventually modify and recompile the graphical engine if we need to.

Development of the simulation

Our simulation uses Winforms and implements libraries and algorithms described in [1]. We separate the software into three UI components :

- **Preparation of the mission:** definition of the mission parameters
- **First stage of the mission:** segment Earth Orbit - Neptune Orbit
- **Second stage of the mission:** segment Neptune Orbit - Triton Orbit

Preparation of the mission

Lambert Solver

Dates

Earth Departure day and year
Day 1 of 1970

Neptune Arrival day and year
Day 1 of 1970

Neptune Orbit Departure (T=0 at arrival on Neptune)
Sec 0 Hour 0

Triton Orbit Arrival (T=0 at arrival on Neptune)
Sec 0 Hour 0

Departure

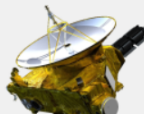
Velocity Vector (km/sec)
X= Y= Z=

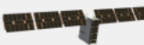
Velocity (km/sec)
Impulse (N x sec)

Arrival

Velocity Vector (km/sec)
X= Y= Z=

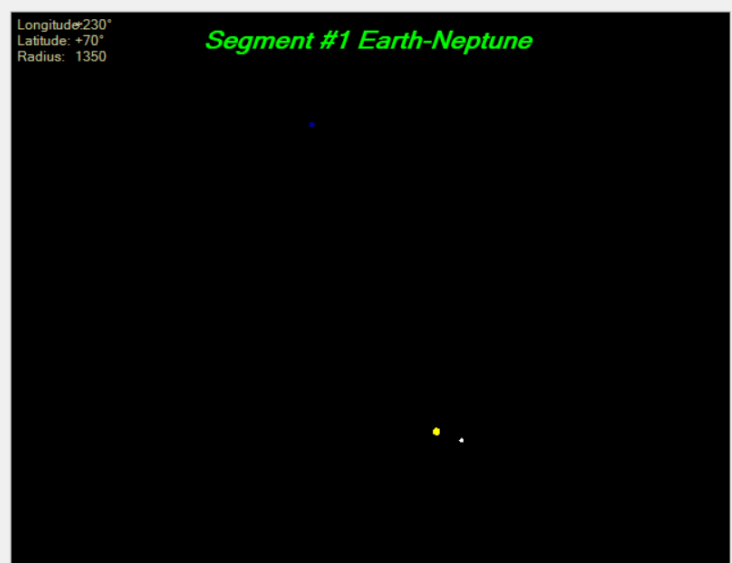
Velocity (km/sec)
Impulse (N x sec)

Mothership

Mass (kg) 478

Cubesat

Mass (kg) 1

Longitude: 230°
Latitude: +70°
Radius: 1350

Segment #1 Earth-Neptune



Earth departure coordinates (AU): (-0.17, 0.96, 0)
Neptune arrival coordinates (AU): (-15.74, 25.9, 0.9)

Retrograde Trajectory

Cubesat Neptune Orbit: 30000km+1000 * x km

Initial angle Triton-Cubesat at Neptune arrival t=0hrs (0-360 degree)

Illustration 4: Preparation of segment#1

Lambert Solver

Dates

Earth Departure day and year
Day 1 of 2024

Neptune Arrival day and year
Day 1 of 2029

Neptune Orbit Departure (T=0 at arrival on Neptune)
Sec 0 Hour 0

Triton Orbit Arrival (T=0 at arrival on Neptune)
Sec 0 Hour 0

Departure

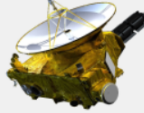
Velocity Vector (km/sec)
X= -12.0667830561059
Y= -48.0033566148704
Z= 0.529546445858381


Velocity (km/sec)
49.4995951403227
Impulse (N x sec)
23660806.4770743

Arrival

Velocity Vector (km/sec)
X= 25.8991161957223
Y= 4.30399237383752
Z= -0.686445653872101

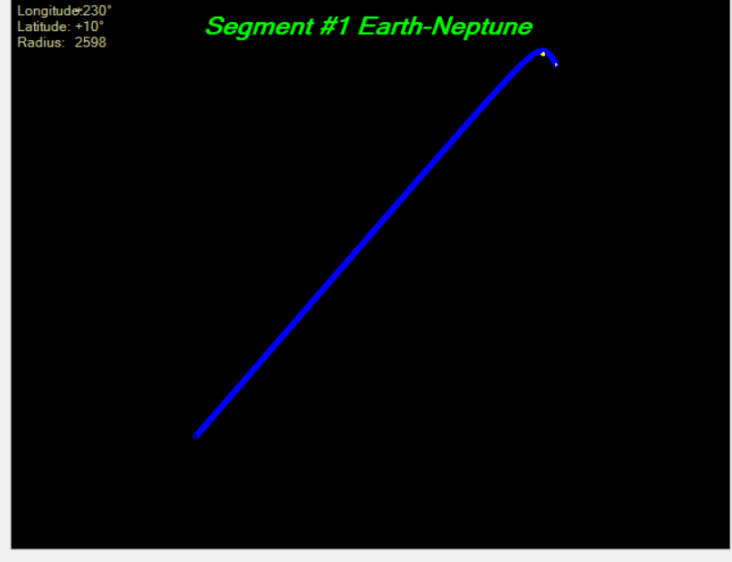
Velocity (km/sec)
26.2632781219195
Impulse (N x sec)
12553846.9422775

Mothership

Mass (kg) 478

Cubesat

Mass (kg) 1

Longitude: 230°
Latitude: +10°
Radius: 2598

Segment #1 Earth-Neptune



Earth departure coordinates (AU): (-0.17, 0.96, 0)
Neptune arrival coordinates (AU): (29.56, 4.15, -0.78)

Retrograde Trajectory

Cubesat Neptune Orbit: 30000km+1000 * x km

Initial angle Triton-Cubesat at Neptune arrival t=0hrs (0-360 degree)

Illustration 5: Solving trajectory for segment#1

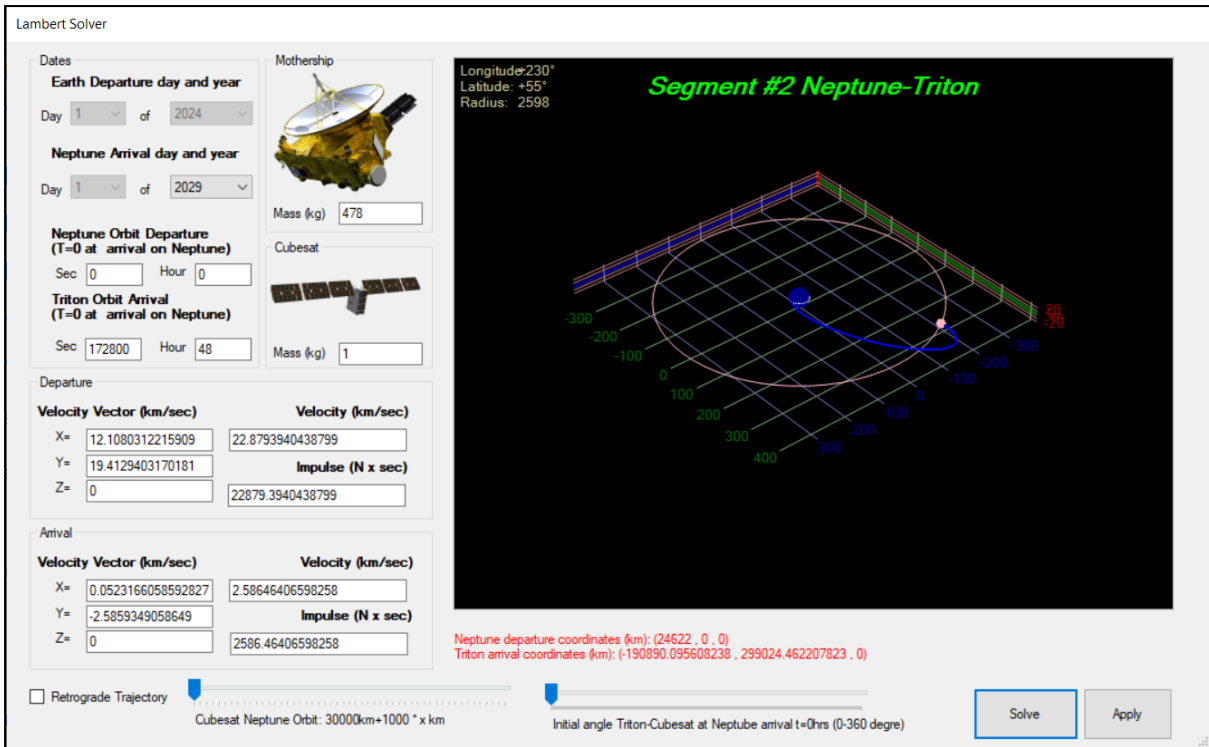


Illustration 6: Solving trajectory for segment#2

The first stage of the mission

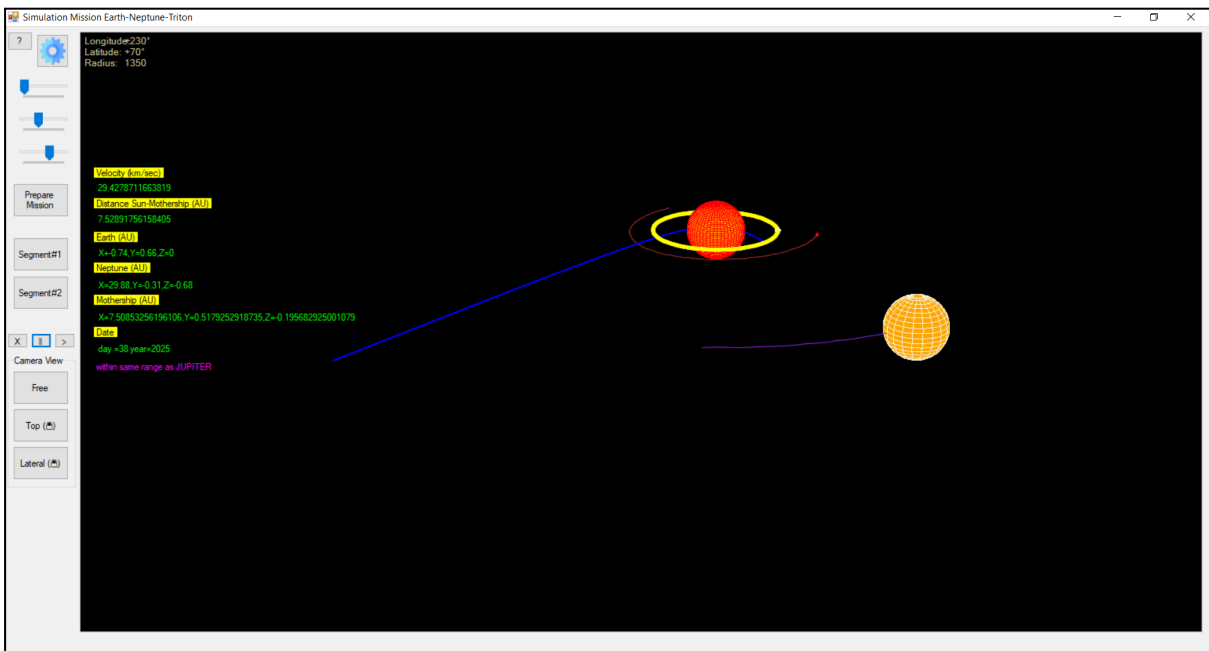


Illustration 7: Departure of the mothership from Earth orbit

The simulation of the segment Earth-Neptune offers various data and information:

- Distance of the spaceship from the center of the sun (in AU units)
- Velocity of the spaceship

- Coordinates of the spaceship in the SE system
- Current date
- Display of the planets of the solar system involved in the segment and their orbits
- Changing simulation speed
- Control for the simulation (pause, resume)
- Choose camera view

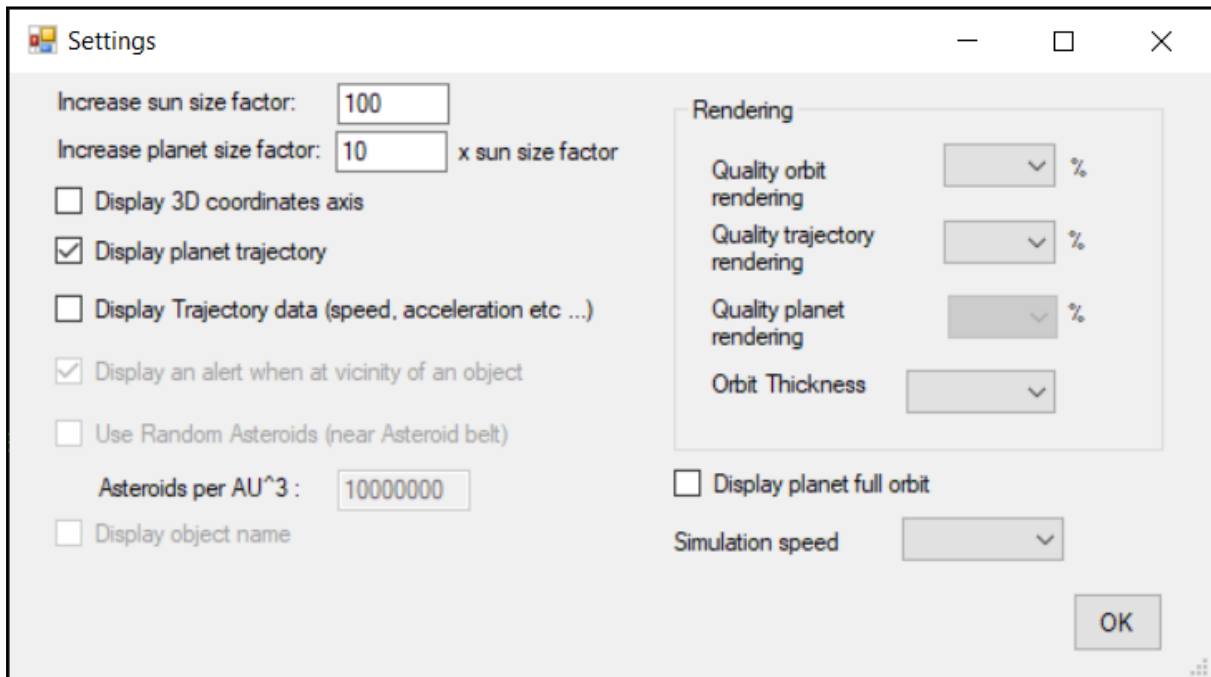


Illustration 8: Simulation Settings

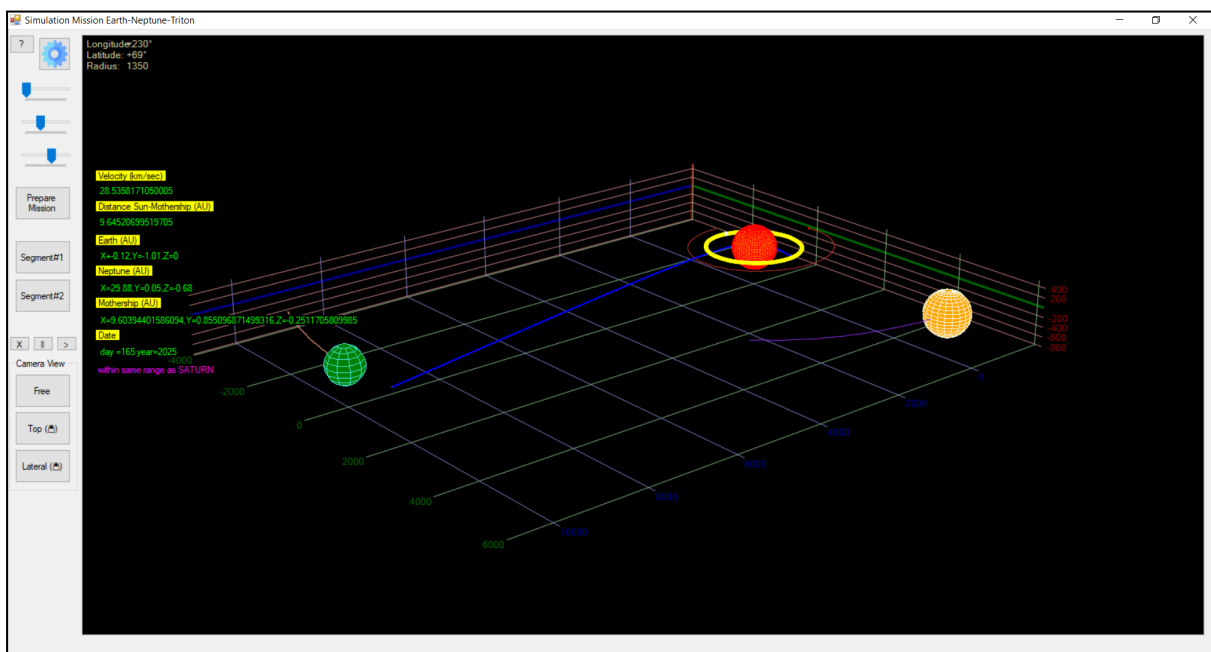


Illustration 9: Segment#1 simulation

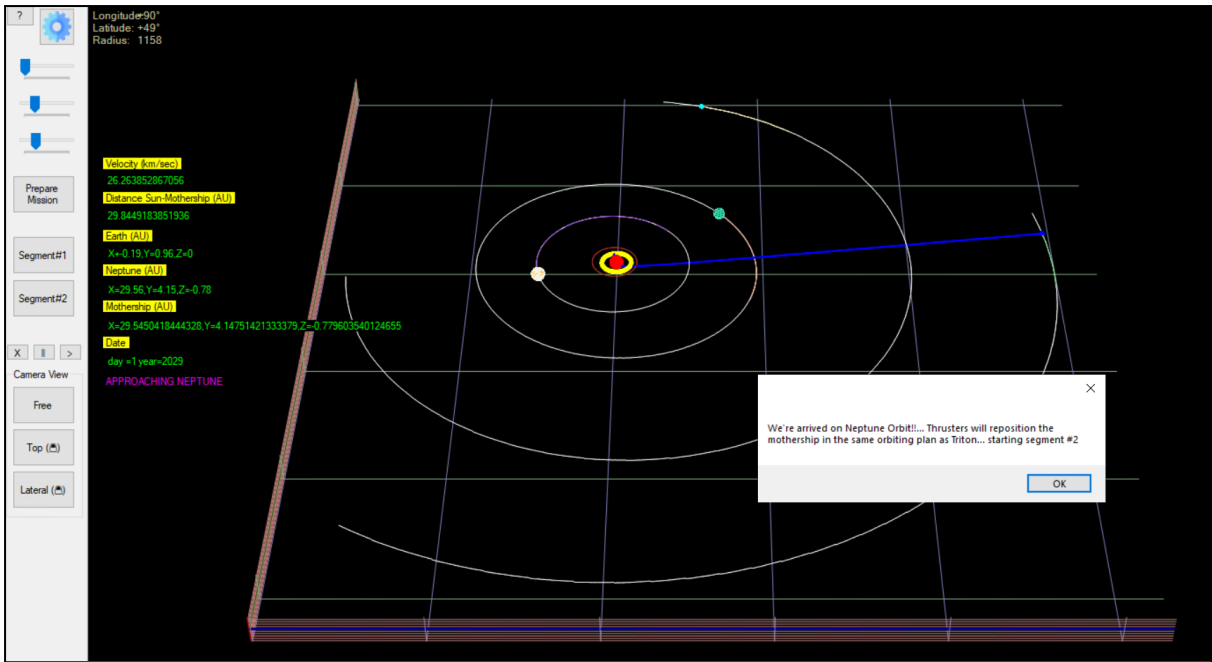


Illustration 10: Segment#1 completion (Neptune arrival)

Second stage of the mission

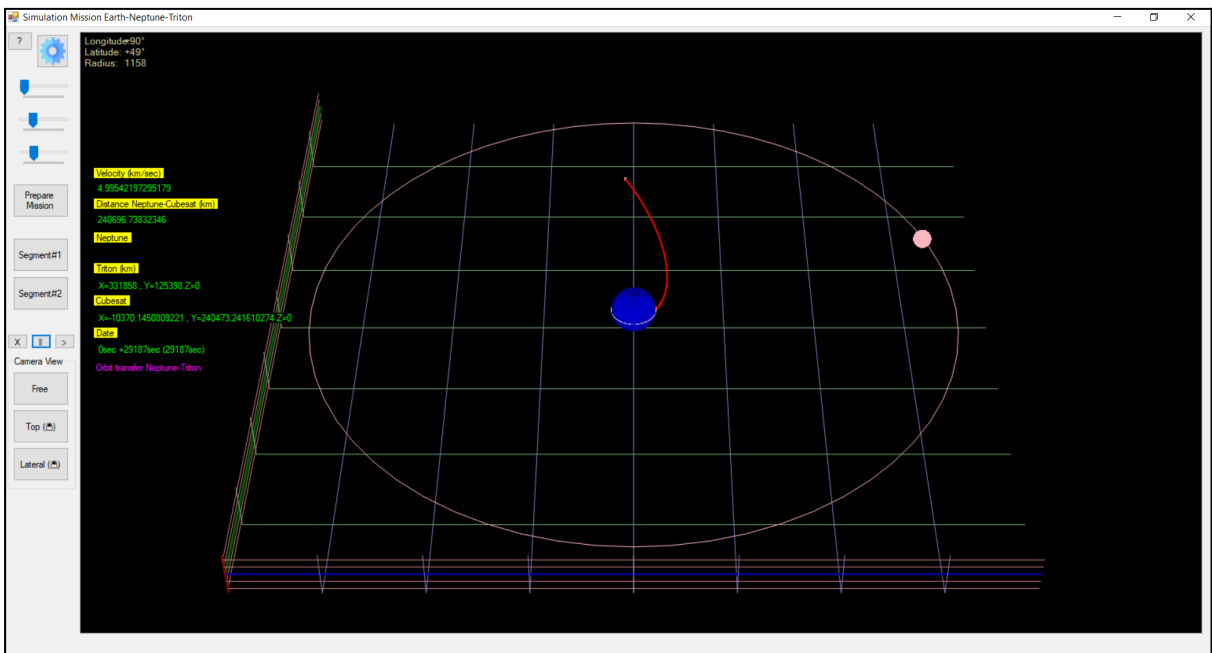


Illustration 11: Segment#2 simulation

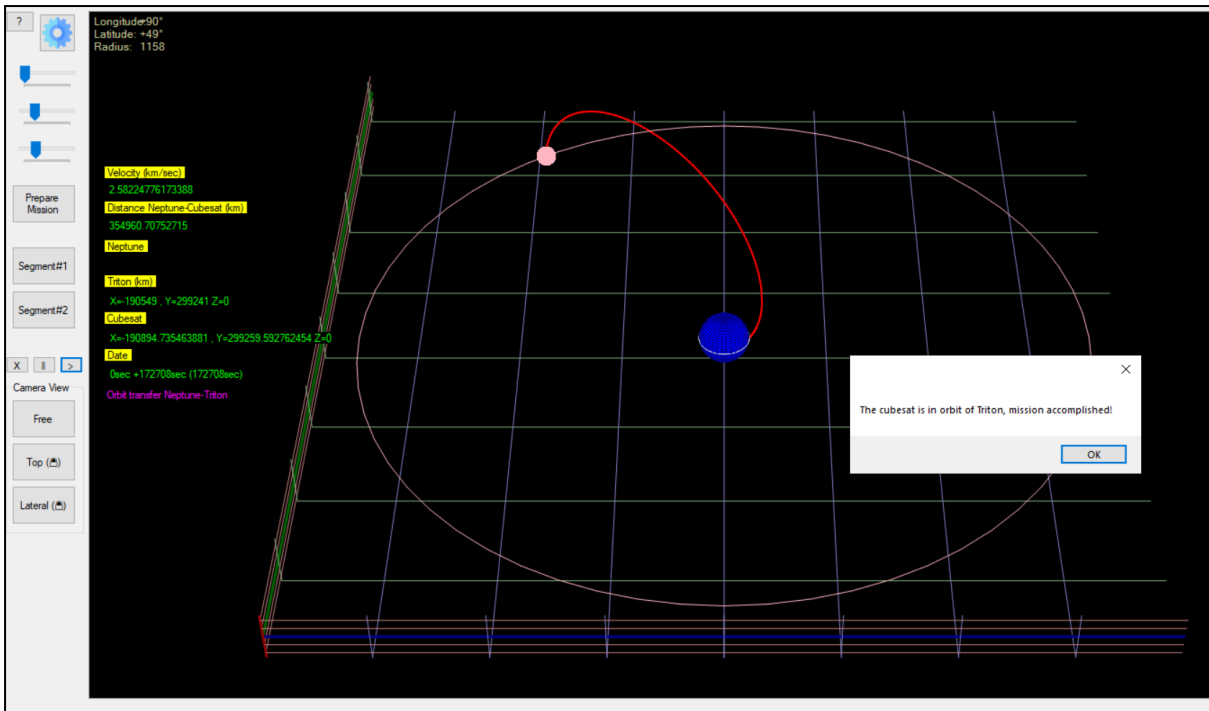


Illustration 12: Segment#2 Completion

Conclusion

The simulator developed is very basic and does not support manoeuvres in space (using thrusters) or the initial segment: multi-stage rocket launch (from Earth). However, despite its simplicity, it provides accurate estimation and computations.

REFERENCES

- [1] Orbital Mechanics for Engineering Students, Howard D. Curtis
- [2] Heliospheric Coordinate Systems M.Franz, D. Harper