T.R.
GEBZE TECHNICAL UNIVERSITY
FACULTY OF ENGINEERING
MECHANICAL ENGINEERING DEPARTMENT

ME108 ADVANCED PROGRAMMING

TERM PROJECT

SPACE SYSTEM SIMULATION

HAMDİ EKIZ (200105005031)

Table of Contents

1. Introduction

1.1. Project Purpose

In this project a software is written to simulate the space-systems. It is essential to apply the concepts that are covered throughout the advanced programming course.

1.2. System Model

The space system described here includes a collection of satellites and ground stations. Within this project, all components of the model are assumed to operate within the same plane. The schematic of this system is presented in Figure 1. Satellites follow circular or elliptical orbits around the Earth, while the ground stations are fixed points on the Earth's surface engaged in communication with the satellites. Due to the Earth's curvature and the physical constraints of communication technology, ground stations can establish communication links with satellites only within a specific region surrounding the station, termed the Field of View (FoV). This region is visually represented in the figure by the yellow area surrounding each ground station.

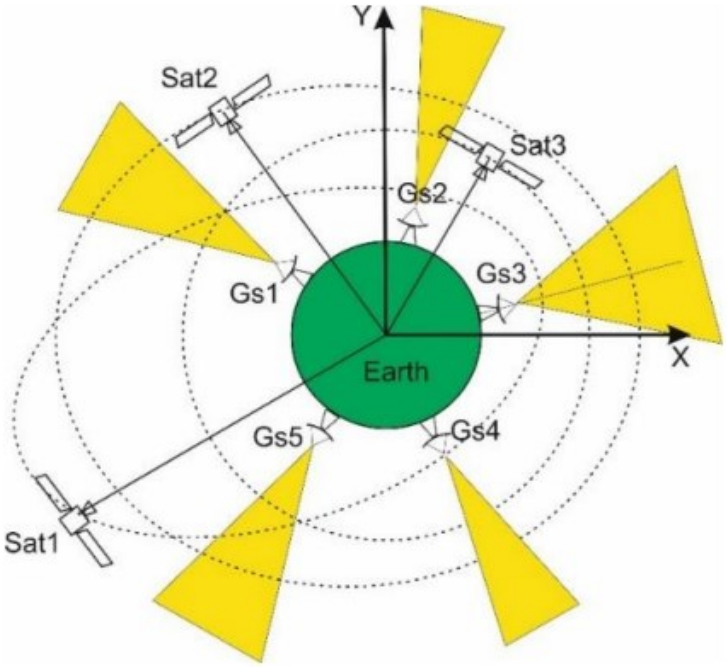


Figure 1 The schematic of the space system

Two types of satellites are assumed to exist in this system:

1) Satellites that only receive data from the ground stations
2) Satellites that only send data to the ground stations

We have also three types of ground stations:

1) Ground stations that only communicate with the satellites
2) Ground stations that only track the satellites
3) Ground stations that both communicate with and track the satellites.

Regardless of the type of the satellite and ground stations, every time that a satellite came into the field of view of a ground station, they send their type and name (each satellite and ground station have a unique name) to each other because of operational needs.

1.3 Project Requirements

1. The software shall receive the initial conditions of the satellites directly from the user.
2. The software shall receive the initial conditions of the satellites by opening a file.
3. The software shall receive the initial conditions of the ground stations directly from the user.
4. The software shall receive the initial conditions of the ground stations by opening a file.
5. The software shall simulate the movement of the satellites around the earth in a time span specified by the user.
6. Satellites shall make a report about their connections with ground stations.
7. Ground stations shall make a report about their connections with satellites.
8. After finishing the simulation, the user shall be able to see the report file of each satellite as a separate file.
9. After finishing the simulation, the user shall be able to see the report file of each ground station as a separate file.
10. The trajectory followed by each satellite shall be saved in a file.
11. The user shall be able to save the data used for simulation to be able to perform the simulations repeatedly.

12. The user shall be able to open the data used for simulation to be able to perform the simulations repeatedly.

13. The user shall be able to edit the satellite locations in the software.

14. The user shall be able to edit the ground station locations in the software.

15. The software shall provide a Windows form as the user interface.

16. The software shall display all trajectories of the satellites graphically.

2. Theories

2.1 Satellites Theories

To simulate the motion of the satellite, numerical integration methods should be used to solve the equations of motion. I use both Runge-Kutta 4th order and Euler integration methods to achieve this. The details will be discussed in the next chapters. The model for simulating satellite motion is depicted in Figure 2.

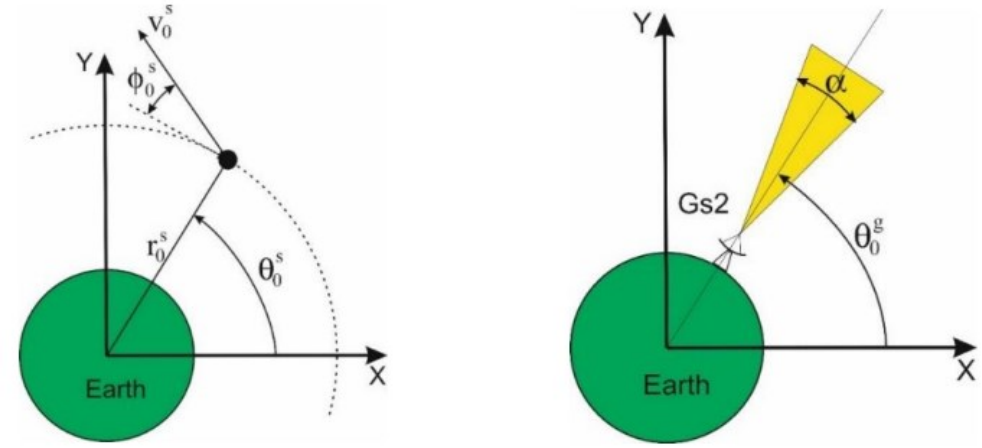


Figure 2 Initializing the satellite and ground station

According to this model, satellites are treated as lumped mass objects, and their equations of motion, in general form, are as follows:

$$\sum F_x = m_s \frac{d^2x}{dt^2} \qquad \text{and} \qquad \sum F_y = m_s \frac{d^2y}{dt^2}$$

Where m_s is the mass of satellite and $\sum F_x$ and $\sum F_y$ are summation of the forces acting on satellite in x and y directions respectively. Let's take a closer look at Figure 2 to get the location and velocity components of the satellite.

2.1.1 Position Components Calculation

We know that software receives the r_0^s, θ_0^s values from the user for each satellite. Thus, the initial components of the position of each satellite can be calculated using the following equations:

$$x_0 = r_0^s cos\theta_0^s \qquad y_0 = r_0^s sin\theta_0^s$$

In Figure 3 you can see the visualization of the satellite's position.

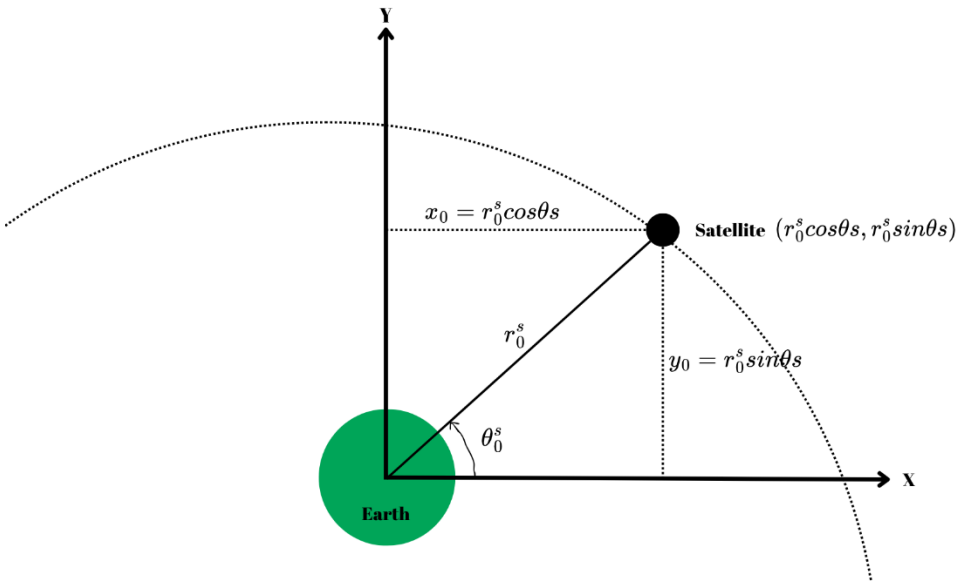


Figure 3 Visualization of the satellite's position

s

2.1.2 Velocity Components Calculation

We also know that software receives the, v_0^s, Φ_0^s values from the user for each satellite. Thus, the initial components of each satellite's velocity can be calculated using the following equations:

$$V_{x_0} = V_0^s \sin(\theta_0^s - \emptyset_0^s) \qquad V_{y_0} = V_0^s \cos(\theta_0^s - \emptyset_0^s)$$

In Figure 4 you can see the visualization of the satellite's velocity components.

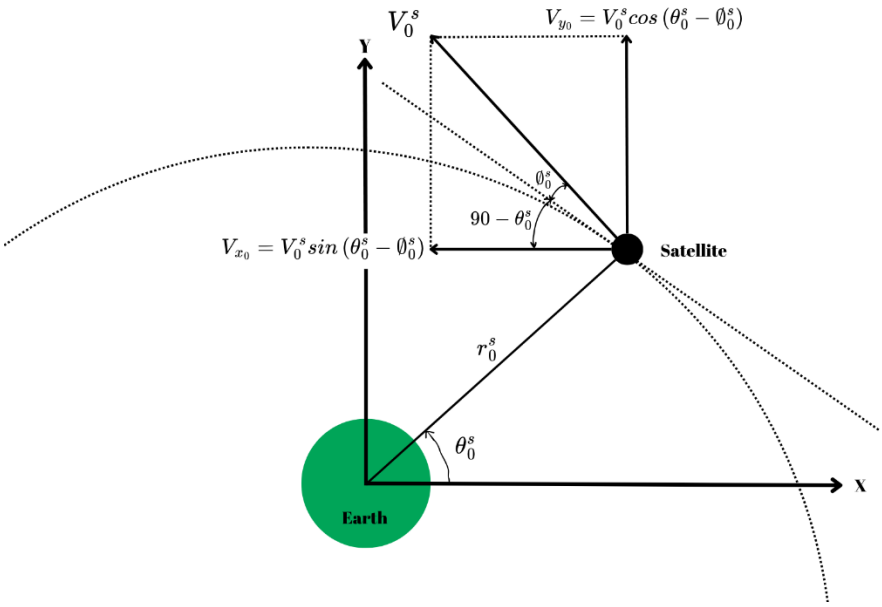


Figure 4 Visualization of the satellite's velocity components

2.1.3 Force Calculation

In this project we assume there are only two forces acting on the satellite:

1. Gravity Force
2. Aerodynamic Force

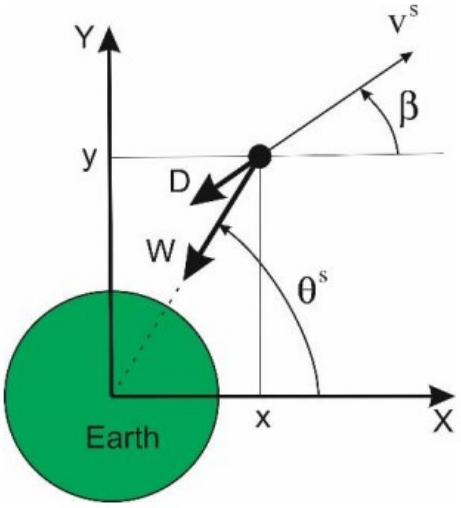


Figure 5 The model used to simulate the dynamics of the satellite

2.1.3.1 Gravity Model

In this project the gravity model is simplified. We use fallowing equation to model the gravitational force, $(F_s = m_s g)$ as a function of time:

$$g = g_0(\frac{R_E}{R_E+h})^2 \quad \text{where;} \qquad h = \sqrt{x^2+y^2} - R_E$$

In these equations g is the gravitational constant, $g_0 = 9.81\frac{m}{s^2}$ is the gravity constant at the earth surface, $R_E = 6371\ km$ is the earth radius, h is the height of the satellite from the earth surface.

Before starting any calculations let's convert R_E into meters. $R_E = 6371\ km\left(\frac{1000m}{1\ km}\right) = 6371000\ m.$

The direction of gravitational force is towards the earth center, so the components of the gravitational force calculated by below formulas:

$$W_x = -m_s g cos\theta^s \ , \ W_y = -m_s g sin\theta^s \quad \text{where;} \quad \theta^s = \text{atan}\,(\frac{y}{x})$$

In Figure 6 you can see the visualization of the gravitational force components.

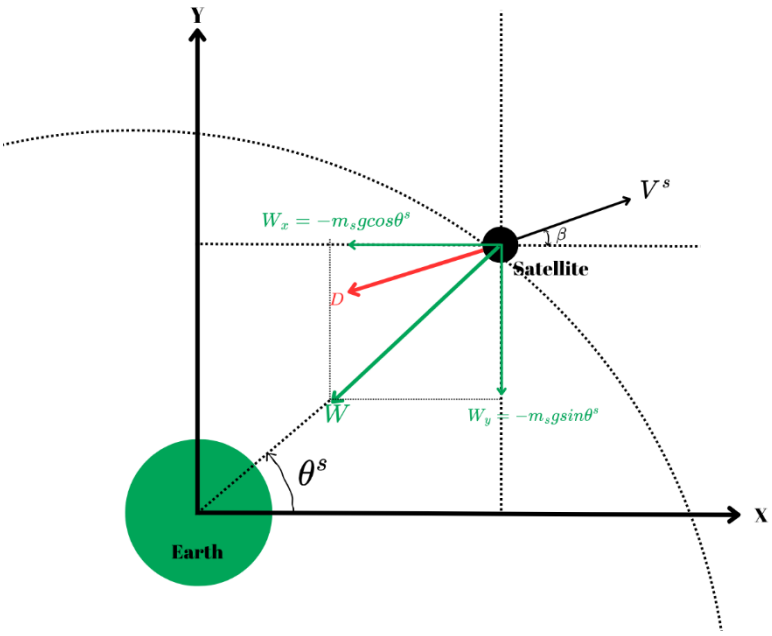


Figure 6 Visualization of the gravitational force components

2.1.3.2 Aerodynamic Model

In this project the aerodynamic model is simplified. We use fallowing equation to model the aerodynamic force:

$$D = \frac{1}{2}\rho(h)V^2SC_D$$

In this equation V is the total velocity and $\rho(h)$ is the air density which depends on the height and SC_D depends on the shape and dimensions of the satellite. For simplicity, we assume that all satellites have $SC_D = 0.3$ and assume that $\rho(h) = 10^{-10}$. So,

$$V = \sqrt{V_x^2 + V_y^2}$$

The direction of the aerodynamic force is tangent to the velocity of the satellite and is in the opposite direction of the velocity vector. So,

$$D_x = -D\cos\beta \quad , \quad D_x = -D\cos\beta \qquad \text{where; } \beta = \text{atan}\left(\frac{V_x}{V_y}\right)$$

In Figure 7 you can see the visualization of the aerodynamic force components.

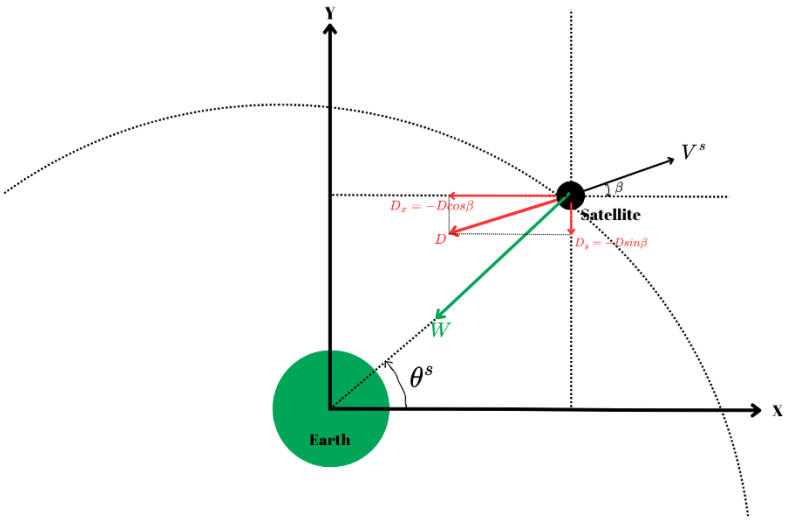


Figure 7 Visualization of the aerodynamic force components

2.1.3.3 Total Force Calculation

- We use fallowing equations to calculate total force x-component:

$$\sum F_x = W_x + D_x$$

$$\sum F_x = (-m_s g cos\theta^s\ ) + (-Dcos\beta)$$

$$\sum F_x = \left(-m_s g_0 \left(\frac{R_E}{R_E + h}\right)^2 cos\theta^s\ \right) + \left(-\left(\frac{1}{2}\rho(h)V^2 SC_D\right)cos\beta\right)$$

$$\sum F_x = \left(-m_s g_0 \left(\frac{R_E}{R_E + \sqrt{x^2 + y^2} - R_E}\right)^2 cos\theta^s\ \right) + \left(-\left(\frac{1}{2}\rho(h)\left(V_x^2 + V_y^2\right)SC_D\right)cos\beta\right)$$

$$\sum F_x = \left(-m_s g_0 \left(\frac{R_E}{\sqrt{x^2 + y^2}}\right)^2 cos\theta^s\ \right) + \left(-\left(\frac{1}{2}\rho(h)\left(V_x^2 + V_y^2\right)SC_D\right)cos\beta\right)$$

- We use fallowing equation to calculate total force y-component:

$$\sum F_y = W_y + D_y$$

$$\sum F_y = (-m_s g sin\theta^s\ ) + (-Dsin\beta)$$

$$\sum F_y = \left(-m_s g_0 \left(\frac{R_E}{R_E + h}\right)^2 sin\theta^s\ \right) + \left(-\left(\frac{1}{2}\rho(h)V^2 SC_D\right)sin\beta\right)$$

$$\sum F_y = \left(-m_s g_0 \left(\frac{R_E}{R_E + \sqrt{x^2 + y^2} - R_E}\right)^2 sin\theta^s\ \right) + \left(-\left(\frac{1}{2}\rho(h)\left(V_x^2 + V_y^2\right)SC_D\right)sin\beta\right)$$

$$\sum F_y = \left(-m_s g_0 \left(\frac{R_E}{\sqrt{x^2 + y^2}}\right)^2 sin\theta^s\ \right) + \left(-\left(\frac{1}{2}\rho(h)\left(V_x^2 + V_y^2\right)SC_D\right)sin\beta\right)$$

2.1.4 Equation of Motion Calculation

The general form of equation of motion is $\sum F = ma$. In our project we know the mass of the satellite and we can also calculate the forces acting on the satellite. So, we can find the acceleration of the satellite by using this equation $a = \frac{\sum F}{m}$. However, this

equation alone doesn't give us a complete solution because it only gives us instantaneous acceleration and we need to find out how the satellite's motion changes over time.

To achieve this, we can use numerical integration techniques, such as Runge-Kutta 4th Order and Euler methods. . In this project I use both Runge-Kutta 4th Order and Euler Integration methods. Before we start how to do numerical integration, we should first model the satellite system.

2.1.4.1 Modelling the Satellite System

We can calculate the initial position, velocity and total force components of the satellite, as mentioned in the previous sections. Before we start modeling, let's write the definitions.

$x = Satellite\ x - position$

$y = Satellite\ x - position$

$V_x = \dot{x} = Satellite\ x - velocity$

$V_y = \dot{y} = \ Satellite\ y - velocity$

If we define such $\vec{K} = [k_1, k_2, k_3, k_4]$ that $k_1 = x, k_2 = y\ , k_3 = \dot{x}\ ,\ k_4 = \dot{y}$

$\dot{k_1} = \ k_3$

$\dot{k_2} = \ k_4$

$$\dot{k_3} = \frac{1}{m_s}\sum F_x = \frac{1}{m_s}(W_x + D_x) = \frac{1}{m_s}[\left(-m_s g cos(\text{atan}\ \frac{y}{x})\right) + \left(\frac{1}{2}\rho(h)(\dot{x}^2 + \dot{y}^2)SC_D \cos\ (atan\frac{\dot{y}}{\dot{x}})\right)]$$

$$\dot{k_3} = \frac{1}{m_s}[\left(-m_s g cos(\text{atan}\ \frac{k_2}{k_1})\right) + \left(\frac{1}{2}\rho(h)(k_3^2 + k_4^2)SC_D \cos\left(atan\frac{k_4}{k_3}\right)\right)]$$

$$\dot{k_4} = \frac{1}{m_s}\sum F_y = \frac{1}{m_s}\left(W_y + D_y\right) = \frac{1}{m_s}[\left(-m_s g sin\left(\text{atan}\frac{y}{x}\right)\right) + \left(\frac{1}{2}\rho(h)(\dot{x}^2 + \dot{y}^2)SC_D \sin\left(atan\frac{\dot{y}}{\dot{x}}\right)\right)]$$

$$\dot{k_4} = \frac{1}{m_s}[\left(-m_s g sin\left(\text{atan}\frac{k_2}{k_1}\right)\right) + \left(\frac{1}{2}\rho(h)(k_3^2 + k_4^2)SC_D \sin\left(atan\frac{k_4}{k_3}\right)\right)]$$

2.1.4.2 Euler Integration Method

In mathematics and computational science, the Euler method (also called the forward Euler method) is a first-order numerical procedure for solving ordinary differential equations (ODEs) with a given initial value. It is the most basic explicit method for numerical integration of ordinary differential equations. It can be solving from below formula.

$$\vec{\boldsymbol{X}}(\boldsymbol{t}+\boldsymbol{\Delta t}) = \vec{\boldsymbol{X}}(\boldsymbol{t}) + \dot{\vec{\boldsymbol{X}}}(\vec{\boldsymbol{X}}, \boldsymbol{t})\boldsymbol{\Delta t}$$

In our satellite system we use the formulas below to use Euler Integration.

$k_{1_n} = k_{1_{n-1}} + \dot{k}_{1_{n-1}} * \Delta t$

$k_{2_n} = k_{2_{n-1}} + \dot{k}_{2_{n-1}} * \Delta t$

$k_{3_n} = k_{3_{n-1}} + \dot{k}_{3_{n-1}} * \Delta t$

$k_{4_n} = k_{4_{n-1}} + \dot{k}_{4_{n-1}} * \Delta t$

- "n" starts at 1 and goes through the number of iterations.

2.1.4.3 Runge-Kutta 4th Order Integration Method

The Runge-Kutta 4th Order Integration method is a widely used numerical method for approximating solutions to ordinary differential equations (ODEs). It improves on the Euler method by considering not only the initial slope but also the slopes at the midpoint and the end of the interval to estimate the solution. The method has four stages (hence the name "4th order"), each of which computes a weighted average of these slopes. It can be solving from below formula.

$$\vec{\boldsymbol{X}}(\boldsymbol{t}+\boldsymbol{\Delta t}) = \vec{\boldsymbol{X}}(\boldsymbol{t}) + \frac{\boldsymbol{\Delta t}}{\boldsymbol{6}}(\boldsymbol{K_1} + \boldsymbol{2K_2} + \boldsymbol{2K_3} + \boldsymbol{K_4})$$

$K_1 = \dot{\vec{X}}(\vec{X}, t)$

$K_2 = \dot{\vec{X}}(\vec{X} + \frac{1}{2}K_1\Delta t, t + \frac{\Delta t}{2})$

$K_3 = \dot{\vec{X}}(\vec{X} + \frac{1}{2}K_2\Delta t, t + \frac{\Delta t}{2})$

$K_4 = \dot{\vec{X}}(\vec{X} + K_3\Delta t, t + \frac{\Delta t}{2})$

Where;

K_1 is the slope at the beginning of the interval, using y (Euler's method),

K_2 is the slope at the midpoint of the interval, using y and K_1,

K_3 is again the slope at the midpoint, but now using y and K_2,

K_4 is the slope at the end of the interval, using y and K_3.

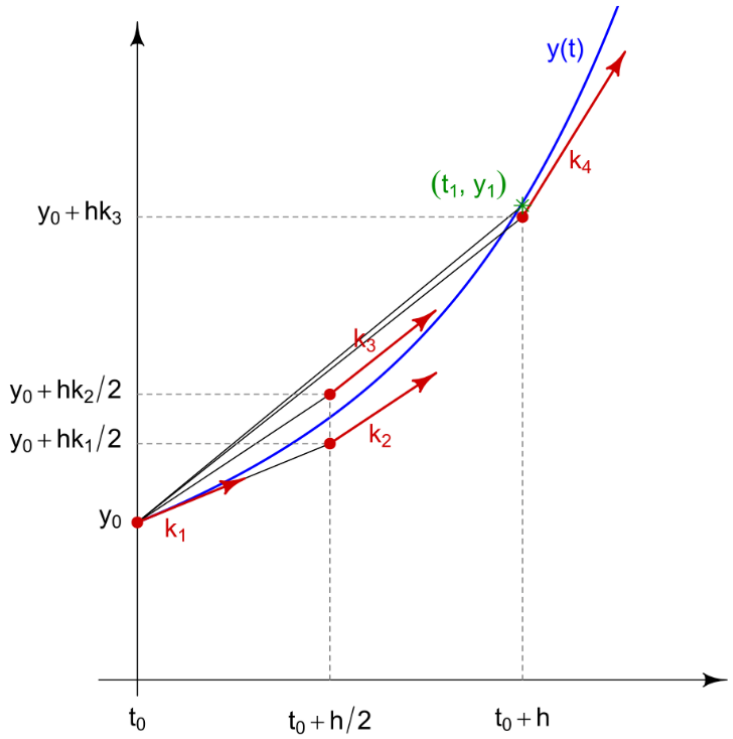


Figure 8 Slopes used by the classical Runge-Kutta method

In our satellite system we use the formulas below to use Runge-Kutta 4th Order Integration.

$\vec{K} = [k_1, k_2, k_3, k_4]$; $k_1 = x, k_2 = y$, $k_3 = \dot{x}$, $k_4 = \dot{y}$

$K_1[0] = \dot{x}_1$, $K_1[1] = \dot{y}_1, K_1[2] = \ddot{x}_1, K_1[3] = \ddot{y}_1$;

$K_2[0] = \dot{x}_2$, $K_2[1] = \dot{y}_2, K_2[2] = \ddot{x}_2, K_2[3] = \ddot{y}_2$;

$K_3[0] = \dot{x}_3$, $K_3[1] = \dot{y}_3, K_3[2] = \ddot{x}_3, K_3[3] = \ddot{y}_3$;

$K_4[0] = \dot{x}_4$, $K_4[1] = \dot{y}_4, K_4[2] = \ddot{x}_4, K_4[3] = \ddot{y}_4$;

As early mentioned, all K_1, K_2, K_3, K_4 arrays are different from each other because of they are calculating differently.

$$k_{1_n} = k_{1_{n-1}} + (K_1[0] + 2K_2[0] + 2K_3[0] + K_4[0])\frac{\Delta t}{6}$$

$$k_{2_n} = k_{2_{n-1}} + (K_1[1] + 2K_2[1] + 2K_3[1] + K_4[1])\frac{\Delta t}{6}$$

$$k_{3_n} = k_{3_{n-1}} + (K_1[2] + 2K_2[2] + 2K_3[2] + K_4[2])\frac{\Delta t}{6}$$

$$k_{4_n} = k_{4_{n-1}} + (K_1[3] + 2K_2[3] + 2K_3[3] + K_4[3])\frac{\Delta t}{6}$$

- "n" starts at 1 and goes through the number of iterations. Also, when "n" changes K_1, K_2, K_3, K_4 arrays recalculated.

2.1.5 Types of Satellite

Two types of satellites are assumed to exist in this system:

1) Satellites that only receive data from the ground stations
2) Satellites that only send data to the ground stations

2.1.5.1 Receiver Satellite

Receiver satellites are only designed to receive data from ground stations, and they are not capable of sending data back to them. In this project, when satellites generate a report, the receiver satellites are not able to transmit this data and they can only receive information from the corresponding ground stations.

2.1.5.2 Sender Satellite

While the sender satellites are only designed to send data from ground stations, and they are not capable of receiving data back to them. In this project, when satellites generate a report, the sending satellites are not able to receive data from a ground station. They only send information to the corresponding ground station.

2.2 Ground Station Theories

We do not need to derive an equation of motion at the ground station. In this section, we will only determine how to implement the field of view algorithm and the location of the ground station.

2.2.1 Position Components Calculation

We know that software receives the $\propto$ (Field of View Angle), θ_0^g (Theta Angle) values from the user for each ground station. Thus, the initial components of the position of each ground station can be calculated using the following equations:

$$x_0 = R_E cos\theta_0^g \qquad\qquad y_0 = R_E sin\theta_0^g$$

Where R_E is the radius of the earth and in this project, we assume that the world is circular, and has an equation of $x^2 + y^2 = R_E^2$. The value of $R_E = 6371\ km = 6371000\ meters$.

In Figure 9 you can see the visualization of the ground station position.

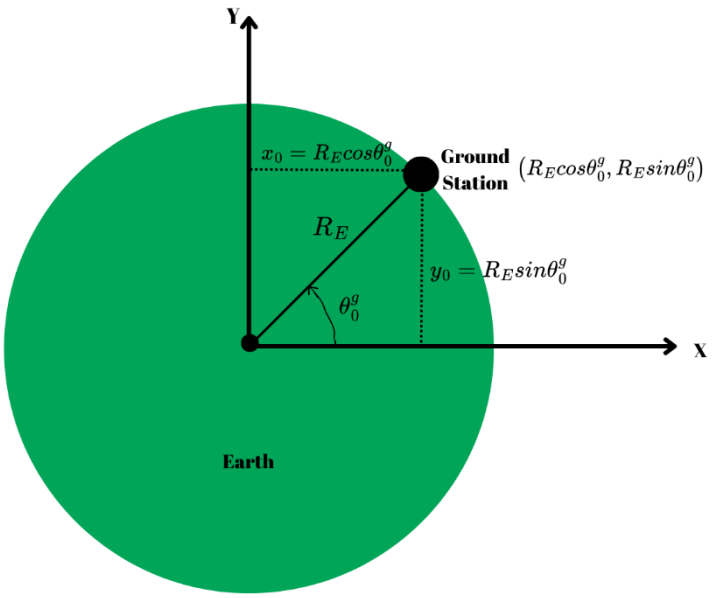


Figure 9 Visualization of the ground station's position

2.2.2 Field of View Angle Algorithm

In order to create the field of view angle algorithm, we should first visualize the field of view angle and satellite position. In Figure 10, you can see that the field of view angle and satellite position are the same figure.

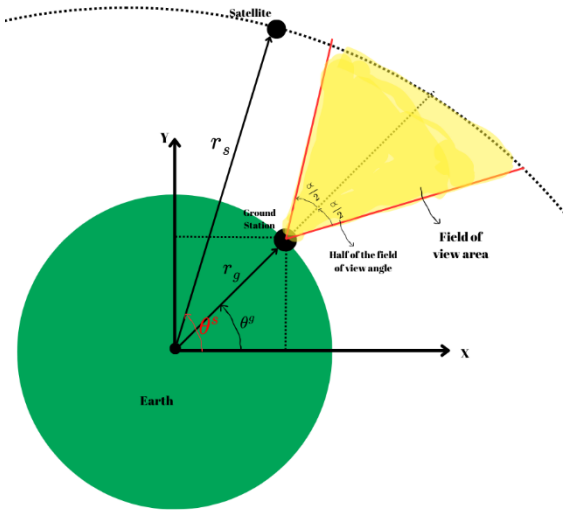


Figure 10 Visualization of the ground station's field of view angle

Figure 10 shows the position vector of the satellite and the position vector of the ground station. We can find this vector because we know everything we need to know. After finding both position vectors, we can draw a vector from the ground station to the satellite. After finding the new vector, we can measure the angle of the new vector drawn with the field of view center. And by using the angle, we can check whether the satellite is in the field of view of the ground station. In Figure 11, we draw the new vector as purple.

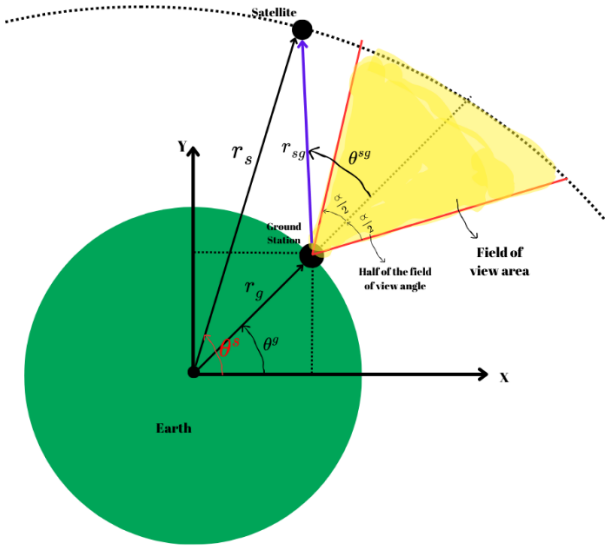


Figure 11 Visualization of the new drawn vector

To find the components of the drawn vector $\vec{r}_{sg}$ in Figure 11 we use below formulas:

$\vec{r}_{sg} = \vec{r}_s - \vec{r}_g \ \ where;$

$\vec{r}_s \ is \ the \ position \ vector \ of \ the \ satellite$

$\vec{r}_g \ is \ the \ postion \ vector \ of \ the \ ground \ station,$

$\vec{r}_s = r_s cos\theta_s i + r_s sin\theta_s j$

$\vec{r}_g = r_g cos\theta_g i + r_g sin\theta_g j$

$\vec{r}_{sg} = \left(r_s cos\theta_s - r_g cos\theta_g\right)i + (r_s sin\theta_s - r_g sin\theta_g)j$

So, we found the components of the vector $\vec{r}_{sg}$ from the ground station to the satellite. Let's measure the angle θ_{sg} between the the vector $\vec{r}_{sg}$ with the center of the field of view. We use the formulas below to measure the the angle.

$$\cos\ \theta_{sg} = \frac{\vec{r}_g.\vec{r}_{sg}}{|\vec{r}_g||\vec{r}_{sg}|} \text{ where;}$$

$\vec{r}_g.\vec{r}_{sg}$ is the dot product of the ground station position vector and vector from the ground station to the satellite.

To calculate the dot vector of these two vectors we use below formula:

$\vec{r}_g.\vec{r}_{sg} = \left(r_g cos\theta_g\right)\left(r_s cos\theta_s - r_g cos\theta_g\right) + (r_g sin\theta_g)(r_s sin\theta_s - r_g sin\theta_g)$

Magnitude of vectors are calculated in below formulas:

$\left|\vec{r}_g\right| = \sqrt{(r_g cos\theta_g)^2 + (r_g sin\theta_g)^2}$

$\left|\vec{r}_{sg}\right| = \sqrt{(r_s cos\theta_s - r_g cos\theta_g)^2 + (r_s sin\theta_s - r_g sin\theta_g)^2}$

From these calculated values we can calculate the angle θ_{sg}. To determine whether satellite is in the field of view or not we should check the angle θ_{sg} is greater or lower than half of the field of view angle $\frac{\propto}{2}$. If,

$\frac{\propto}{2} < \theta_{sg} \longrightarrow Satellite \ is \ in \ the \ field \ of \ view$

$\theta_{sg} < \frac{\propto}{2} \longrightarrow Satellite \ is \ not \ in \ the \ field \ of \ view$

2.2.3 Types of Ground Stations

Three types of ground stations are assumed to exist in this system:

1) Ground stations that only communicate with the satellites
2) Ground stations that only track the satellites
3) Ground stations that both communicate with and track the satellites.

2.2.3.1 Ground Stations with Only Communicate

These types of ground stations are only designed to communicate with the satellite, and they are not able to track the satellite. In this project, when the ground station generates a report, these types of ground stations are not able to track the satellite position and they can only communicate with the corresponding satellite type.

2.2.3.2 Ground Stations with Only Track

These types of ground stations are only designed to track the position of the satellite, and they are not able to communicate with the satellites. In this project, when the ground station generates a report, these types of ground stations are not able to communicate with the satellite, they can only track the position of the satellite, whether sender or receiver types.

2.2.3.3 Ground Stations with Both Communicate and Track

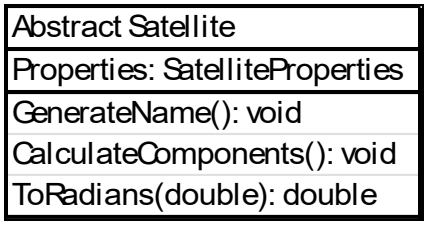
These types of ground stations are designed only to track the position of the satellite and also to communicate with the satellites. In this project, when the ground station generates a report, these types of ground stations are both communicating with the satellite and only tracking the position of the satellite, with the corresponding satellite type.

3. Code Structure

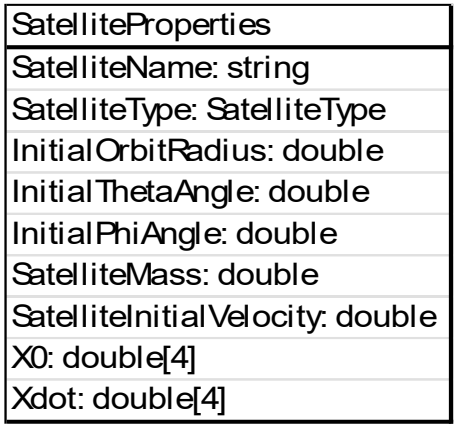
After understanding the theory behind the project. Now we can start with the code structure of the project.

3.1 Satellite Classes

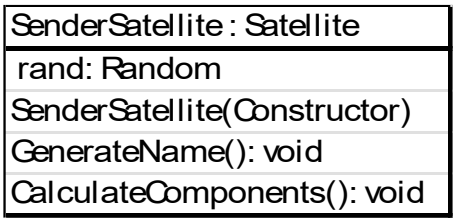
In the below diagrams you can see the Class diagram of each satellite related class.



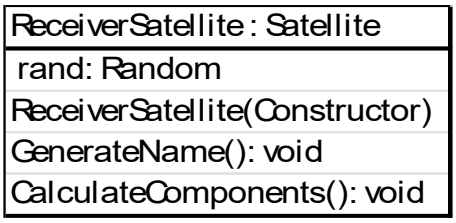
Class Diagram 1 Abstract Satellite



Class Diagram 2 SatelliteProperties



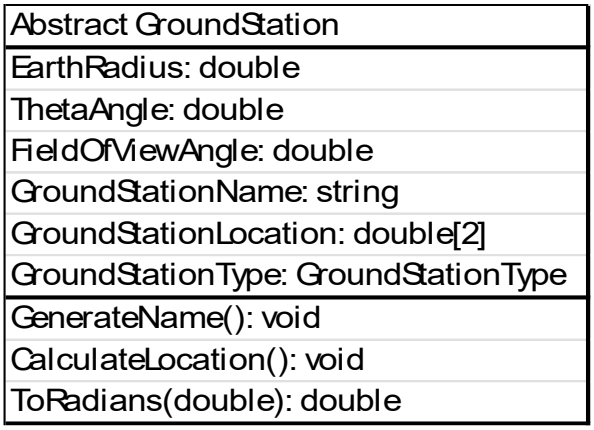
Class Diagram 3 SenderSatellite



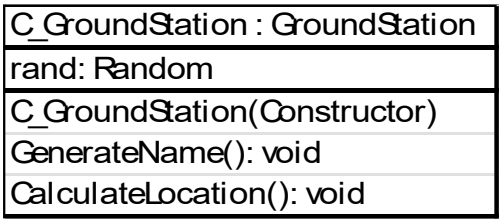
Class Diagram 4 ReceiverSatellite

3.2 Ground Station Classes

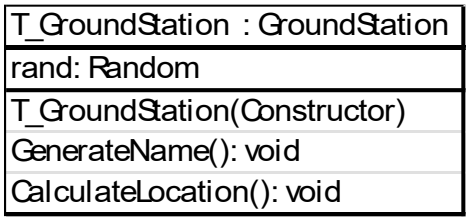
In the below diagrams you can see the Class diagram of each ground station related class



Class Diagram 5 Abstract GroundStation



Class Diagram 6 C_GroundStation



Class Diagram 7 T_GroundStation

BCT_GroundStation : GroundStation
rand: Random
BCT_GroundStation(Constructor)
GenerateName(): void
CalculateLocation(): void

Class Diagram 8 T_GroundStation

3.3 Space Model Classes

In the below diagrams you can see the Class diagram of each space model related class

Abstract SpaceModels
EarthRadius: double
ThetaAngle: double
BetaAngle: double
CalculateForceComponents(): void
CalculateForce(): double

Class Diagram 9 Abstract SpaceModels

GravityModel: SpaceModels
GravitationalForce_XComponent: double
GravitationalForce_YComponent: double
satellite: Satellite
CalculateForce(): double
CalculateForceComponents(): void
CalculateGravity(): double
CalculateHeight(): double
CalculateThetaAngle(): double

Class Diagram 10 GravityModel

AerodynamicModel : SpaceModels
AerodynamicForce_XComponent: double
AerodynamicForce_YComponent: double
satellite: Satellite
CalculateForce(): double
CalculateForceComponents(): void
CalculateVelocity(): double
CalculateBetaAngle(): double

Class Diagram 11 AerodynamicModel

TotalForceCalculator : SpaceModels
TotalForce_XComponent: double
TotalForce_YComponent: double
gravityModel: GravityModel
aerodynamicModel: AerodynamicModel
CalculateForce(): double
CalculateForceComponents(): void
CalculateTotalForce(): void

Class Diagram 12 TotalForceCalculator

3.4 Simulation Class

In the below diagrams you can see the Class diagram of the Simulation class

Simulation
FinalTime: double
TimeStep: double
InitialTime: double
form: MainForm
satellite: Satellite
groundStation: GroundStation
totalForceCalculator: TotalForceCalculator
satelliteType: SatelliteType
groundStationType: GroundStationType
SatelliteList: List<Satellite>
GroundStationList: List<GroundStation>
SatellitePositions: Dictionary<Satellite, List<(double X, double Y)>>
SatelliteVelocity: Dictionary<Satellite, List<(double Vx, double Vy)>>
Simulation(form: MainForm,
satelliteType: SatelliteType,
groundStationType: GroundStationType)
InitializeSimulation(): void
InitializeGroundStation(): void
SaveGroundStationData(saveGroundStation: GroundStation): void
SaveGroundStationReport(satellite: Satellite,
saveGroundStation: GroundStation,
message: string): void
ChooseCustomAddress(saveGroundStation: GroundStation): string
InitializeSatellite(): void
SaveSatelliteData(saveSatellite: Satellite): void
SaveSatelliteReport(saveSatellite: Satellite,
groundStation: GroundStation,
message: string): void
ChooseCustomAddress(saveSatellite: Satellite): string
Derivative(satellite: Satellite): double[]
Integrate_Euler(satellite: Satellite): void
Integrate_RungeKutta4thOrder(satellite: Satellite): void
IsInFieldOfView(satellite: Satellite,
groundStations: List<GroundStation>): void

Class Diagram 13 Simulation

3.5 DrawToPicture Class

In the below diagrams you can see the Class diagram of DrawToPicture related classes.

DrawToPictureAbstract
pictureBoxSatelliteViewer: PictureBox
pictureBoxWorld: PictureBox
pictureBoxSatellite: PictureBox
textBoxThetaAngle: TextBox
textBoxPhiAngle: TextBox
textBoxOrbitRadius: TextBox
pictureBoxGroundStationViewer: PictureBox
pictureBoxWorld_2: PictureBox
pictureBoxGroundStation: PictureBox
textBoxGroundStationTheta: TextBox
textBoxGroundStationFoVAngle: TextBox
Draw_Radius(): void
Draw_CoordinateSystem(): void
CoordinateSystemConverter(sender: object, e: EventArgs): void

Class Diagram 14 DrawToPictureAbstract

DrawGroundStation: DrawToPictureAbstract
width: int
height: int
graphics: Graphics
GroundStationThetaAngle: double
FieldOfViewAngle: double
EarthRadius: double
PixelScale: double
Draw_Radius(): void
Draw_RadiusCon(): void
Draw_FieldOfView(): void
Draw_FieldOfViewCon(): void
Draw_CoordinateSystem(): void
SetPictureBoxWorldPosition(): void
SetPictureBoxGroundStationPosition(): void
CoordinateSystemConverter(sender: object, e: EventArgs): void

Class Diagram 15 DrawGroundStation

DrawSatellite : DrawToPictureAbstract
width: int
height: int
graphics: Graphics
ThetaAngle: double
PhiAngle: double
OrbitRadius: double
PixelScale: double
Draw_Orbit(): void
Draw_Tangent(): void
Draw_Radius(): void
Draw_CoordinateSystem(): void
CoordinateSystemConverter(sender: object, e: EventArgs): void
Draw_VelocityLine(): void
SetPictureBoxWorldPosition(): void
SetPictureBoxSatellitePosition(): void
RescalePicture(newOrbitRadius: double, originalWorldSize: Size, originalSatelliteSize: Size, originalPixelScale: double): void

Class Diagram 16 DrawSatellite

3.6 GraphDrawer Class

In the below diagrams you can see the Class diagram of GraphDrawer related classes.

Abstract GraphDrawer
graphics: Graphics
abstract WriteGraphName(): void
abstract Draw_CoordinateSystem(): void
abstract Draw_Border(): void
abstract Draw_PositionZero(): void

Class Diagram 17 GraphDrawer

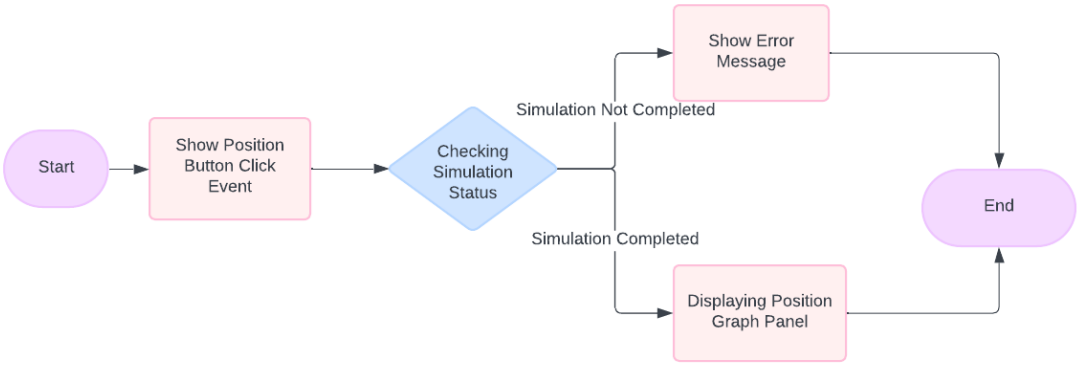
DrawTrajectoryVelocity: Abstract GraphDrawer
pictureBoxSatelliteVelocity
TrajectoryViewer: PictureBox
Draw_Trajectory(velocity: List<(double X, double Y)>): void
WriteGraphName(): override void
Draw_CoordinateSystem():override void
Draw_Border(): override void
Draw_PositionZero(): override void

Class Diagram 18 DrawTrajectoryVelocity

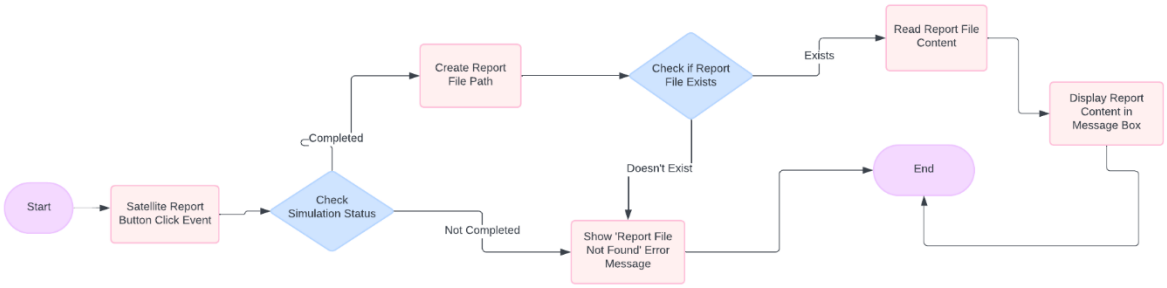
DrawTrajectoryPosition : Abstract GraphDrawer
pictureBoxSatelliteTrajectoryViewer: PictureBox
Draw_Trajectory(velocity: List<(double X, double Y)>): void
WriteGraphName(): override void
Draw_CoordinateSystem():override void
Draw_Border(): override void
Draw_PositionZero(): override void

Class Diagram 19 DrawTrajectoryPosition

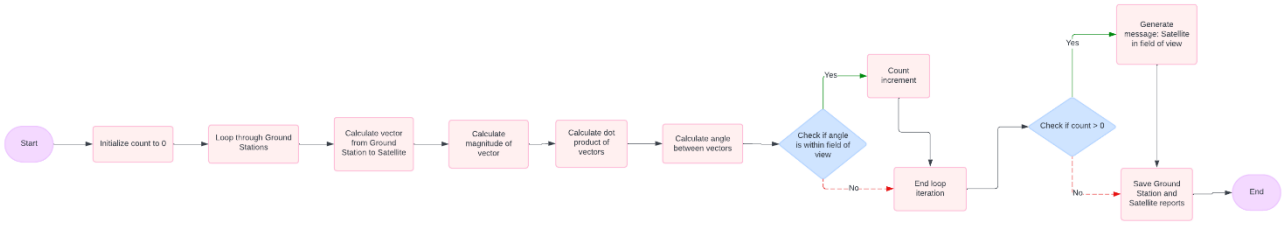
4. Flow Charts



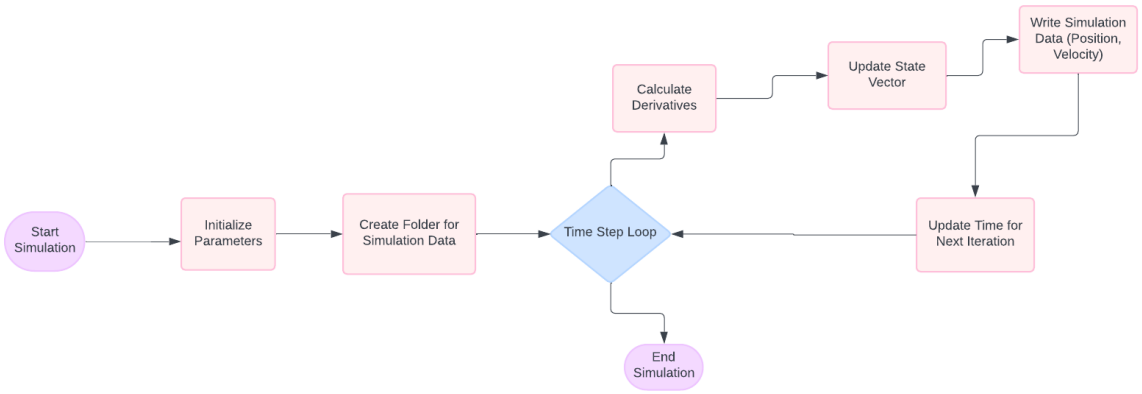
Flow Chart 1 Show Position Trajectory Button Click Event



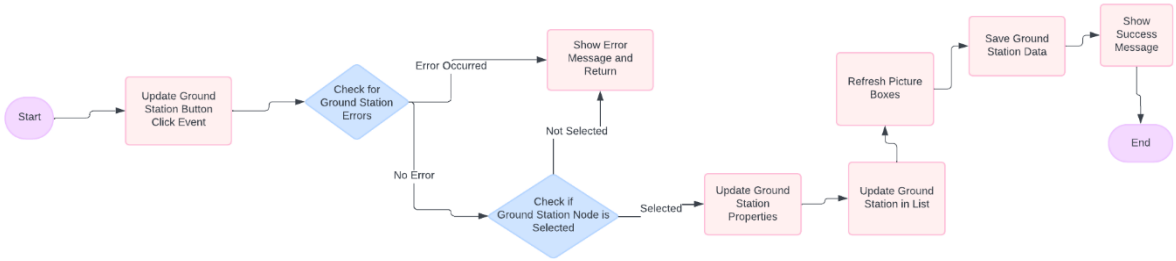
Flow Chart 2 Satellite Report Button Click Event



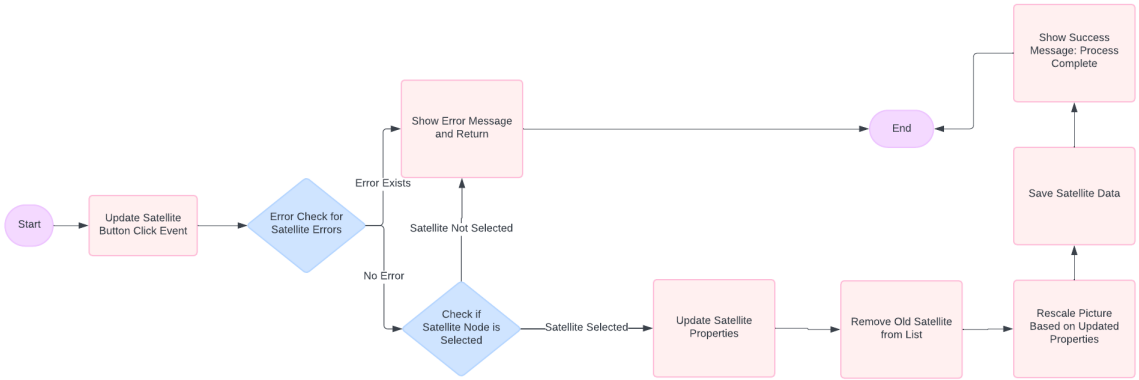
Flow Chart 3 Field of View Checker



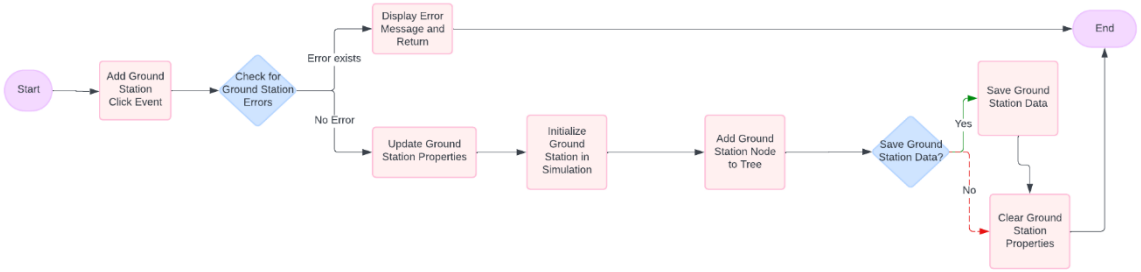
Flow Chart 4 Euler Integration



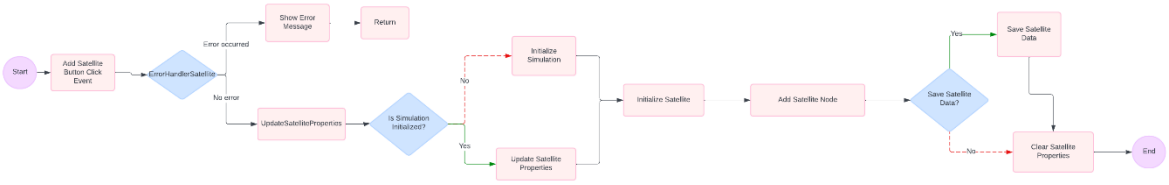
Flow Chart 5 Update Ground Station Button Click Event



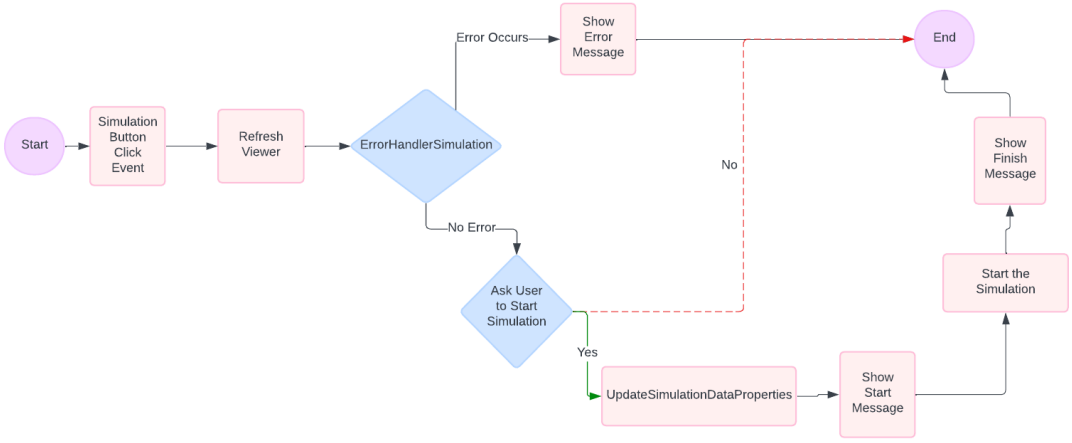
Flow Chart 6 Update Satellite Button Click Event



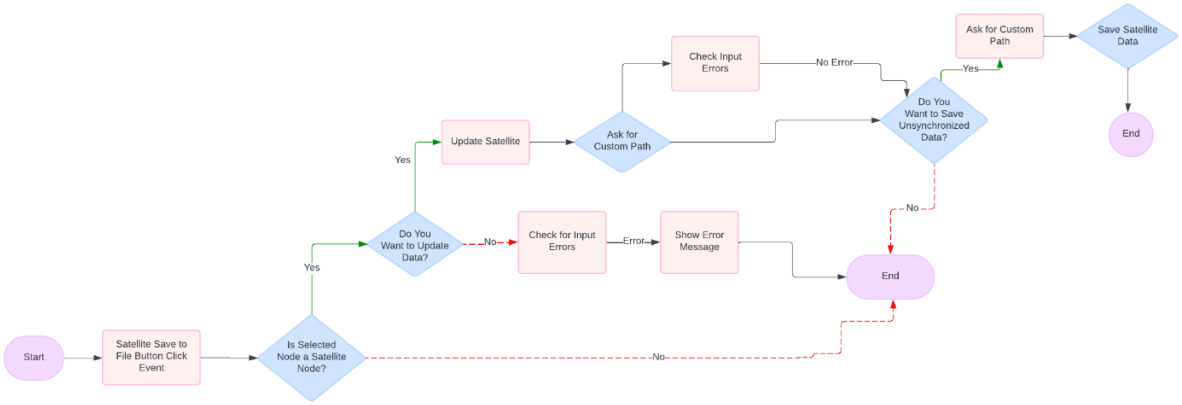
Flow Chart 7 Create Ground Station Button Click Event



Flow Chart 8 Create Satellite Button Click Event



Flow Chart 9 Start Simulation Button Click Event



Flow Chart 10 Satellite Save to File Click Event

5. User Manuel

5.1 Main Screen

When the user opens the program, first they see Figure 5.1.

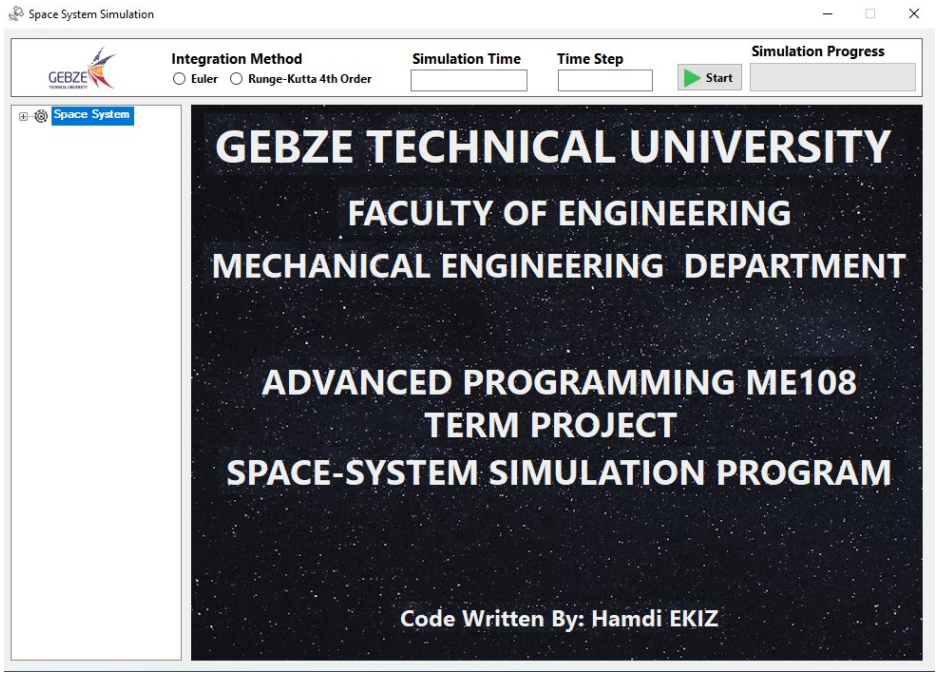


Figure 5. 1 Opening Screen of the Program

Click where the cursor located is in Figure 5.2. This action expands the tree view.

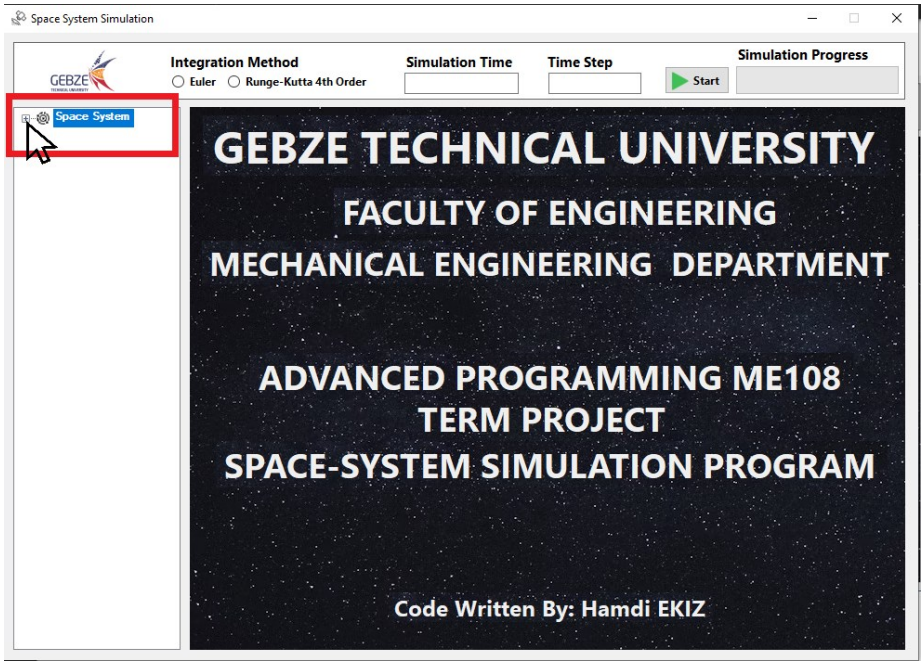


Figure 5. 2 Action to Expands the Tree View

After clicking we see Figure 5.3. In that figure there are two nodes called Satellites and Ground Stations.

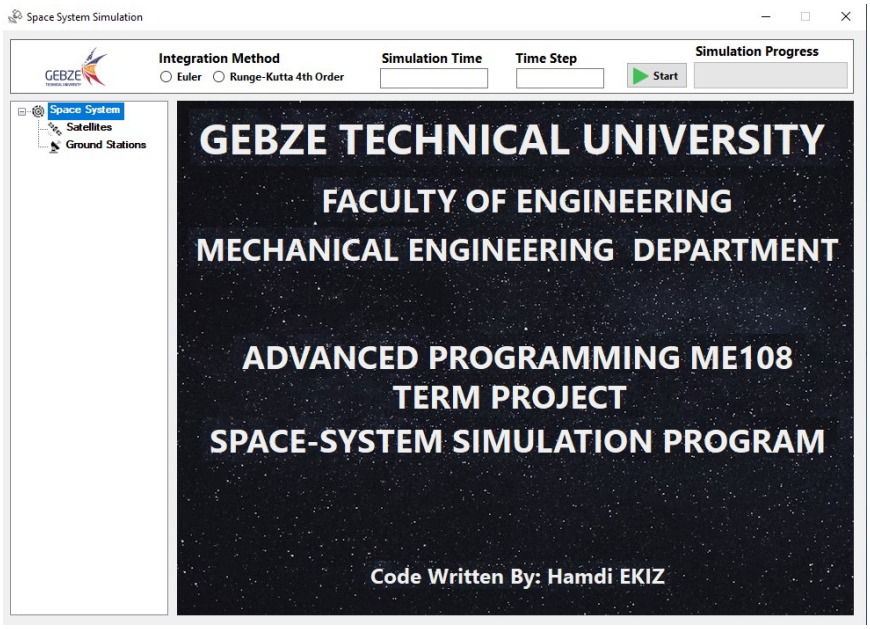


Figure 5. 3 Expanded Tree View

5.2 Satellites Screen

If the user clicks the Satellites node, the user reaches the Satellites page. In Figure 5.4 shows that satellites page.

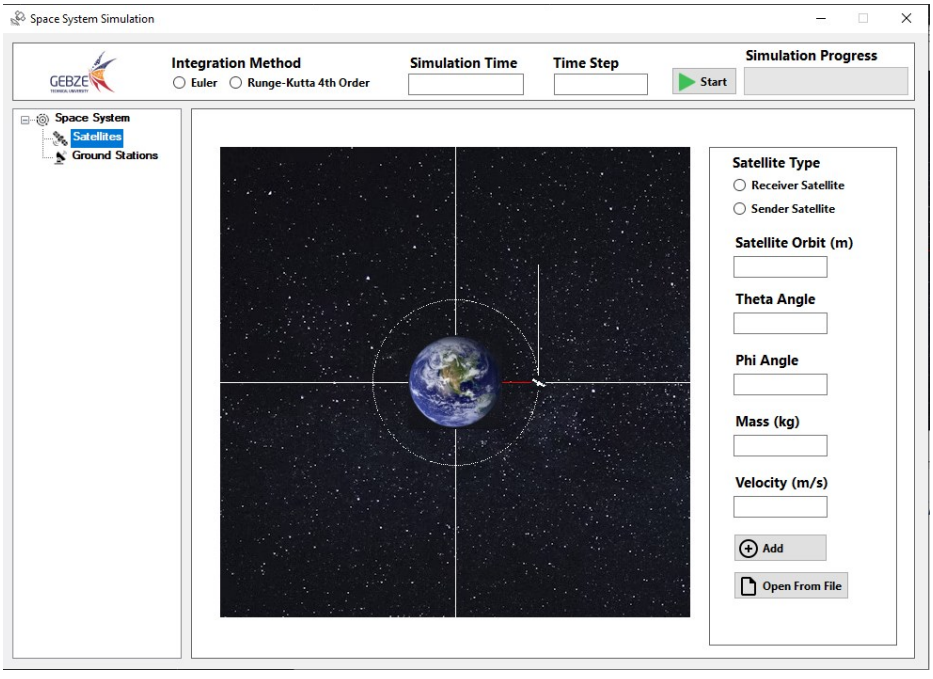


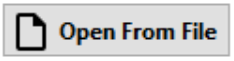
Figure 5. 4 Satellites Page

In this page we can see that on the right side there are some text boxes, radio buttons and buttons, and, in the middle, there is a satellite orbit representation.

Firstly, let's define how to works each button.

- **Add button** is a button that works to create a satellite and add it to the Satellites child node after the user has entered all the values correctly.

- **The Open from File button** is a button that allows the user to enter previously saved data without having to re-enter it.



There are some radio buttons called Receiver Satellite and Sender Satellite. This radio button works that determining the type of satellite.

- **Receiver Satellite** if the user clicks this radio button, the program will determine that the satellite type is Receiver Satellite.
- **Sender Satellite** if the user clicks this radio button, the program will determine that the satellite type is Sender Satellite.

We have five text boxes; we can understand that each text box indicates by the text written above it. In the Figure 5.5 shows that the satellite type is ***Sender Satellite***, the satellite orbit is ***7000000 meters,*** satellite theta angle is ***40 degrees,*** phi angle is ***30 degrees,*** mass of the satellite is ***1200 kg,*** **velocity of the satellite is** ***7000 m/s.***

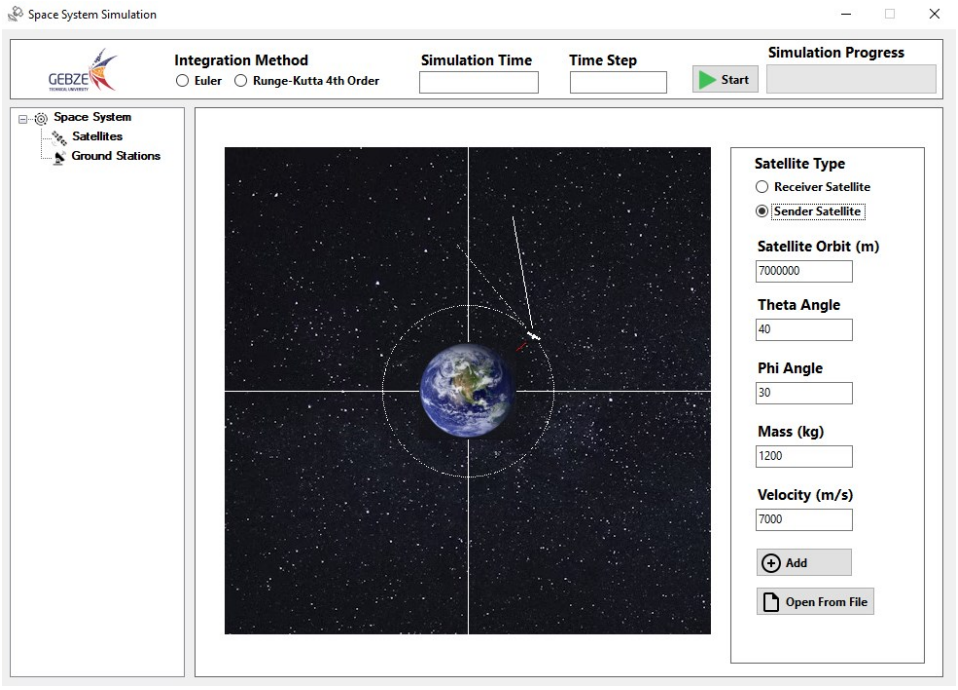


Figure 5. 5 Example of entering properties of the satellite

After user click the add button it asks the user to “Do you want to save the {Satellite Name} satellite data?” In the Figure 5.6 shows this message.

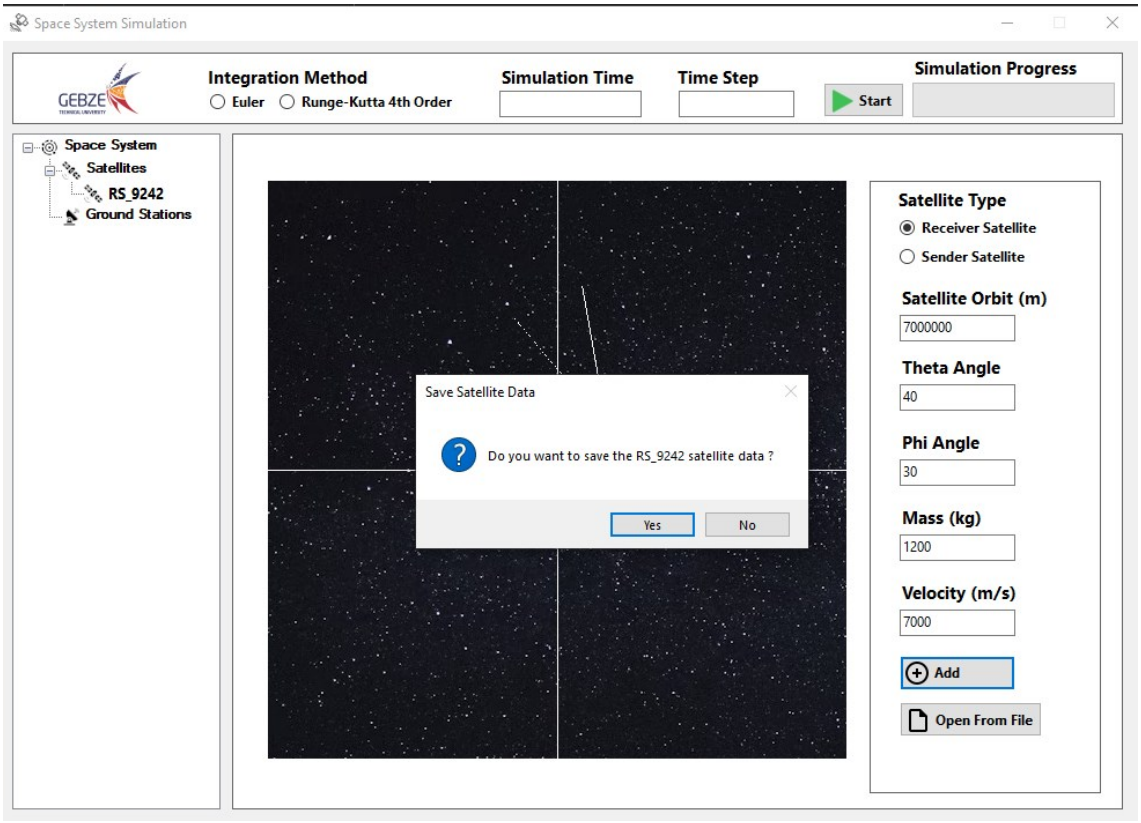


Figure 5. 6 Save Satellite Data Message

If user clicks the “**Yes**” button the program saves satellite data to the default address. The saved Satellite data shown in Figure 5.7.

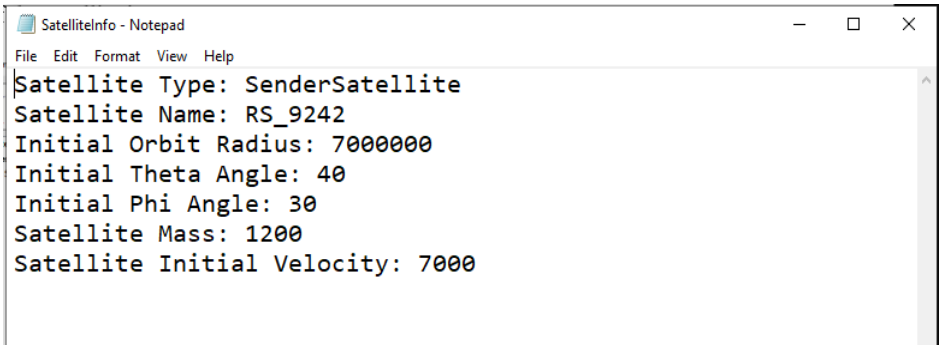


Figure 5. 7 Saved Satellite Data

If user clicks the “**No**” button the program does not save satellite data, but we can save it later.

If user clicks the **Open From File** button user shows the Figure 5.8

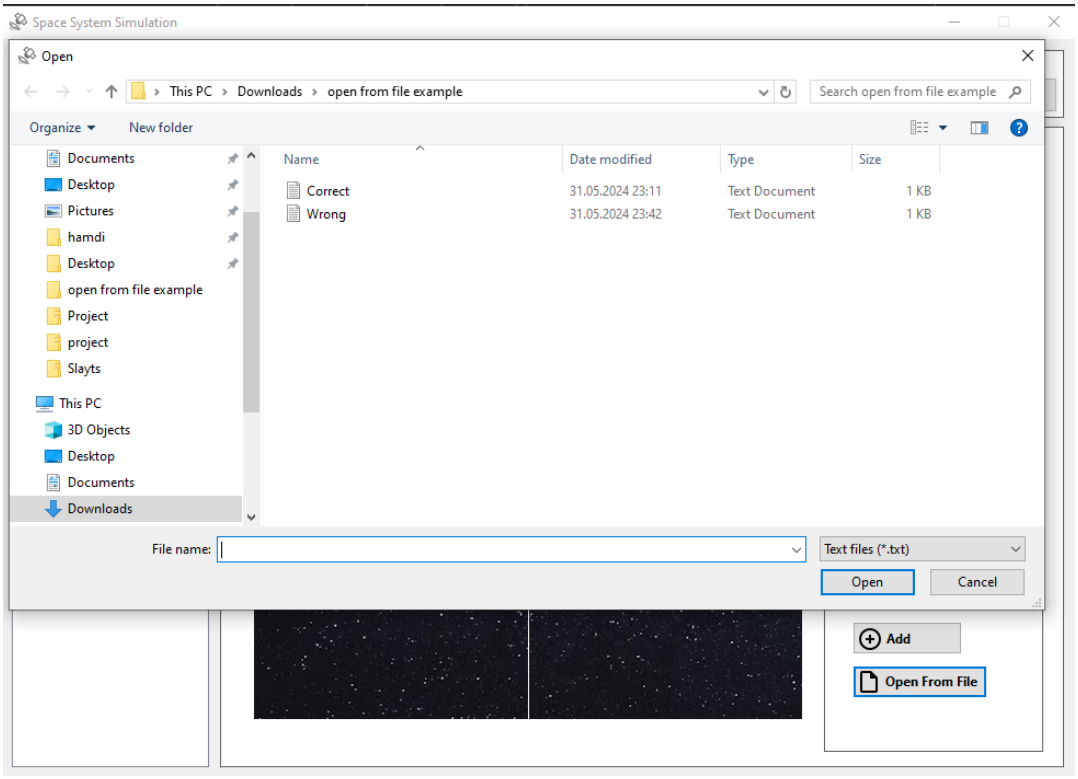


Figure 5. 8 After Click Open From File Button

If user opens the "Correct format file" the program gets the properties of the satellite in Figure 5.9 you can see that the result.

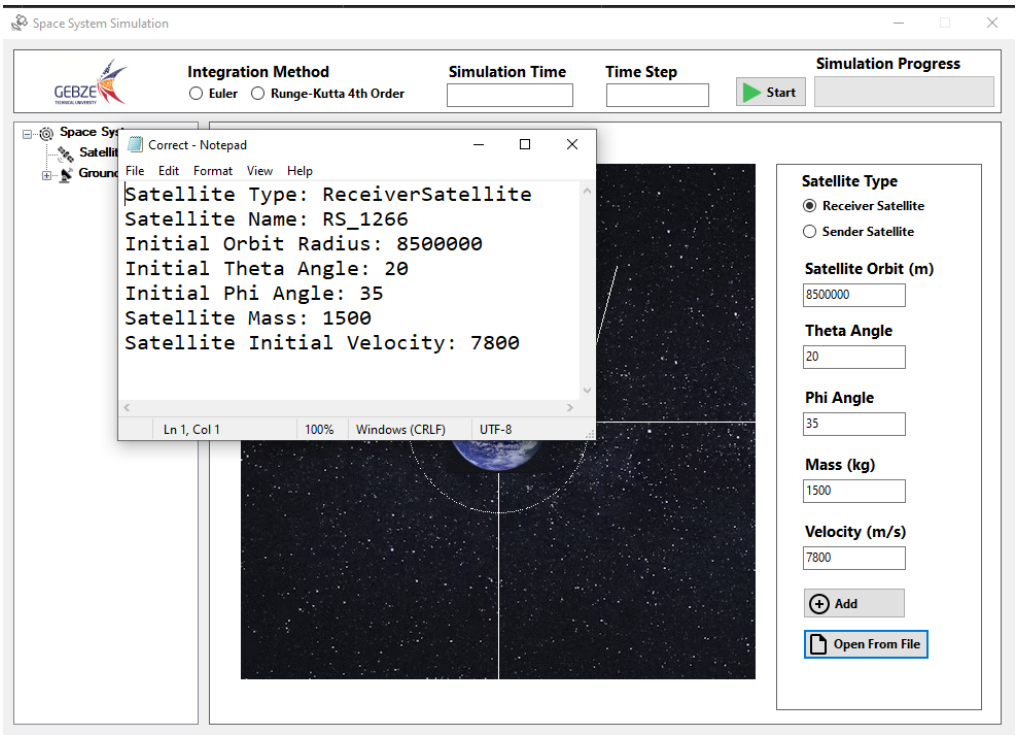


Figure 5. 9 Satellite correct format file open

If user opens the “Wrong format file” the program gives an error message in Figure 5.10 you can see that the result.

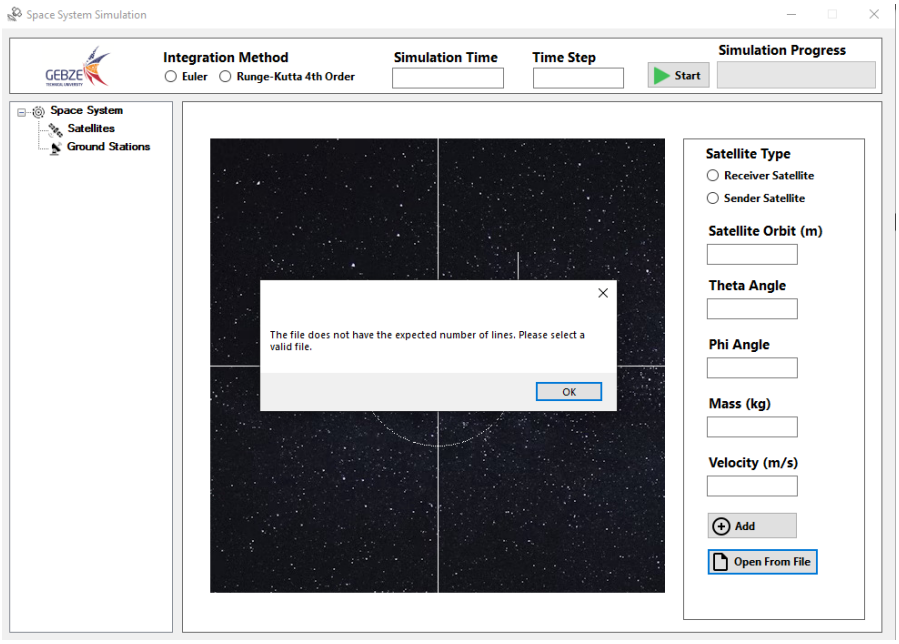


Figure 5. 10 Satellite wrong format file open result

The user can add any number of satellites to the program. You can access all added satellite information from the Satellites node on the left. The location is shown in Figure 5.11.

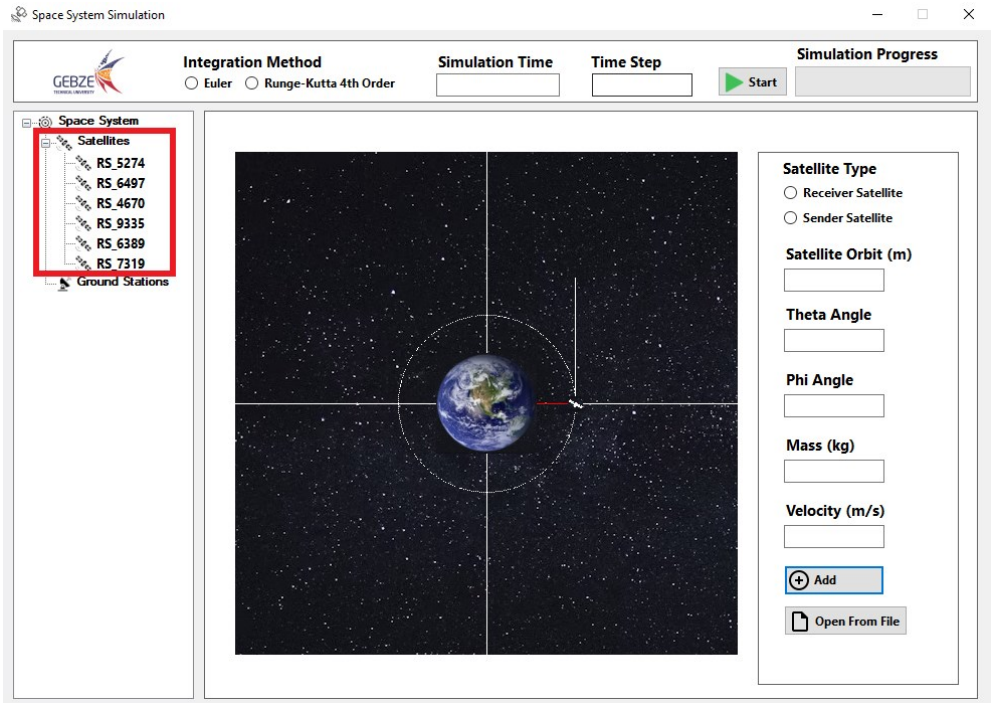


Figure 5. 11 Satellite Nodes Representation

5.2.1 Satellite Node Screen

In the child satellite node page, there are some differences between the main satellite node page. In figure 5.12 you can child satellite node page.

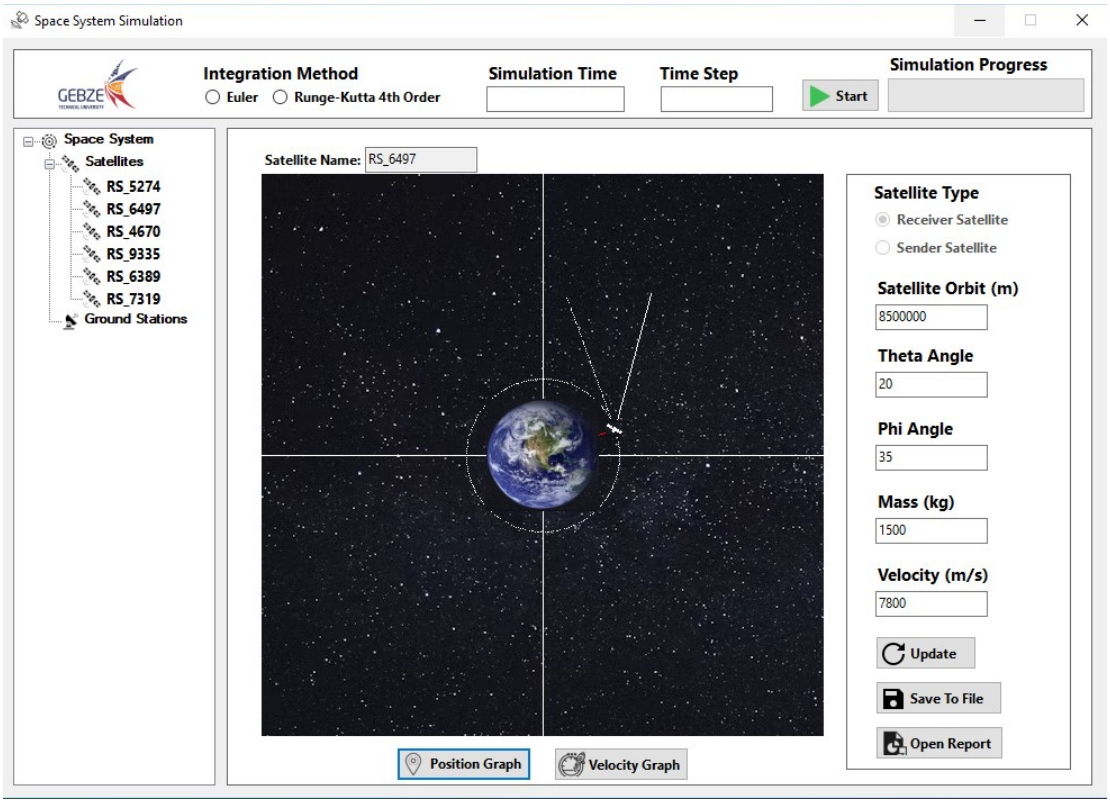


Figure 5. 12 Satellites Child node page

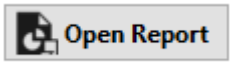
In this page we can see that on the right side there are the same text boxes as main satellite page, but they filled with the selected satellite properties. The radio buttons are frozen, so user cannot change after creating the satellite. There are also different types of buttons such as Update, Save To File, Open Reports, Position Graph, Velocity Graph. In the middle we can see the satellite representation.

Firstly, let's define how to works each button.

- **Update button** is a button that works to update the selected satellite.

- **Save to File button** is a button that allows the user to save file after creating it in this button the user also can save the file custom address.

- **Open Report button** this button works to open the satellite report. Note that before using this button, the user should



start the simulation.

- **Position Graph button** this button works to open satellite position graph with respect to simulation time. Note that before using this button, the user should start the simulation.

- **Velocity Graph button** this button works to open satellite velocity graph with respect to simulation time. Note that before using this button, the user should start the simulation.

5.3 Ground Stations Screen

If the user clicks the Ground Stations node, the user reaches the Ground Station page. In Figure 5.13 shows that ground station page.

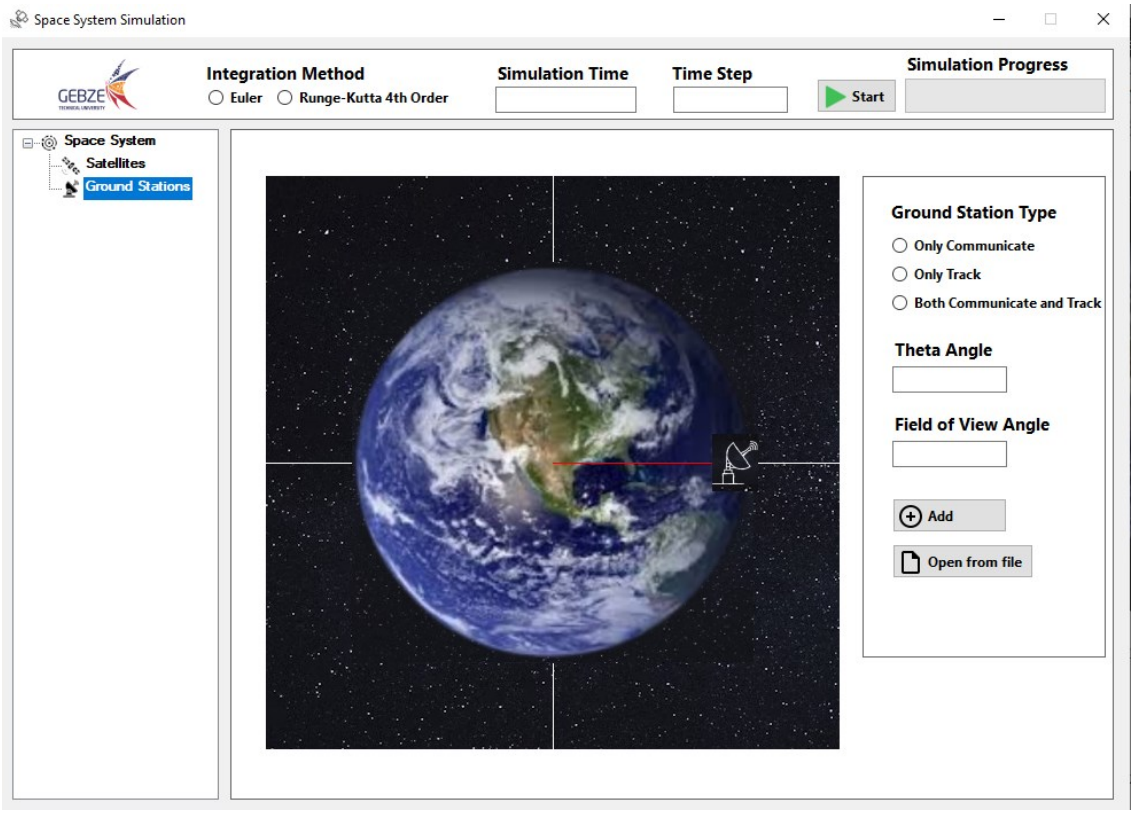
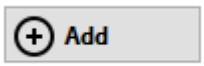


Figure 5. 13 Ground Station Page

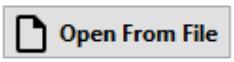
In this page we can see that on the right side there are some text boxes, radio buttons and buttons, and, in the middle, there is a ground station position representation.

Firstly, let's define how to works each button.

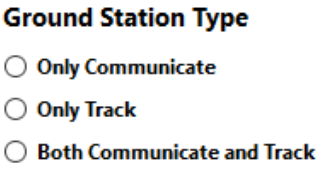
- **Add button** is a button that works to create a ground station and add it to the Ground Stations child node after the user has entered all the values correctly.



- **The Open from File button** is a button that allows the user to enter previously saved data without having to re-enter it.



There are some radio buttons called Only Communicate, Only Track, and Both Communicate and Track. This radio button works that determining the type of satellite.



- **Only Communicate** if the user clicks this radio button, the program will determine that the ground station type is Only Communicate Ground Station.
- **Only Track** if the user clicks this radio button, the program will determine that the ground station type is Only Track Ground Station.
- **Both Communicate and Track** if the user clicks this radio button, the program will determine that the ground station type is Both Communicate and Track Ground Station.

We have only two text boxes; we can understand that each text box indicates by the text written above it. In the Figure 5.14 shows that the ground station type is **Only Track Ground Station**, the ground station theta angle is ***50 degrees,*** the field of view angle is ***45 degrees.***

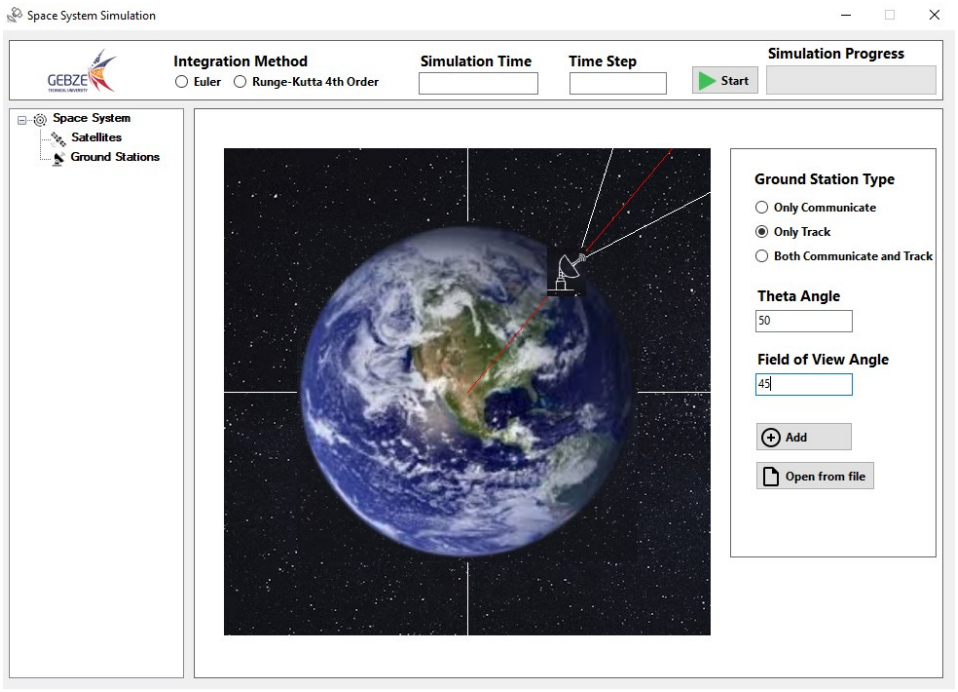


Figure 5. 14 Example of entering properties of the ground station

The **Add button** and the **Open From File button** work like satellite page. So you can check how it works.

The user can add any number of ground station to the program. You can access all added ground station information from the Ground Stations node on the left. The location is shown in Figure 5.15.

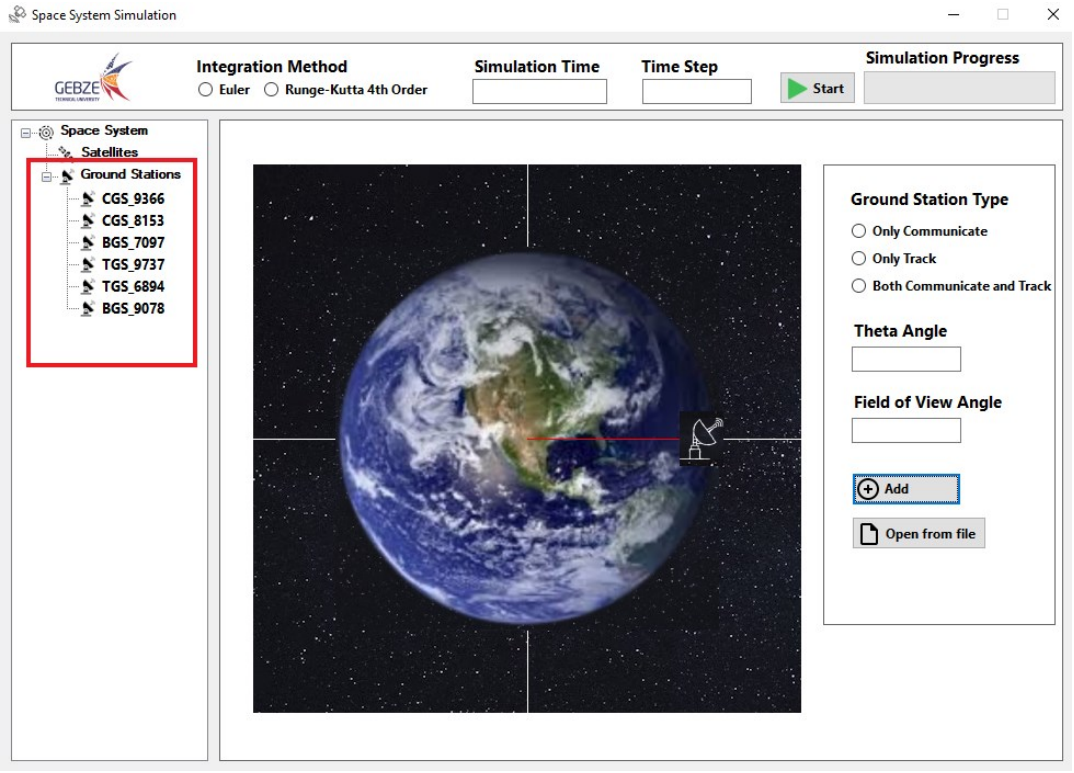


Figure 5. 15 Ground Station Nodes Representation

5.3.1 Ground Station Node Screen

In the child ground station node page, there are some differences between the main ground station node page. In figure 5.16 you can child ground station node page.

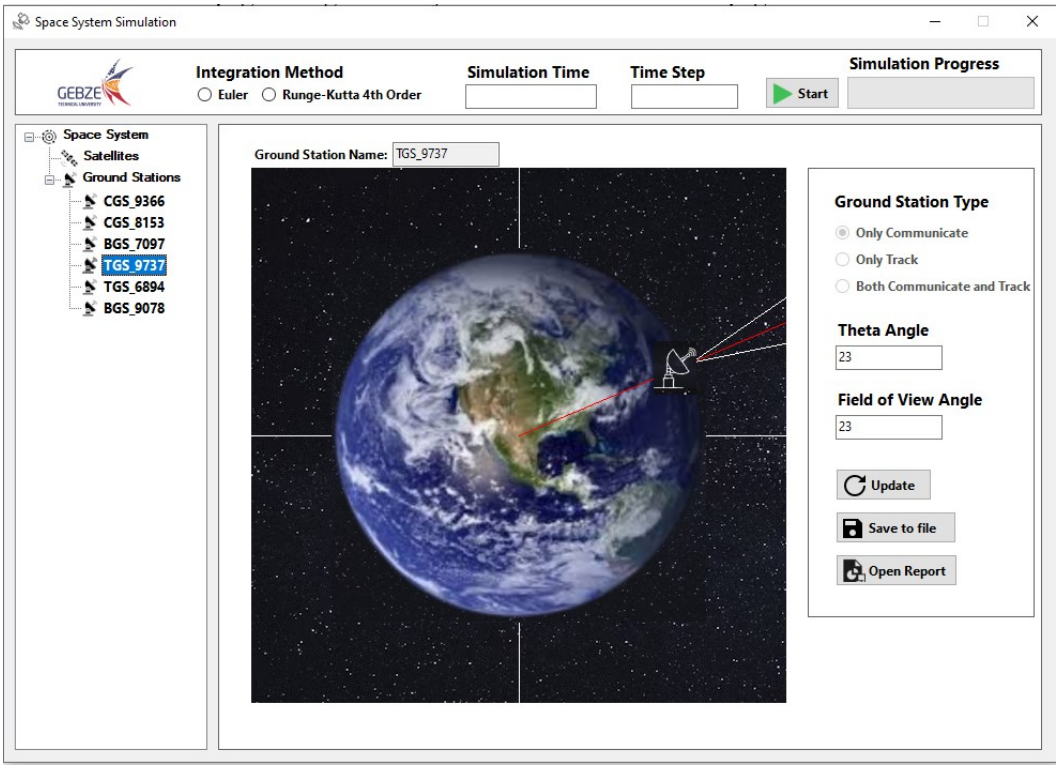


Figure 5. 16 Ground Station Child node page

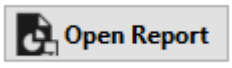
In this page we can see that on the right side there are the same text boxes as main ground station page, but they filled with the selected ground station properties. The radio buttons are frozen, so user cannot change after creating the ground station. There are also different types of buttons such as Update, Save To File, Open Reports. In the middle we can see the ground station representation.

Firstly, let's define how to works each button.

- **Update button** is a button that works to update the selected ground station.

- **Save to File button** is a button that allows the user to save file after creating it in this button the user also can save the file custom address.

- **Open Report button** this button works to open the ground



station report. Note that before using this button, the user should start the simulation.

5.4 Starting the Simulation

After the user has created a satellite and a ground station, the user can now start the simulation. **Note that**, if the user has not created a satellite or ground, the simulation cannot be started.

In Figure 5.17 you can see that a satellite and a ground have been created. So the simulation can be started after entering the green area.

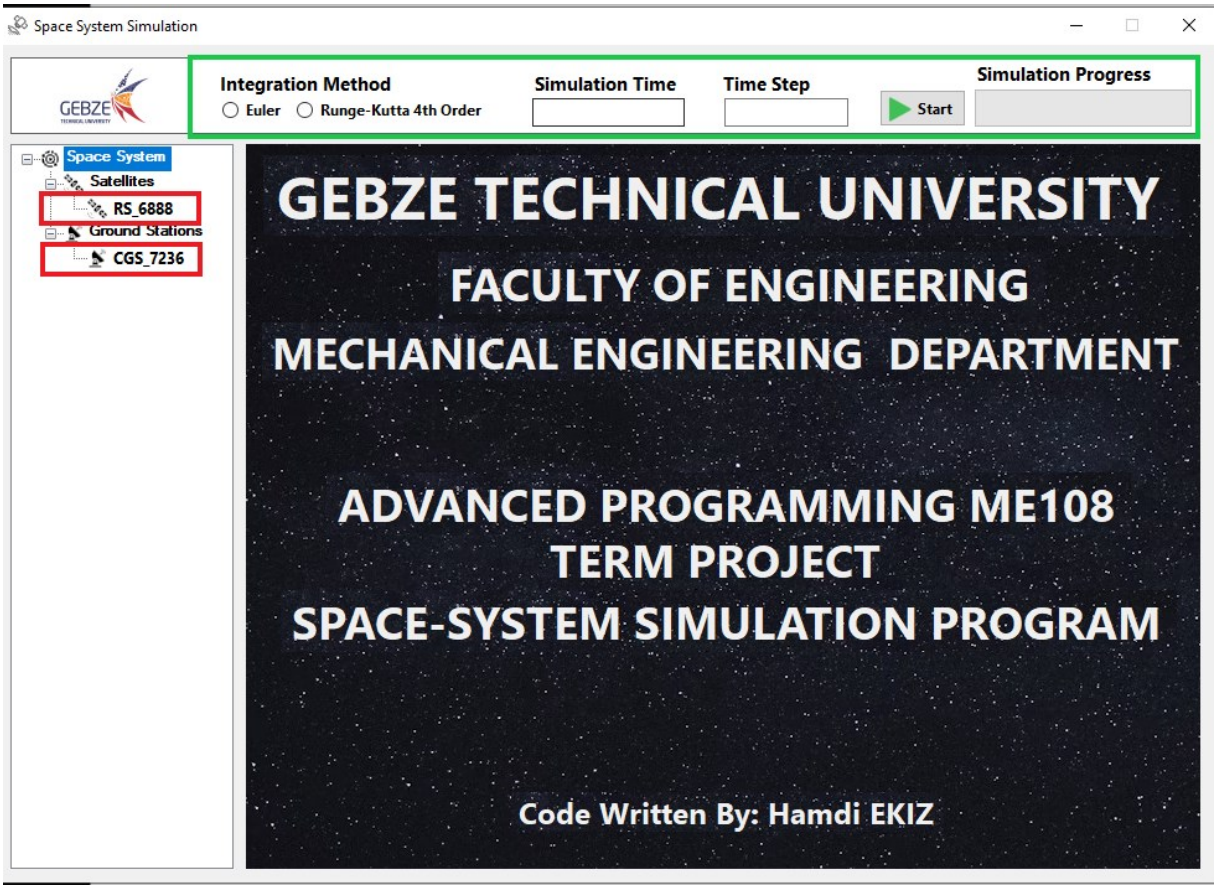


Figure 5. 17 Shows the simulation area

In that area we can see that there are two radio buttons, one button and two text boxes.

The user should select an integration method for clicking the related radio button, either Euler or Runge-Kutta 4th order method.

After selecting the integration method to be applied, the user should enter the simulation time and time step of the simulation.

After entering all related areas, the user can start the simulation by clicking 

the Start button.

Also, simulation progress can be shown in the progress bar. 

In Figure 5.18 you can see that the simulation parameter is entered, the integration method chosen as ***Runge-Kutta 4th Order***, simulation time is ***10000 seconds***, time step is ***0.1 seconds***. After starting the simulation, the progress bar moves and shows the progress of the simulation.

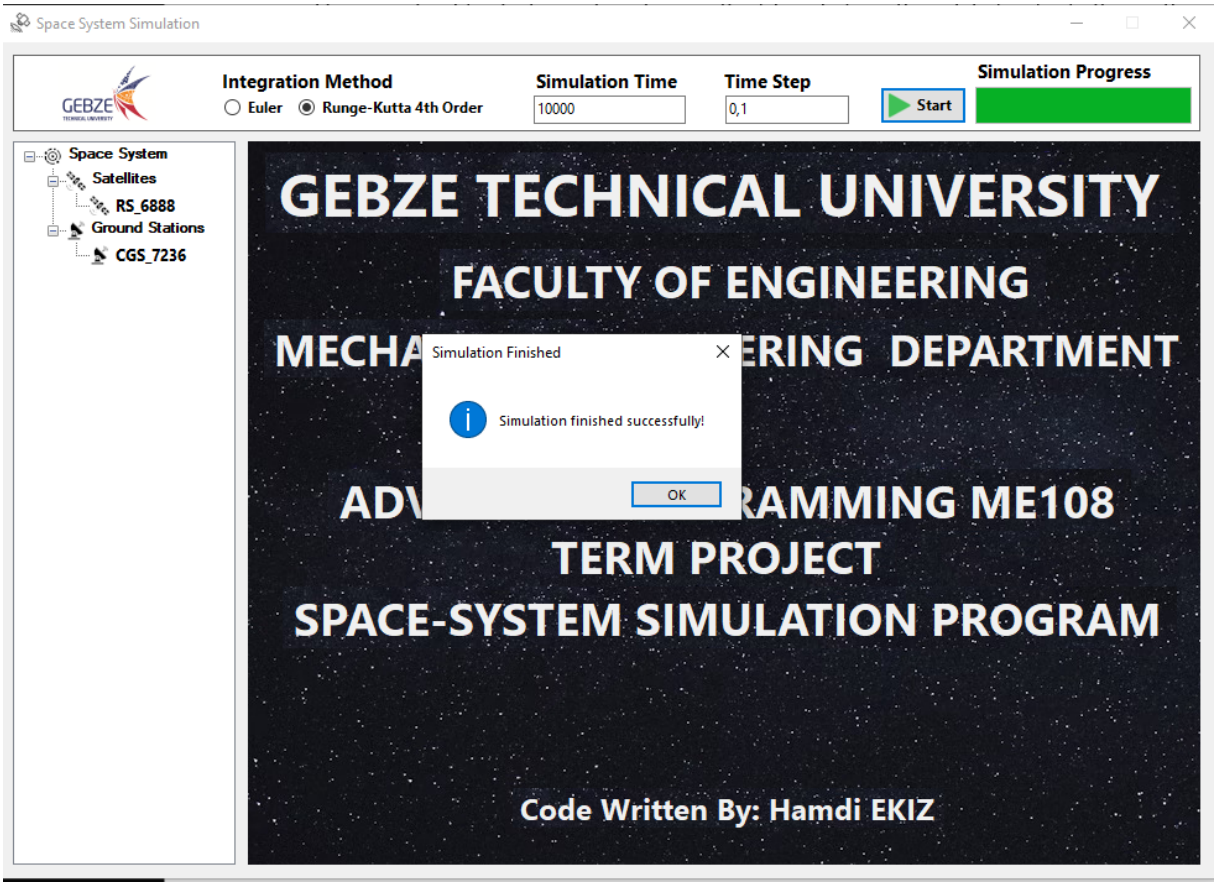


Figure 5. 18 Simulation finished message

Once the simulation is complete, we can now access both satellite and ground station reports. And also position and velocity graphs.

In figure 5.19 shows the position graph of the simulated satellite. You can access this graph by clicking the **Position Graph button.** You can also save the position graph onto your computer by clicking the **Save Image button.**

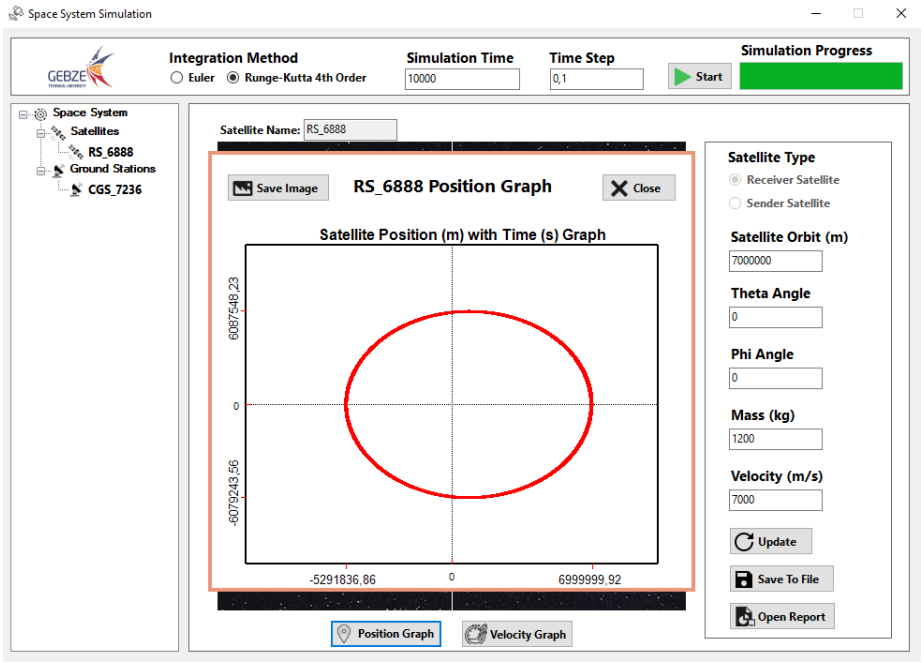


Figure 5. 19 Position Graph of the Satellite RS_6888

In figure 5.20 shows the velocity graph of the simulated satellite. You can access this graph by clicking the **Velocity Graph button.** You can also save the velocity graph onto your computer by clicking the **Save Image button.**

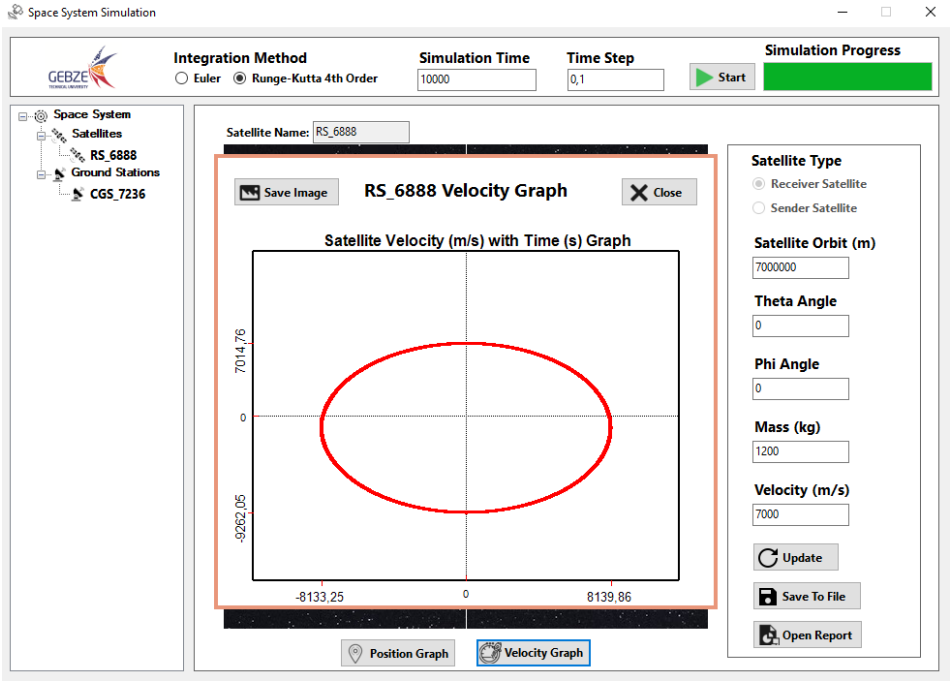


Figure 5. 20 Velocity Graph of the Satellite RS_6888

To illustrate the report algorithm perfectly, let's create more satellites and ground stations. Figure 5.21 shows that ***four*** **Receiver** and **Sender satellites** are created. Also created ***three*** **Only Communicate**, **Only Track**, and **Both Communicate and Track Ground Stations**.

Figure 5. 21 Starting a new simulation for representing report algorithm

In the figures below you can see all the messages that appear when the user clicks the **Open Report File button**. If the program cannot find a report, it gives an error message, which is also shown in the figures below.

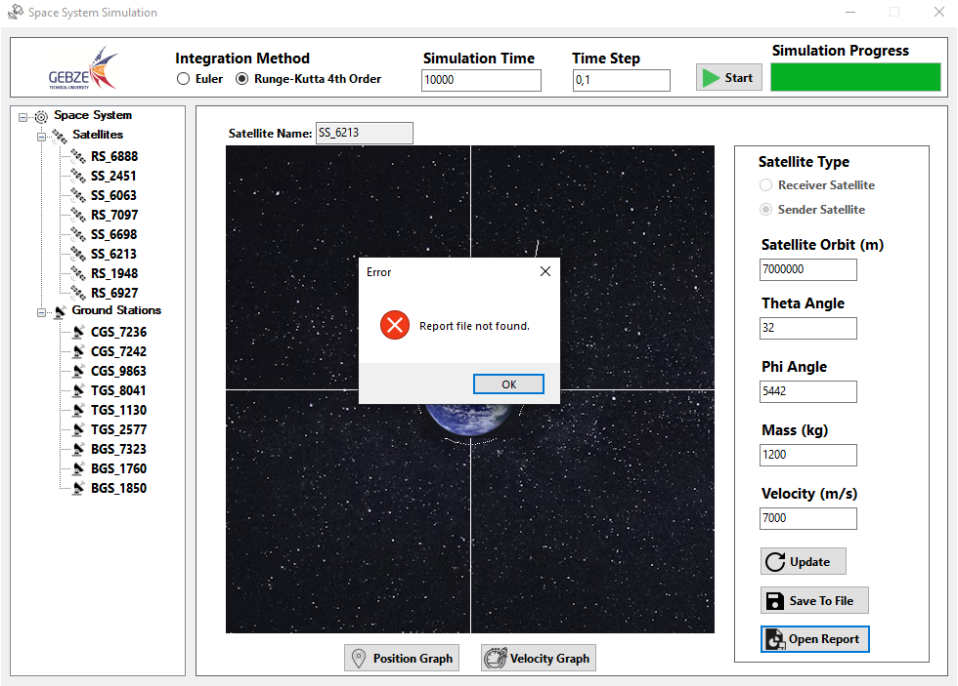


Figure 5. 22 Error message for not finding any report

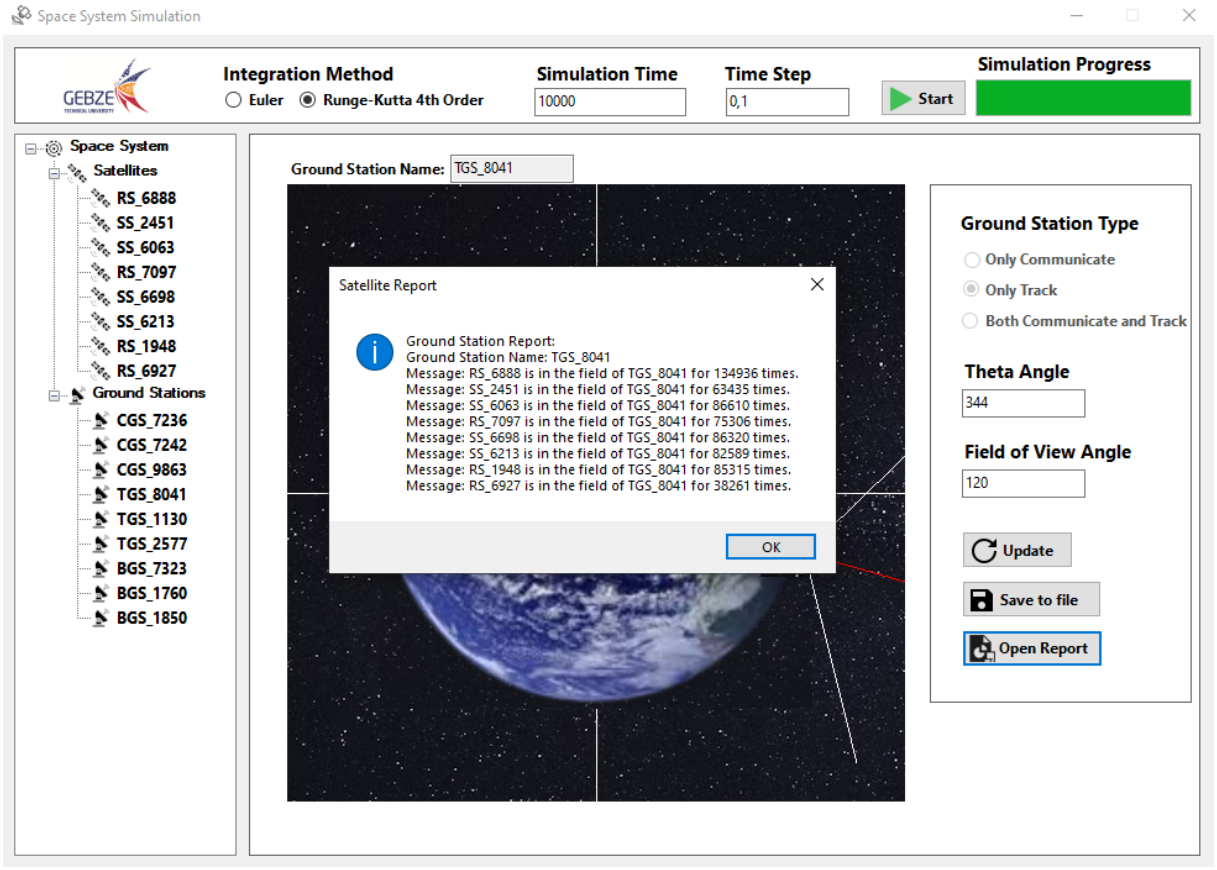


Figure 5. 23 Report file of the ground station TGS_8041

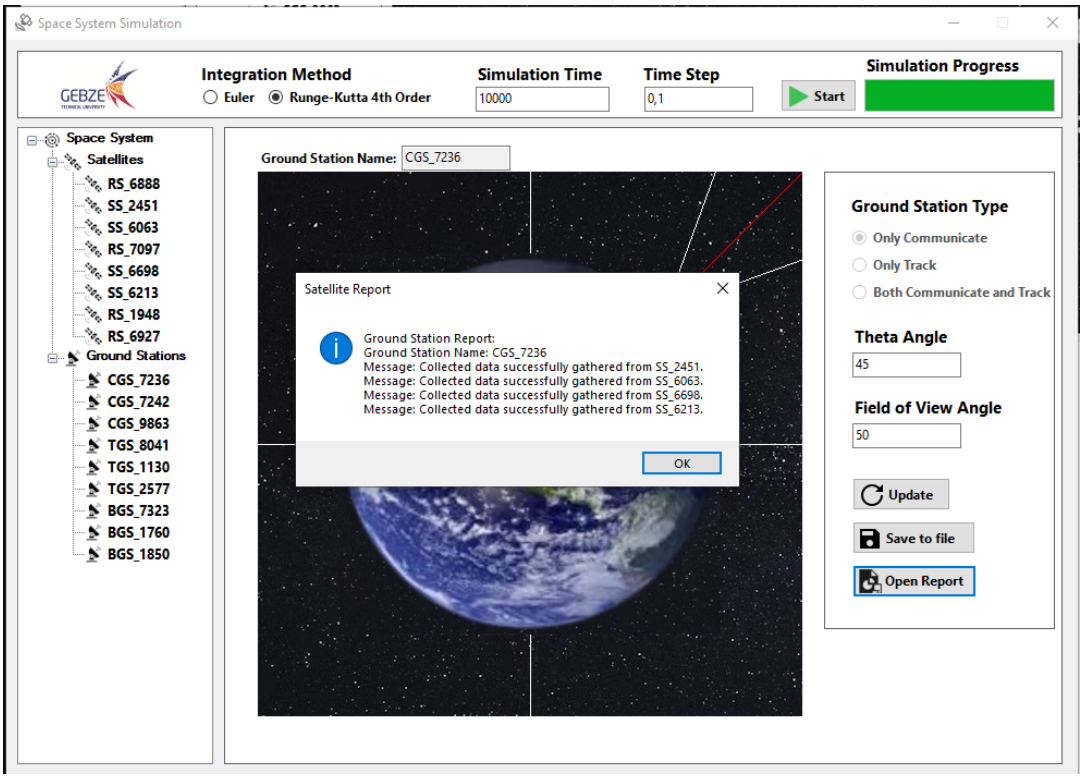


Figure 5. 24 Report file of the ground station CGS_8041

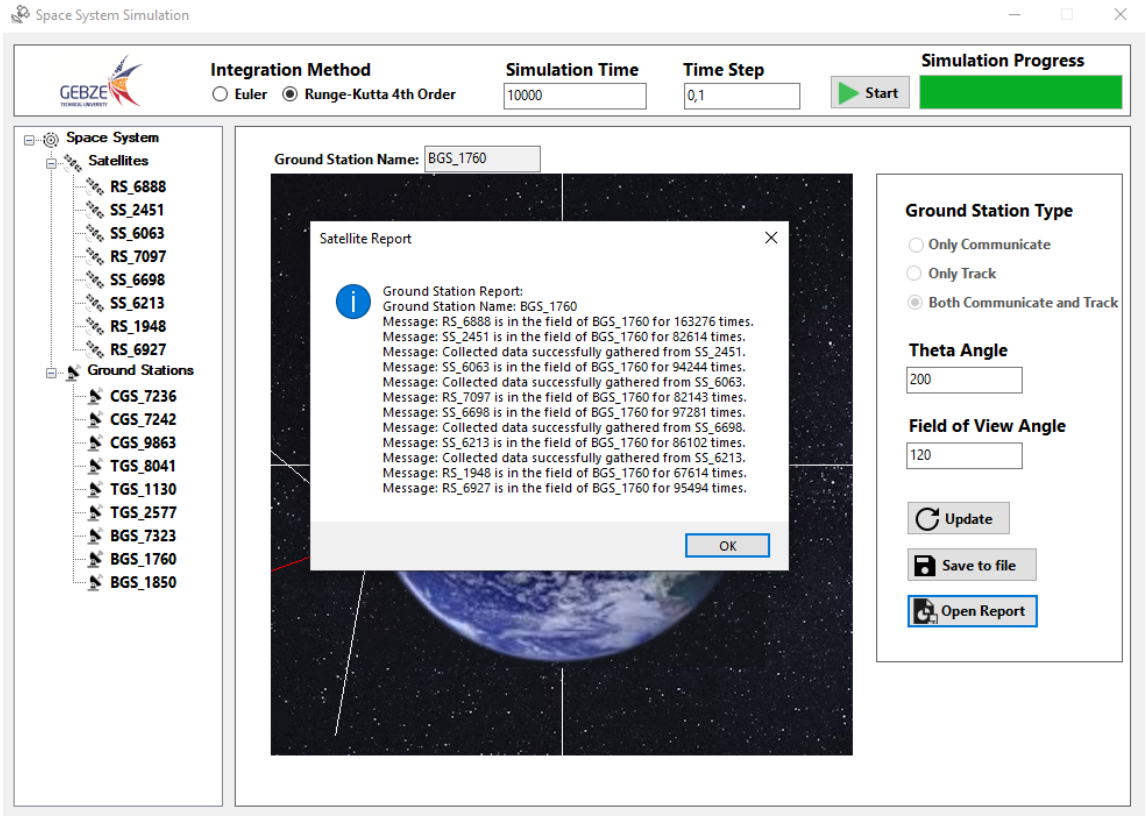


Figure 5. 25 Report file of the ground station BGS_8041

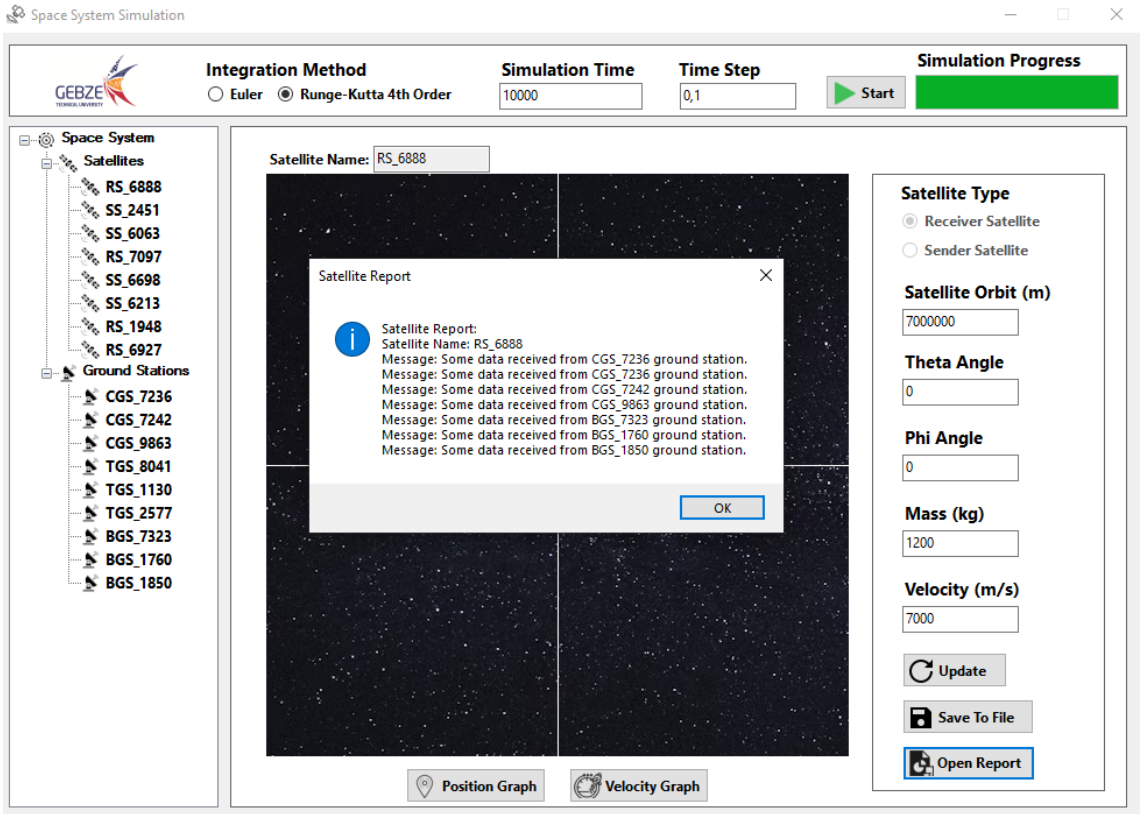


Figure 5. 26 Report file of the satellite RS_8041