Institut Superior de l'Aéronautique et de l'Espace

MAE 502: Object Oriented Software Development

SUPAERO's Satellite Space Mission: Project Report

Submitted on:
30 March, 2022

Authors:
Akash SHARMA, Harshini AICH

Introduction to the UML Diagrams

The report describes a space imaging mission which is coded using objects in JAVA based on shapes generated in UML. Use-Case, Sequence and Class diagrams are prepared in order to model the Satellite Space Mission. The class diagrams were then imported into Eclipse IDE as JAVA codes, and the classes were further developed. The final class diagram takes into account changes to the classes made in the course of code development.

The system models two main parts of the mission:

The first part requires the satellite to take good quality pictures of the ISAE-SUPAERO campus and send them to the ground station for quality inspections.
The second part focusses on the rocket launch where functions like fuel monitoring, engines release sequences and trajectory control are implemented through command sequences given by the ground station.

Objects are created for the various components of the mission which are: **Ground Station** on Earth, the **Rocket** which will be launched from Earth to carry the satellite to orbit, and the **Satellite Controller** mounted on the satellite.
The satellite has component classes like a **Camera** to capture images and a **Transmitter** to send the image along with coordinate and time data to the ground station.

Assumptions Taken

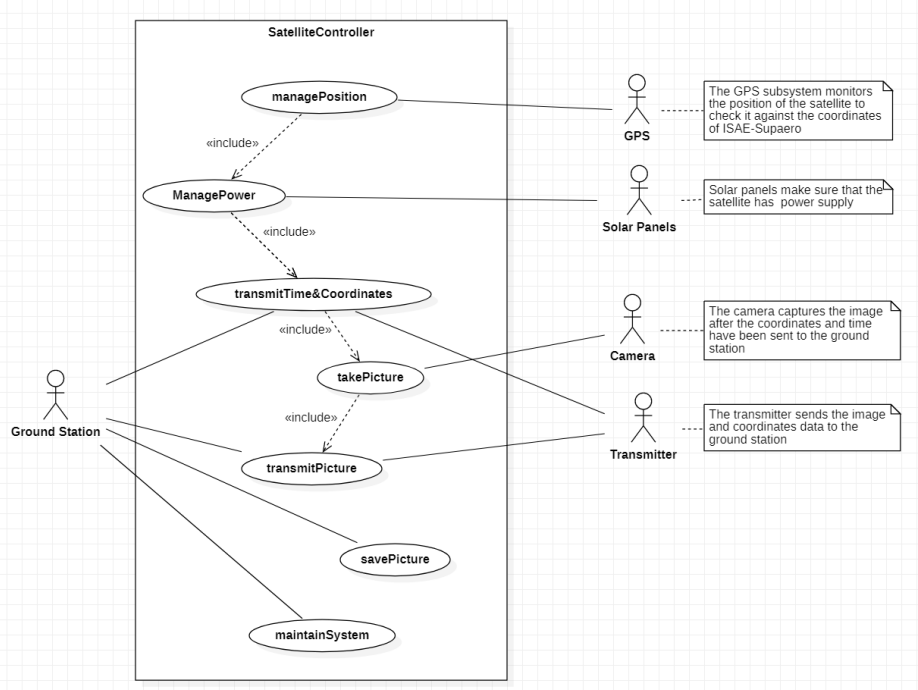
The diagrams were constructed taking into some reasonable assumptions that simplify the model, in order to simulate the most relevant and critical functions to be carried out by the satellite during the mission.

1. It is assumed that the satellite is entirely powered by the solar panels without any power storage. Image capturing and transmission is thus only possible when the panels are on.
2. We assume that the orbit chosen for the satellite passes over the ISAE-SUPAERO campus area and there is no requirement for Attitude Control and Orbital Adjustment to capture the image of the campus.
3. We have taken 1 set of coordinates to represent the location of the campus: 43.5655° N, 1.4740° E.
4. We assume that there is enough power to return a power “OFF” status even if the solar panels are off.
5. The rocket engine separation is assumed to be implemented in two stages. The fuel mass distribution between both engines is represented by a fuel ratio attribute.
6. Image data is assumed to be sent in the background. Only a message of “Image Sent” will be displayed by the concerned object when the transmission occurs. The actual image will not be transmitted.

1. UML diagrams: Use Case & Sequence

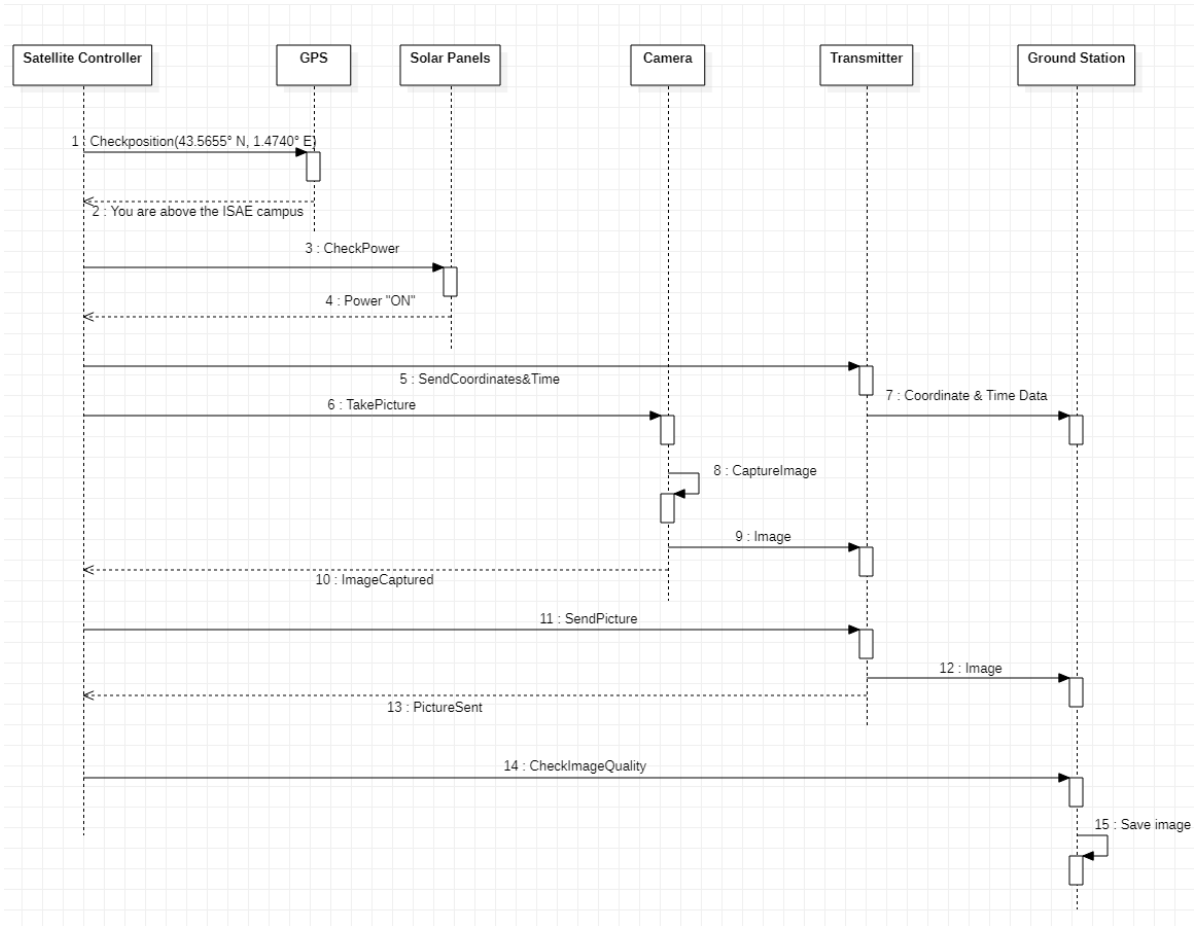
Use-Case Diagram

The actors for this UCD are all the subsystems that send/receive data and the ground control. The use case depicted here is for the first part of the specification, which is the image capture procedure.
The process is initiated when the SatelliteController confirms with the GPS that the satellite is above ISAE. Then it checks for available power, and transmits the current location and time coordinates. Next, it directs the camera to capture the image and this image is transmitted by the transmitter subsystem.
The connectors between the actions are <<include>> to indicate that each of the actions is necessary, and follow each other as consequent steps in the procedure.
The UCD also includes the ground station functions of checking the image quality and saving the image. An auxiliary function of system maintenance is added which is carried out by the ground station.



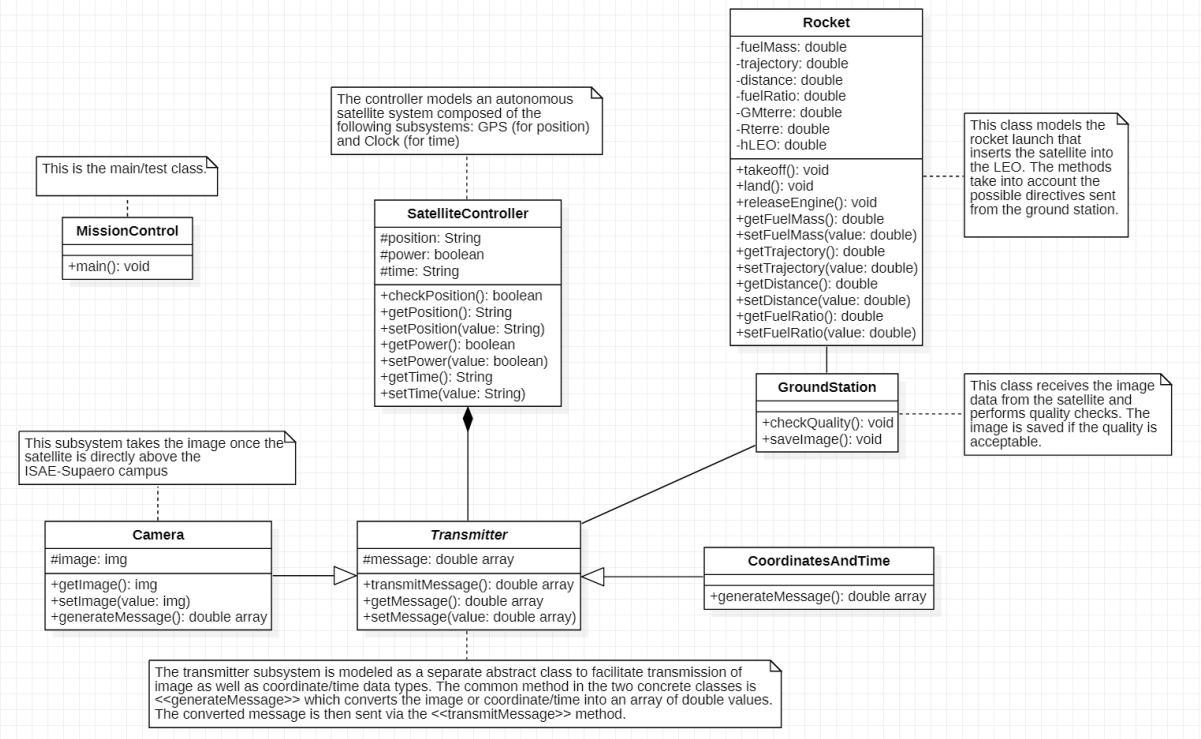
Sequence Diagram

The sequence diagram shows the messages that pass between the actors described in the UCD during the image capture procedure. The messages are structured as queries, replies, and data transmissions. This sequence follows the condition of power ON, and in the case of power OFF it is assumed that the image capture does not take place. Thus, there is no need for a sequence diagram for the power OFF case and it is assumed that the satellite is not operational until power is ON again.



2. UML Diagrams: Class Diagrams Comparison

First Draft



The **"SatelliteController" class** takes care of all the autonomous functions of the satellite platform like power management, position management and clock funtions.

An abstract class is used here to model the **Transmitter**, which can either send image data or time/coordinate data. Concrete classes of **"Camera" and "CoordinatesAndTime"** are implemented to process image data and coordinate data separately. The common method "generateMessage" returns an array of double values which can then be transmitted by the transmitter. The transmitter is a component of the satellite which is vital to its function, so a composition relation is used here.

This follows usual satellite transmitter technology, where images and coordinates are converted to RGB values or bits before transmission.

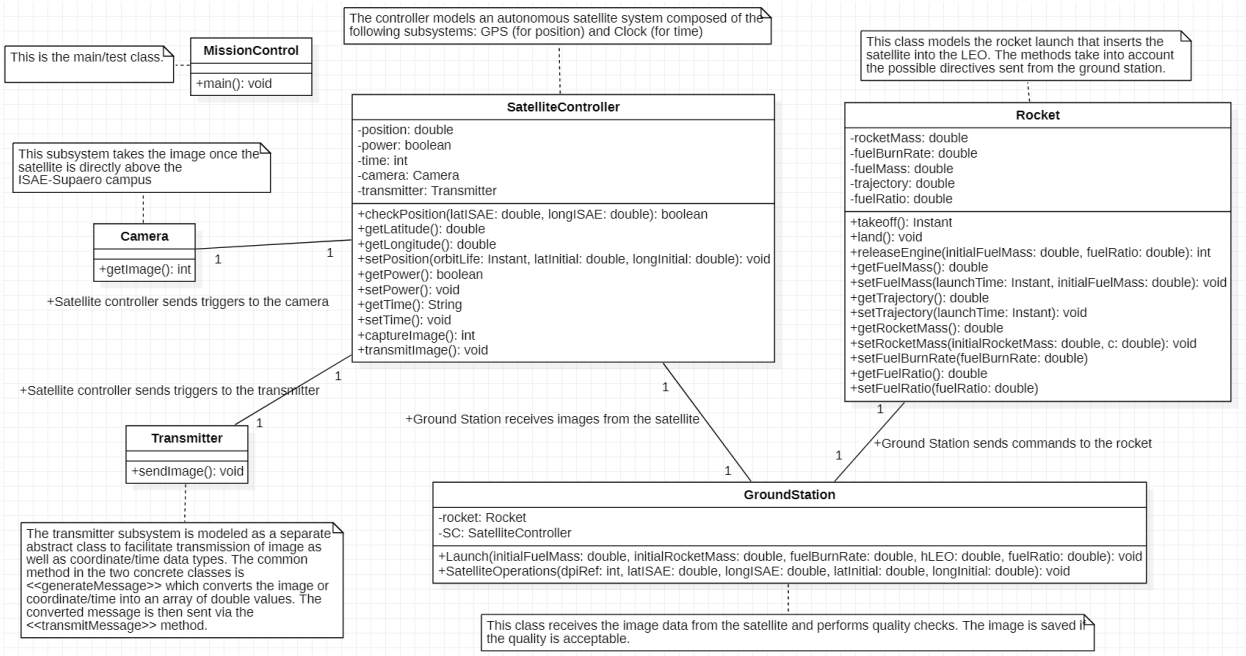
To account for the extended second part of the specification, the **"Rocket"** class is added which will be used for the satellite launch. The rocket contains the "trajectory" attribute to calculate the path through the release rate equation defines in the specification document. For this computation, it uses the attributes corresponding to distance from Earth, GM value, radius of earth and LEO altitude.

Engine release functions are also supported by the rocket, which is split into 2 stages. The "fuelRatio" attribute is defined as,

$$\boldsymbol{fuelRatio = \frac{fuel\ mass\ in\ Stage\ 1}{fuel\ mass\ in\ Stage\ 2}}$$

This ratio is used to determine when each engine stage will be released. Additionally, take-ff and landing functions are supported.

Final Version



The **"SatelliteController"** class takes care of all the autonomous functions of the satellite platform like power management and position management.

The controller directs the **"Camera"** class to capture the image each time the satellite passes over SUPAERO. The **"Transmitter"** class then sends this image to the ground station.

To account for the extended second part of the specification, the **"Rocket"** class is added which will be used for the satellite launch. The rocket contains the "trajectory" attribute to calculate the altitude. The hLEO constant is used to check if orbit is reached, and the other constants are not used to prevent over-constraining the problem.

Engine release functions are also supported by the rocket, where the engine is split into 2 stages. The "fuelRatio" attribute is defined as,

$$\boldsymbol{fuelRatio = \frac{fuel\ mass\ after\ separation}{fuel\ mass\ before\ separation}}$$

This ratio is used to determine when each engine stage will be released. Take-off and landing functions are also supported.

3. Introduction to the JAVA code

PART-I

The satellite controller conducts its mission once the satellite has been inserted into orbit by the rocket launcher. The mission for the satellite controller is defined as: "*Take pictures each time the satellite passes over ISAE-SUPAERO*." The code simulates the ground track of a real satellite (shown in Figure 1) to evaluate the code performance in an actual mission. Capturing of the image is initiated when the satellite is within a 2° radius of ISAE-SUPAERO.

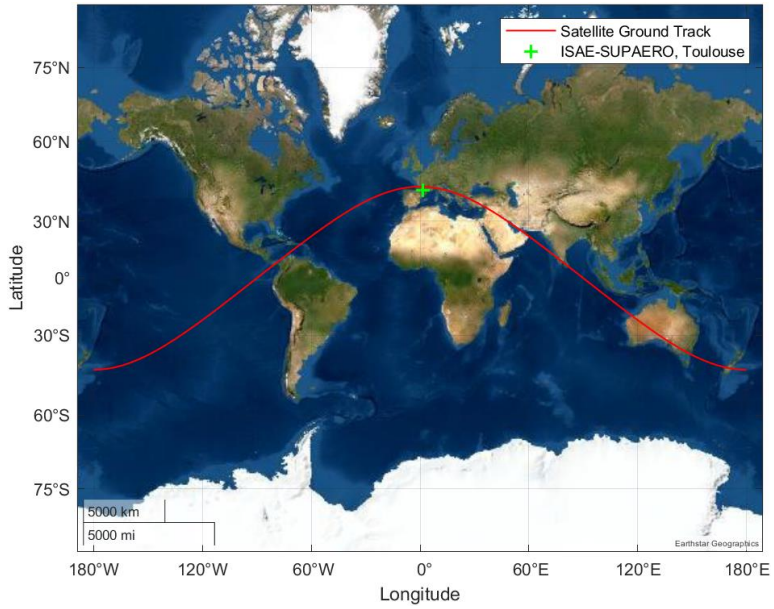


Figure 1: Ground track (marked in red) of a satellite passing close to ISAE-SUPAERO, Toulouse, France (marked in green).

PART-II

The mission for the rocket is defined as: "*Carry a satellite to space and place it in a desired orbit (at 750 km altitude)*." The functions of the rocket include launching it from Earth, tracking trajectory and fuel, release of engines as the fuel is consumed and landing the released engines. The commands for these functions are sent from the ground station.

The rocket is assumed to have two stages: the first stage engine is released when 50% fuel is remaining, and the second stage is released when all the fuel is burnt. Each stage is instructed to land immediately after separation.
The rocket is programmed to reach orbit using the available fuel, so when the second stage is released, the satellite has been inserted into orbit. From this point the rocket class is no longer used and the satellite controller takes over the mission.

4. JAVA Code

Mission Control Class

The Mission Control (MC) class begins the mission by creating the GroundStation class "GS". The various operations of the mission will be carried out within this GroundStation class. This class will create objects of the Rocket class, the SatelliteController class, the Camera class and the transmitter class.The MC class calls two methods of the GS, **GS.Launch()** to start the rocket launch phase and **GS.SatelliteOperations()** to start the satellite operations phase.

Ground Station Class

The Ground Station (GS) class is responsible for two phases of the mission:

1. Rocket launch
2. Image processing

For the first phase, the GS creates a Rocket object and specifies its attributes. It then calls the relevant methods defined in the Rocket class. These methods are called in sequence with appropriate loops and conditional statements, to complete the events starting from Rocket launch and ending with Satellite Insertion in orbit.
The **launch()** method assigns the following attributes to the rocket: fuelMass, fuelBurnRate, fuelRatio, trajectory and hLEO. It also begins the iteration loop to decrement fuel (**setFuel()**) and increment the altitude (**setTrajectory()**) of the rocket. This loop also contains conditions to call the **releaseEngine()** method defined in the rocket class.

```
We have liftoff!

Fuel Mass = 119860.0 kg          Altitude = 0.876 km             Time = 1.3 seconds
Fuel Mass = 119720.0 kg          Altitude = 1.752 km             Time = 2.89 seconds
Fuel Mass = 119580.0 kg          Altitude = 2.628 km             Time = 4.49 seconds
Fuel Mass = 119440.0 kg          Altitude = 3.504 km             Time = 5.99 seconds
Fuel Mass = 119300.0 kg          Altitude = 4.38 km              Time = 7.48 seconds
Fuel Mass = 119160.0 kg          Altitude = 5.256 km             Time = 9.08 seconds
Fuel Mass = 119020.0 kg          Altitude = 6.132 km             Time = 10.57 seconds
Fuel Mass = 118880.0 kg          Altitude = 7.008 km             Time = 12.17 seconds
Fuel Mass = 118740.0 kg          Altitude = 7.884 km             Time = 13.67 seconds
Fuel Mass = 118600.0 kg          Altitude = 8.76 km              Time = 15.16 seconds
Fuel Mass = 118460.0 kg          Altitude = 9.636 km             Time = 16.76 seconds
Fuel Mass = 118320.0 kg          Altitude = 10.512 km            Time = 18.35 seconds
Fuel Mass = 118180.0 kg          Altitude = 11.388 km            Time = 19.91 seconds
Fuel Mass = 118040.0 kg          Altitude = 12.264 km            Time = 21.5 seconds
Fuel Mass = 117900.0 kg          Altitude = 13.14 km             Time = 23.05 seconds
Fuel Mass = 117760.0 kg          Altitude = 14.016 km            Time = 24.57 seconds
Fuel Mass = 117620.0 kg          Altitude = 14.892 km            Time = 26.12 seconds
Fuel Mass = 117480.0 kg          Altitude = 15.768 km            Time = 27.68 seconds
Fuel Mass = 117340.0 kg          Altitude = 16.644 km            Time = 29.23 seconds
Fuel Mass = 117200.0 kg          Altitude = 17.52 km             Time = 30.74 seconds
Fuel Mass = 117060.0 kg          Altitude = 18.396 km            Time = 32.33 seconds
Fuel Mass = 116920.0 kg          Altitude = 19.272 km            Time = 33.91 seconds
```

The status of the rocket is printed as shown in the result images. For the second phase, the GS creates a Satellite Controller object. This tracks the satellite and its status.

When this Controller directs the satellite to transmit an image, the GS evaluated the image. If the transmitted pixels are more than 300/sq inch, the image is saved.

```
Satellite coordinates = 34.5°N 40.0°W          Timestamp = mercredi 30 mars 2022 à 23:29:37 heure d'été d'Europe centrale          Power = ON
Satellite coordinates = 36.4°N 36.0°W          Timestamp = mercredi 30 mars 2022 à 23:29:37 heure d'été d'Europe centrale          Power = ON
Satellite coordinates = 38.2°N 32.0°W          Timestamp = mercredi 30 mars 2022 à 23:29:37 heure d'été d'Europe centrale          Power = ON
Satellite coordinates = 39.7°N 28.0°W          Timestamp = mercredi 30 mars 2022 à 23:29:37 heure d'été d'Europe centrale          Power = ON
Satellite coordinates = 41.1°N 24.0°W          Timestamp = mercredi 30 mars 2022 à 23:29:37 heure d'été d'Europe centrale          Power = ON
Satellite coordinates = 42.3°N 20.0°W          Timestamp = mercredi 30 mars 2022 à 23:29:37 heure d'été d'Europe centrale          Power = ON
Satellite coordinates = 43.3°N 16.0°W          Timestamp = mercredi 30 mars 2022 à 23:29:37 heure d'été d'Europe centrale          Power = ON
Satellite coordinates = 44.0°N 12.0°W          Timestamp = mercredi 30 mars 2022 à 23:29:37 heure d'été d'Europe centrale          Power = ON
Satellite coordinates = 44.6°N 8.0°W           Timestamp = mercredi 30 mars 2022 à 23:29:37 heure d'été d'Europe centrale          Power = ON
Satellite coordinates = 44.9°N 4.0°W           Timestamp = mercredi 30 mars 2022 à 23:29:37 heure d'été d'Europe centrale          Power = ON

Picture taken
Image transmitted from the Satellite.
Image received at Ground Station

The quality is good
The image is saved
```

If the image quality is not good, the image is discarded.

```
Satellite coordinates = 32.4°N 44.0°W          Timestamp = mercredi 30 mars 2022 à 23:37:58 heure d'été d'Europe centrale          Power = ON
Satellite coordinates = 34.5°N 40.0°W          Timestamp = mercredi 30 mars 2022 à 23:37:58 heure d'été d'Europe centrale          Power = ON
Satellite coordinates = 36.4°N 36.0°W          Timestamp = mercredi 30 mars 2022 à 23:37:58 heure d'été d'Europe centrale          Power = ON
Satellite coordinates = 38.2°N 32.0°W          Timestamp = mercredi 30 mars 2022 à 23:37:58 heure d'été d'Europe centrale          Power = ON
Satellite coordinates = 39.7°N 28.0°W          Timestamp = mercredi 30 mars 2022 à 23:37:58 heure d'été d'Europe centrale          Power = ON
Satellite coordinates = 41.1°N 24.0°W          Timestamp = mercredi 30 mars 2022 à 23:37:58 heure d'été d'Europe centrale          Power = ON
Satellite coordinates = 42.3°N 20.0°W          Timestamp = mercredi 30 mars 2022 à 23:37:58 heure d'été d'Europe centrale          Power = ON
Satellite coordinates = 43.3°N 16.0°W          Timestamp = mercredi 30 mars 2022 à 23:37:58 heure d'été d'Europe centrale          Power = ON
Satellite coordinates = 44.0°N 12.0°W          Timestamp = mercredi 30 mars 2022 à 23:37:58 heure d'été d'Europe centrale          Power = ON
Satellite coordinates = 44.6°N 8.0°W           Timestamp = mercredi 30 mars 2022 à 23:37:58 heure d'été d'Europe centrale          Power = ON
Satellite coordinates = 44.9°N 4.0°W           Timestamp = mercredi 30 mars 2022 à 23:37:58 heure d'été d'Europe centrale          Power = ON

Picture taken
Image transmitted from the Satellite.
Image received at Ground Station

The quality is not good enough
```

When the satellite completes 7 orbits the satellite mission is declared completed.

```
Satellite coordinates = 41.1°N 24.0°E          Timestamp = mercredi 30 mars 2022 à 23:29:45 heure d'été d'Europe centrale          Power = ON
Satellite coordinates = 39.7°N 28.0°E          Timestamp = mercredi 30 mars 2022 à 23:29:45 heure d'été d'Europe centrale          Power = ON
Satellite coordinates = 38.2°N 32.0°E          Timestamp = mercredi 30 mars 2022 à 23:29:45 heure d'été d'Europe centrale          Power = ON
Satellite coordinates = 36.4°N 36.0°E          Timestamp = mercredi 30 mars 2022 à 23:29:45 heure d'été d'Europe centrale          Power = ON
Satellite coordinates = 34.5°N 40.0°E          Timestamp = mercredi 30 mars 2022 à 23:29:45 heure d'été d'Europe centrale          Power = ON
Satellite coordinates = 32.4°N 44.0°E          Timestamp = mercredi 30 mars 2022 à 23:29:45 heure d'été d'Europe centrale          Power = ON
Satellite coordinates = 30.1°N 48.0°E          Timestamp = mercredi 30 mars 2022 à 23:29:45 heure d'été d'Europe centrale          Power = ON
Satellite coordinates = 27.7°N 52.0°E          Timestamp = mercredi 30 mars 2022 à 23:29:45 heure d'été d'Europe centrale          Power = ON
Satellite coordinates = 25.2°N 56.0°E          Timestamp = mercredi 30 mars 2022 à 23:29:45 heure d'été d'Europe centrale          Power = ON
Satellite coordinates = 22.5°N 60.0°E          Timestamp = mercredi 30 mars 2022 à 23:29:45 heure d'été d'Europe centrale          Power = ON
Satellite coordinates = 19.7°N 64.0°E          Timestamp = mercredi 30 mars 2022 à 23:29:45 heure d'été d'Europe centrale          Power = ON
Satellite coordinates = 16.9°N 68.0°E          Timestamp = mercredi 30 mars 2022 à 23:29:45 heure d'été d'Europe centrale          Power = ON
Satellite coordinates = 13.9°N 72.0°E          Timestamp = mercredi 30 mars 2022 à 23:29:45 heure d'été d'Europe centrale          Power = ON
Satellite coordinates = 10.9°N 76.0°E          Timestamp = mercredi 30 mars 2022 à 23:29:45 heure d'été d'Europe centrale          Power = ON

The mission of the satellite has ended.
Au revoir, bissou, bissou :(
```

Rocket Class

The **takeoff()** method starts the mission clock. The **land()** function facilitates landing of the reusable rocket stages after separation.The **releaseEngines()** method is used to initiate separation when fuel remaining is 50% (first stage released) and when all the fuel is burnt. The function throws up a flag when 50% fuel is burnt, and later when orbit altitude is reached the second stage is released.

```
Fuel Mass = 61760.0 kg          Altitude = 364.416 km          Time = 647.74 seconds
Fuel Mass = 61620.0 kg          Altitude = 365.292 km          Time = 649.3 seconds
Fuel Mass = 61480.0 kg          Altitude = 366.168 km          Time = 650.9 seconds
Fuel Mass = 61340.0 kg          Altitude = 367.044 km          Time = 652.54 seconds
Fuel Mass = 61200.0 kg          Altitude = 367.92 km           Time = 654.1 seconds
Fuel Mass = 61060.0 kg          Altitude = 368.796 km          Time = 655.74 seconds
Fuel Mass = 60920.0 kg          Altitude = 369.672 km          Time = 657.37 seconds
Fuel Mass = 60780.0 kg          Altitude = 370.548 km          Time = 658.85 seconds
Fuel Mass = 60640.0 kg          Altitude = 371.424 km          Time = 660.41 seconds
Fuel Mass = 60500.0 kg          Altitude = 372.3 km            Time = 662.01 seconds
Fuel Mass = 60360.0 kg          Altitude = 373.176 km          Time = 663.56 seconds
Fuel Mass = 60220.0 kg          Altitude = 374.052 km          Time = 665.12 seconds
Fuel Mass = 60080.0 kg          Altitude = 374.928 km          Time = 666.66 seconds

1st stage separation successful.
Landing initiated
Rocket Mass = 7500.0kg
```

The **land()** function is called after each stage is released, to send it back to Earth to be reused.

```
Fuel Mass = 440.0 kg            Altitude = 748.104 km          Time = 1483.37 seconds
Fuel Mass = 300.0 kg            Altitude = 748.98 km           Time = 1484.95 seconds
Fuel Mass = 160.0 kg            Altitude = 749.856 km          Time = 1486.51 seconds
Fuel Mass = 20.0 kg             Altitude = 750.732 km          Time = 1488.12 seconds

2nd stage separation successful.
Landing initiated
Rocket Mass = 0.0kg


Orbit reached


The satellite has started its orbit
```

The fuel mass is tracked using **getFuelMass()** and decremented using **setFuelMass()**. The latter also initailises the fuel mass to **initialFuelMass** if the rocket has just launched (delta t = 0).
The **getTrajectory()** method tracks the altitude of the rocket, and the **setTrajectory()** method increments it. The latter also initailises the trajectory to zero if the rocket has just launched (delta t=0).

The mass of the rocket that is propelling the satellite is tracked by **getRocketMass()**. The method **setRocketMass()** updates the rocket mass based on what stage of separation has already happened. It is set to 100% initially, updated to 50% once the first stage has separated and finally set to zero when the second stage separates and only the satellite remains.
setFuelBurnRate() sets the comsumption rate of the rockt fuel and **getFuelBurnRate()** returns its value of 140 as set in the specification document.
Fuel ratio is declared in setFuelRatio() and its value of 0.5 is returned by getFuelRatio().

Satellite Controller Class

The Satellite Controller creates two objects: a Camera and a Transmitter, as subsystems of the satellite. It contains the following functions:

1. **checkPosition()** to determine if it is above ISAE-SUPAERO
2. **getLattitude()** and getLongitude() to determine current position
3. **setPosition()** to update the position and simulate the orbit in Figure 1
4. **getPower()** to determine of the solar panels are ON/OFF
5. **setPower()** to set panels to ON/OFF depending on whether orbit is in Earth's shadow
6. **getTime()** to determine the current time
7. **captureImage()** to command the camera to take a picture
8. **transmitImage()** to command the transmitter to send the captured image