

# POLITECNICO DI TORINO

**Corso di Laurea Magistrale  
in Ingegneria Matematica**

**Tesi di Laurea Magistrale**

**Assessment of the performance of a visual payload mounted on a small satellite involved in an in-orbit servicing mission during the observation phase.**



**Relatori**

Sabrina Corpino  
Fabrizio Stesina  
Guglielmo Daddi

**Candidato**

Matteo Paschero

Anno Accademico 2021-2022



## Summary

1	Abstract .....	9
2	State of Art .....	11
2.1	Computer Graphics.....	11
2.1.1	Computer Graphics Principles .....	12
2.1.2	3D Graphic Design.....	12
2.1.3	Animation .....	13
2.1.3.1	Animation: Computational Cost.....	14
2.1.4	Computer Science .....	14
2.2	Graphical simulation for engineering.....	14
2.3	3D graphic design programs .....	18
2.3.1	3D graphic design: paid-for programs.....	18
2.3.2	3D graphic design: free programs.....	20
2.4	Blender.....	21
2.5	PANGU .....	22
2.6	Space Scene Simulation .....	23
2.7	Render Engines.....	24
2.7.1	Render Engine: Cycles .....	25
2.7.2	Render Engine: Eevee.....	27
2.7.3	Render Engine: Mitsuba2 .....	27
3	Radiometrically accurate Model .....	28
3.1	Radiometry vs. Photometry .....	28
3.2	Creation of light source .....	29
3.2.1	Parameters: Strength .....	29
3.2.1.1	Irradiance and Illuminance .....	30
3.2.2	Solid Angle .....	30
3.2.3	Rotation .....	33
3.2.4	Color .....	34
3.2.4.1	BlackBody Shader Node.....	35
3.2.4.2	Python .colour library .....	36
3.3	Scene Exposure .....	38
3.4	Light Model .....	39
4	Scene Calibration: Exposure.....	40
4.1	Calibration Method .....	40
4.1.1	Totally Dark Sky.....	41
4.1.2	No White.....	45
4.1.3	Back: total information.....	47
4.1.4	Literature mean value .....	51
4.2	Calibration with a totally reflective plane .....	54
4.2.1	Exposure analysis and evaluation .....	55
4.3	Exposure in Blender.....	58
5	Image Format .....	61
5.1	Image Export Format: OpenEXR.....	61

5.2	Image Export Format: PNG .....	62
<b>6</b>	<b>From Scene to Simulated Image .....</b>	<b>64</b>
6.1	SCENE.....	64
6.1.1	SCENE: Analysis Workflow.....	64
6.1.2	From OpenEXR/PNG to RGB map .....	65
6.1.3	RGB values to Relative Luminance.....	65
6.1.4	Relative Luminance to Light Flux .....	66
6.1.5	Photons Number Calculation.....	66
6.1.6	Output and Results.....	67
6.2	OPTICS.....	67
6.2.1	OPTICS: Irradiance.....	68
6.2.2	OPTICS: Relative illumination .....	68
6.2.3	OPTICS: Blurring .....	69
6.2.3.1	Blurring: Diffraction-limited optics .....	69
6.2.3.2	Blurring: Lens Flare from the Sun.....	69
6.2.3.3	Blurring: Shift-Invariant image formation .....	70
6.3	SENSOR .....	70
6.3.1	SENSOR: Photodiode .....	71
6.3.2	SENSOR: Numerical Integration.....	72
6.3.3	SENSOR: Real effects .....	73
6.3.4	SENSOR: Noise .....	73
6.3.4.1	Photon shot noise .....	74
6.3.4.2	Dark current.....	74
6.3.5	SENSOR: Pixel fill factor .....	74
6.3.6	SENSOR: Radiative Environment.....	75
6.4	CONVERTER.....	76
6.5	PROCESSOR and DISPLAY.....	76
<b>7</b>	<b>Photometric and Radiometric Analysis .....</b>	<b>77</b>
7.1	Simulator description .....	77
7.1.1	PostProcessing.py .....	78
7.1.2	Absolute Physical parameters calculation.....	79
7.2	Blender and Post Processing Results.....	80
7.2.1	Results: Bad Lighting Conditions.....	81
7.2.2	Results: Good Lighting Condition.....	83
7.3	Case Study: Cargo Bay .....	86
<b>8</b>	<b>Pangu.....</b>	<b>89</b>
8.1	Scene creation .....	89
8.1.1	Planet Model.....	90
8.1.2	Space Rider Model .....	94
8.1.3	Atmosphere Model .....	95
8.1.3.1	Atmosphere Parameters.....	96

8.1.4	Camera Model.....	99
8.2	Issues: Atmosphere .....	100
8.2.1	Issue Solution .....	101
8.3	Results.....	104
8.3.1	Pixel Flux Map: PANGU.....	106
9	Blender&PANGU: Comparison .....	108
10	Conclusion .....	109
11	References.....	110

## Figure Index

Figure 1: CFD simulation example (Credit (3)) .....	16
Figure 2: Structural simulation example (Credit (4)) .....	17
Figure 3: Thermal simulation example (Credit (5)) .....	17
Figure 4: Blender Logo.....	21
Figure 5: PANGU - Planet simulation (Credit (10)).....	23
Figure 6 Light rays bounces (Credit (13)).....	26
Figure 7: Light Strength variation .....	29
Figure 8: ATV render with 60° Sun angle .....	31
Figure 9: ATV render with correct Sun angle .....	31
Figure 10 Earth-Sun light system (Credit (17)).....	32
Figure 11 Sun Solid angle effect (Credit (18)) .....	33
Figure 12: Plank's radiation law Graph (Credit: Wikipedia) .....	34
Figure 13 Blackbody emission: temperature variation (Credit (7)).....	35
Figure 14 Light temperature effect on ATV Module.....	36
Figure 15: XYZ color space (CIE 1931).....	37
Figure 16: ATV Module with different exposure .....	38
Figure 17: ATV module in False Color Rendering.....	39
Figure 18: Filmic Top view (EV -8.17).....	42
Figure 19: False color top view (EV -8.17) .....	42
Figure 20: Filmic Bottom view (EV -8.17).....	43
Figure 21: False color Bottom view (EV -8.17) .....	43
Figure 22: False color Side View (EV -8.17).....	44
Figure 23: Filmic Side View (EV -8.17) .....	44
Figure 24: False color Top View (EV -9.68) .....	45
Figure 25: Filmic Top View (EV -9.68) .....	45
Figure 26: Filmic Bottom View (EV -9.68) .....	46
Figure 27: False color Bottom View (EV -9.68) .....	46
Figure 28: False Color Side View (EV -9.68) .....	47
Figure 29: Filmic Side View (EV -9.68) .....	47
Figure 30: False Color Bottom View (EV -3.58).....	48
Figure 31: Filmic Bottom View (EV -3.58).....	48
Figure 32: False color Top View (EV -3.58) .....	49
Figure 33: Filmic Top View (EV -3.58) .....	49
Figure 34: Filmic Side View (EV -3.58) .....	50
Figure 35: False Color Side View (EV -3.58) .....	50
Figure 36: Filmic Top view (EV -6.5).....	51
Figure 37: False Color Top View (EV -6.5).....	51
Figure 38: Filmic Bottom View (EV -6.5).....	52
Figure 39: False Color Bottom View (EV -6.5).....	52
Figure 40: False Color Side View (EV -6.5) .....	53
Figure 41: Filmic Side View (EV -6.5) .....	53
Figure 42: Reflective plane: angle variation .....	54
Figure 43: SpaceRider & Mirror .....	55
Figure 44: Filmic Top View (EV -11.98).....	56
Figure 45: Falsecolor Top View (EV -11.98) .....	56
Figure 46: Filmic Side View (EV -11.98) .....	57
Figure 47: False Color View (EV -11.98).....	57
Figure 48: Exposure value Chart (Credit (22)).....	59

Figure 49: Correct exposure rendering.....	60
Figure 50: CIE 1931 xy chromaticity diagram and sRGB color space (Credit: Wikipedia).....	63
Figure 51: Scene workflow .....	65
Figure 52: Photodiode model (Credit (27)).....	71
Figure 53: pixel photodiodes distribution.....	72
Figure 54: Pixel Fill Factor (Credit (28)) .....	75
Figure 55: Simulator Workflow.....	77
Figure 56: Post Processing Workflow .....	79
Figure 57: SR Eclipse.....	81
Figure 58: SR Eclipse Flux heatmap .....	82
Figure 59: SR Too Bright .....	82
Figure 60: SR too bright heatmap.....	83
Figure 61: SpaceRider .....	84
Figure 62: SpaceRider heatmap.....	84
Figure 63: Space Rider over Earth .....	85
Figure 64: SpaceRider over Earth heatmap .....	86
Figure 65: SR Cargo Bay.....	87
Figure 66: Cargo Bay – Flux .....	88
Figure 67: Cargo Bay - Number of Photons .....	88
Figure 68: PANGU simulator workflow .....	90
Figure 69: PANGU Earth Model .....	91
Figure 70: Earth dark face and Sun Model .....	92
Figure 71: Mediterranean Sea .....	93
Figure 72: Earth mesh under Mediterranean Sea .....	93
Figure 73: SpaceRider model in PANGU .....	94
Figure 74: SpaceRider View over Earth .....	95
Figure 75: Earth Atmosphere effect .....	96
Figure 76: Atmosphere - Sky color variation .....	98
Figure 77: Atmosphere: Sunset over the sea.....	99
Figure 78: Atmosphere issue #1 .....	100
Figure 79: Atmosphere issue #2 .....	101
Figure 80: SpaceRider and Earth – Wrong Atmosphere .....	102
Figure 81: Atmosphere issue solution .....	103
Figure 82: SpaceRider View - 20° .....	105
Figure 83: SpaceRider View - 28° .....	105
Figure 84: SpaceRider View - 90°.....	106
Figure 85: PANGU - Pixel Flux.....	106





## 1 Abstract

Illumination analysis tool for preliminary assessment of proximity operations.

Interest in graphical simulations to support proximity operations is increasing along with the number of private and public missions involving robotic spacecraft operating close to each other for commercial or scientific (and military) purposes.

Using the open-source computer graphics software Blender, a simulation environment was developed to perform a preliminary assessment of lighting conditions for proximity operations in Low Earth Orbit. The main purpose of the simulator is to guide mission requirements in terms of trim, position and payload performance. The instrument created was used to assess the observation and docking conditions of an observe-and-dock mission of a CubeSat near its mothership. An instrument was sought that accurately simulated the transport of light because the factor of view of the Earth from a spacecraft in low orbit is almost 50%. This makes diffused and reflected light a non-negligible component of scene lighting and under these conditions, indirect lighting (if neglected) could lead to incorrectly marking valid observation states as invalid.

Central to our tool's realism is Blender's Cycles rendering engine, which is a physically based path tracing engine. Starting from a simulation in Blender previously developed; containing an accurate model of the Earth, in which realistic atmospheric dispersion and decay, a layer of clouds, 3D topological characteristics and different reflective characteristics of water and land mass were implemented; the radiative environment of space around the Earth was initially developed as physically accurately as possible. Sunlight is simulated as a uniform vector field with energy flow  $1367 \text{ [W/m}^2\text{]}$  and the spectral characteristics of a black body  $5777 \text{ [}^\circ\text{K]}$  that can be oriented depending on the position of the Sun.

The initial prototype focuses on the visual spectral light captured by RGB sensors. The numerical limits of Blender that usually arise when placing objects with scale differences in the many orders of magnitude next to each other have been circumvented, hitting the target observed in the origin of the global frame and moving all other objects around it. An ideal optical sensor model is developed for the evaluation of lighting conditions, to which the various signal disturbances due to geometry or physical phenomena of the system are subsequently added. In order to correctly evaluate the performance of a payload, the rendering outputs are post-processed to extract radiance information that is subsequently fed to a monochrome sensor simulator as output to account for aperture and exposure time.

To fill the gaps in terms of physical accuracy of Blender, as it was born as a graphic rendering program, without taking care of the scientific analysis part in depth, a second simulator was created using the PANGU program developed by ESA. PANGU allows users to graphically implement the model, going to evaluate in a physically accurate way the quantities of interest for a lighting analysis, paying particular attention to the numerical aspect of the results as well as to an excellent graphic display.

At the end of the work, a comparison was made between the two simulators to understand their differences and strengths.

The graphics simulator can help high-level decision-making for orbit, trim, and payload selection, but it lacks the performance and real-time effects needed for GN&C software-in-the-loop simulation.



## 2 State of Art

In this first section we provide a general view of the existing programs and technologies to perform simulations and graphic analysis, paying particular attention and interest to the simulations of spatial scenes.

The world of computer graphics is very wide and varied and over the years numerous programs and tools have been developed that allow you to perform graphic simulations of all kinds.

### 2.1 Computer Graphics

According to (1), computer graphics is a discipline that deals with generating images with the help of computers. Today, computer graphics is a basic technology in digital photography, movies, video games, mobile phone and computer displays, and many specialized applications.

It was born in the 60s as described in (2) and over time has become increasingly interesting and important in the development of film productions. Computer graphics are often abbreviated with the acronym CG, or more commonly in the film world with CGI (Computer Graphics Imagery).

Over the years it has evolved extremely, passing from the first one- or two-dimensional simulations able to create a few pixels or very small static images, to being able to create complex three-dimensional scenarios completely animated and decidedly realistic.

In recent decades computer graphics have been increasingly integrated into cinematography, creating incredible special effects, or coexisting real characters played by actors with creatures or subjects made and animated entirely with the help of the computer.

Computer graphics must be subject to strict rules: it is necessary to understand in detail the principles of geometry, optics and physics in order to create accurate and plausible simulations compared to the real world. Of great importance is also the understanding of the perception of the images by the observer, in order to create scenes not only physically accurate, but also of excellent understanding by the viewer.

Computer graphics in recent years has taken place in numerous areas outside the mere creation of scenes for cinematographic use; all non-artistic aspects of computer graphics are identified in a branch of science called Computer Science.

Starting from computer graphics, to achieve better and more accurate results, numerous categories of scientific interest have been born and evolved such as the study of the science of colors, the study of lighting effects or the study of geometries.

Nowadays there are many different computer graphics programs, each with its own characteristics, with its own pros and cons. It goes from programs in which it is only possible to modify 2D images by applying filters or modifying shapes to real 3D simulators capable of creating realistic scenes.

The rapid evolution of computer graphics goes hand in hand with the improvements obtained from the hardware and software point of view of computers, in fact it is necessary an enormous computing power to be able to withstand the creation and animation of extremely complex models; just think of the creation and use of any video game of the current times. Of great importance is also the possibility of using increasingly high-performance displays, a greater resolution and a wider color range allow you to create increasingly detailed and realistic scenes.

From the scientific point of view, graphic analysis is increasingly used to be able to somehow predict the behavior of physical parameters in each condition, perhaps replicable in reality in the future or difficult to reach physically. Consider, for example, in the architectural and engineering field, the possibility of creating 3D models of bridges or buildings and evaluating a priori the behavior of the

objects in question when subjected to particular atmospheric conditions or structural stresses. An in-depth analysis therefore allows you to make important design decisions such as the materials to be used or the technologies to be applied.

As for the work under consideration, it is necessary to create a 3D spatial scene and perform a light-based analysis, to evaluate the physical conditions that occur among the spacecraft present in the simulated scene.

The following paragraphs describe in more detail the principles of computer graphics and the analyses related to computer science.

### 2.1.1 Computer Graphics Principles

Before starting to detail the potential and applicability of computer graphics, it is necessary to understand appropriately the principles and concepts that govern it. It is therefore of fundamental importance to understand how a real scene, captured by means of optical sensors, is transformed and represented to be displayed on a screen.

In computer graphics, raster images are usually used; a raster graphic represents a two-dimensional image such as a rectangular matrix or grid of square pixels, viewable using a computer display, paper, or other display. A raster is technically characterized by the width and height of the image in pixels and the number of bits per pixel. Raster images are stored in image files with variable diffusion, production, generation, and capture formats.

The main concepts of computer graphics are described below:

- **PIXEL:** A pixel is the small particle of a digital image and it represents the unit of graphics. Pixels are placed on a two-dimensional grid and are often represented using dots or squares. Each pixel is a sample of an original image, where multiple samples typically provide a more accurate representation of the original. Each pixel is defined by its intensity and its colour which is characterized by has three components such as red, green, and blue (RGB pixels).
- **PRIMITIVES:** Primitives describe relations between pixels and they are basic units which a graphics system may combine to create more complex images or models.
- **RENDERING:** Rendering is the process that allows, using a special software, to create digital images and videos of a scene. It is divided into 2D or 3D rendering depending on what you want to simulate. The digital image is simulated starting from a series of information, described in the simulation of the scene within the software such as: the description of the three-dimensional objects, the point of view on the scene, the lighting, the use of colors, the relationships between the different objects, the shadows and the simulated materials. Rendering is often used to create digital video, video games, special effects for film and television, and in engineering and architecture to create digital models of buildings or urbanized areas. It therefore allows, starting from a code, to create physically and visually accurate simulations of real-world scenes.

### 2.1.2 3D Graphic Design

The entire project is based on the creation and use of a space scene in three dimensions, it is therefore necessary to understand in detail the operation of a 3D graphic simulation environment. In reference of (1), at the base of a simulated scene are allocated the individual 3D models of the different parts in play; they are usually simulated and realized using complex geometries, often with the help of external Computer-Aided Design (CAD) programs, able to provide the correct geometries in the form of networks of points connected to each other by segments, forming a detailed mesh

that represents as accurately as possible the model to be realized. CAD is often implemented directly inside graphical design software so the whole process is more accessible and easier.

Once the 3D models have been created or imported, it is necessary to arrange them appropriately within the simulation space, in order to create the basis for describing the relationships between the different objects.

For an accurate 3D simulation, many aspects such as shading or light propagation must also be taken into account. It is therefore necessary to evaluate and create shadow models able to give realism and truthfulness to the simulated scene, going to distinguish very light areas from dark areas.

As for the propagation of light and consequently the visualization of the scene, the ray-tracing technique is usually used, which allows to simulate the path of light, generated and reflected, between the various objects of a scene through the application of mathematical algorithms, going to transfer the 3D scene on a two-dimensional plane. Over the years, increasingly complex algorithms have developed, which have made it possible to achieve excellent photorealism with the only cons of dilating computational rendering time.

Another important step for the realization of a scene is represented by the application of materials and surface finishes to the various models to add realism to the scene rendering. Usually within computer graphics programs it is possible to create and apply to the surfaces and volumes of materials, characterized by numerous properties such as color, density, opacity and reflectivity, in such a way as to simulate real materials in the best possible way. They are applied to the surfaces of the polygons that make up the 3D models through the texture mapping technique, which allows you to match each individual surface to particular characteristics.

The choice and application of the correct materials to the 3D models is fundamental to be able to properly simulate the behavior of the objects to the simulated light beams; it is also possible to carry out scientific analysis on the models, going to evaluate characteristics and behaviors due to external stresses, which are closely linked to the properties of the materials.

### **2.1.3 Animation**

Once the necessary 3D models and the environment of the graphic simulation of the scene, including colors, surface finishes and materials, have been completed, it is necessary to uniquely define the relationships between the different objects present. We then move on to the animation phase, often implemented within the same computer graphics program, which by mathematical laws allows to simulate the movement and inter-relationships between the different bodies. The underlying goal of the animation part is to simulate real-world movements as accurately as possible. At the software level, the individual polygons that make up the models are considered and mathematical laws are applied that can modify their position and direction in space. Of fundamental importance is the time factor, it is in fact necessary to impose a time interval that elapses between two positions in space assumed by the object that must be animated.

A static representation of a scene, as if it were a photograph, is called a frame. As a result, a moving scene is characterized by numerous frames arranged in rapid succession in such a way as to create an idea of fluid and continuous movement. As a result, the quality of a video is closely linked to the number of frames per second (fps) that are simulated by the graphics engine.

As far as graphic design software is concerned, it is usually possible to define an initial position, a final position and a path that a single body must follow, then setting the frame (or image) of the beginning and the frame of the end, in such a way as to consistently simulate the movement.

In particular, the known points of animations are called start-frames and end-frames in graphic programs and can be used not only to move and animate objects, but also to insert special effects in postprocessing, such as lighting effects or the presence of moving smoke.

The animation of a scene is therefore rather simple at a conceptual level, but greatly expands the working time due to the huge number of mathematical calculations necessary to obtain the desired result.

### ***2.1.3.1 Animation: Computational Cost***

The amount of time necessary to render a video of an animated scene is strictly related to the concept of computational cost.

The computational cost represents the number of operations and calculations that the computer processor must perform to complete a user-imposed command. In the case of the animation of a simulated scene with a computer graphics software, the command is represented by bringing the desired object from one spatial position to another. Since the object is simulated with a geometric model whose polygons are divided into meshes composed of a very large number of points, the processor must calculate the variation in position of each individual point for each frame that will make up the final result. As a result, a very high number of calculations is reached, to which must be added all the effects related to the external environment, the variation of lighting and the relationships with the remaining part of the scene. It is therefore clear the need to use high-performance processors and graphics cards, which are able to carry out numerous calculations and display the desired result in a relatively short time.

### **2.1.4 Computer Science**

Everything that does not have a strictly artistic purpose in the generation of an image or video using a computer graphics program, is grouped within computer science, a branch of science that studies different phenomena and areas of interest by exploiting the capabilities of a computer.

In particular, with regard to graphics, it is possible to simulate real-world scenes and obtain predictions on the behavior of objects created under certain conditions. Computer science is divided into numerous disciplines and touches on numerous areas, it is possible for example to graphically evaluate the behavior of a certain material to external stresses or to evaluate the lighting conditions of a body in a certain position in space. It is also very effective to combine strictly numerical analysis with graphic simulations, to make the understanding of the physical phenomena that characterize the analysis itself more immediate. The next chapter provides an overview of the use of graphical simulations in engineering.

## ***2.2 Graphical simulation for engineering***

In the engineering field, it increasingly makes use of the graphical interface for simulations or analyzes of any kind, as it simplifies the understanding of the analyzes themselves and provides an alternative point of view for the interpretation of the results. Since the dawn of computer graphics, we have tried to integrate it for physical evaluations on the different phenomena that need to be studied.

Initially it was exclusively made use of two-dimensional images, in such a way as to simulate maps or simple behaviors of phenomena, with the passing of the years it is increasingly in common use a

CAD type software able to accurately simulate the geometries and shapes of any object, eliminating the need to create objects to be able to carry out analysis of any kind.

The advantages of using graphic simulations in engineering are therefore many:

1. **Economical:** with the possibility of creating models, it is obsolete to create a large amount of different test samples, being able to carry out experimental tests directly on a graphics engine. This allows to reduce production costs and the quantity of material used.
2. **Qualitative:** the use of a graphical interface combined with simulations and engineering analysis allows to obtain more precise results and allows to have a direct correlation of the phenomena analyzed to the real world. This results in more reliable analyses, which reduce the margin of error in every area.
3. **Time:** the analysis and completion times of an entire process are greatly reduced as it is possible to carry out a huge number of operations and analyzes in a short period of time, exploiting the computing power of the computers and graphics software used.

Another aspect to be underestimated is related to the presentation of an engineering project, it is in fact possible to create graphic simulations that can be used as images or advertising videos, in order to present a potential customer with a detailed and attractive solution. Regarding works of a public or administrative nature, this is fundamental, as through well-structured advertising it is possible to obtain good feedback from public opinion.

In the following sections some examples of graphical simulation for engineering are presented.

### 1. Computational Fluid Dynamics (CFD)

Computational fluid dynamics (CFD), as described in (3) allows you to use Navier Stokes equations, which govern fluid movement for a wide range of complex situations, providing both detailed information and quantitative predictions. Fluid equations are replaced by discrete approximations at grid points that must be close enough so that the solution is independent of the spacing of the grid points. Discrete equations are derived using finite differences or finite volumes, connecting the different points of the grid to each other.

Computers are used to perform the calculations necessary to simulate the free flow of fluid and the interaction of fluid (liquids and gases) with surfaces defined by boundary conditions. In this way it is possible to calculate the turbulent or linear nature of the flow lines and highlight possible critical points or dangerous vortices around the analyzed structure.

CFD is applied to a wide range of research and engineering issues in many fields of study and industries, including aerodynamics and aerospace analysis, hypersonic, weather simulation, natural sciences and environmental engineering, industrial systems design and analysis, biological engineering, fluid flows and heat transfer, engine and combustion analysis, and visual effects for films and games.

The CFD programs were created starting from situations analyzed in real equipment such as wind tunnels, it was therefore possible to create programs that carried out a large number of calculations able to simulate the fluid dynamic conditions to be analyzed. It is therefore clear that the use of CFD has made it possible to significantly streamline the processes for the analysis to be carried out on, for example, an aircraft, greatly reducing the physical tests to be carried out with scale models before the final product is put on the market.

Below is an example of a CFD analysis with chart support, in which the flows that invest the object in question in the simulation conditions are graphically highlighted.

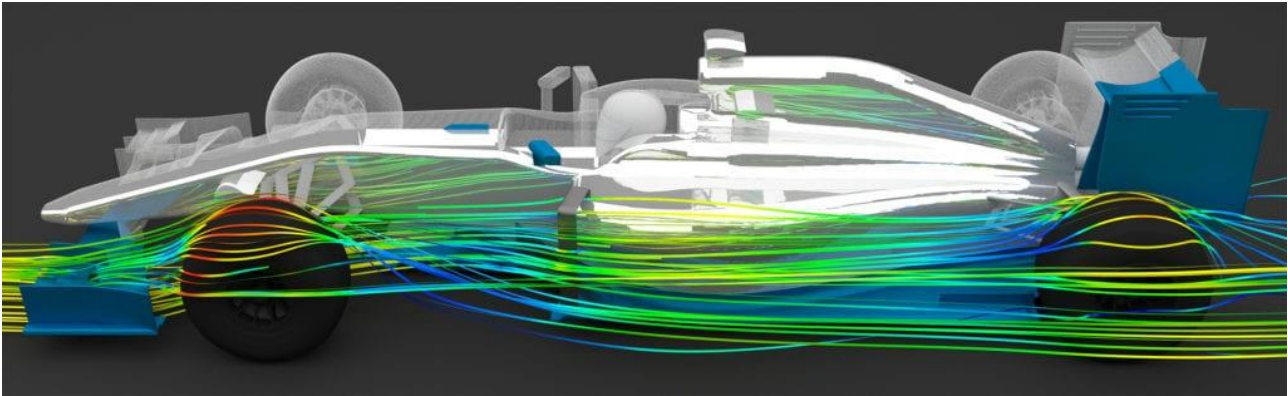


Figure 1: CFD simulation example (Credit (3))

Therefore, CFD exploits the computer's computing power for the numerical evaluation of aerodynamic phenomena, integrating it with graphic visualization using 3D models and color maps, to simplify the understanding of phenomena.

In the last century this discipline has developed a lot and many specific programs have been created that can each perform different types of analysis; many of the CFD programs are still free and accessible even to students or beginners.

## 2. Structural Analysis

In a very similar way to fluid dynamics analysis, structural analysis can be treated, which represents the determination of the effects of loads on physical structures and their components. Structures subject to this type of analysis include everything that must withstand loads, such as buildings, bridges, planes and ships. Computer graphics are now commonly integrated with software dedicated to structural analysis as it allows, using 3D models of the structures to be analyzed, to visualize graphically and give a more concrete idea of what are the values of tensions, deformations and loads of the structures themselves. It is therefore easier to understand the behavior of the models at the stresses and thanks to the possibility of physically accurately inserting materials, dimension and characteristics of the objects a simulation completely consistent with reality is obtained. Below is an example of a structural analysis in which the deformations of a structure are graphically displayed, described in (4), in which finite element analysis is performed.



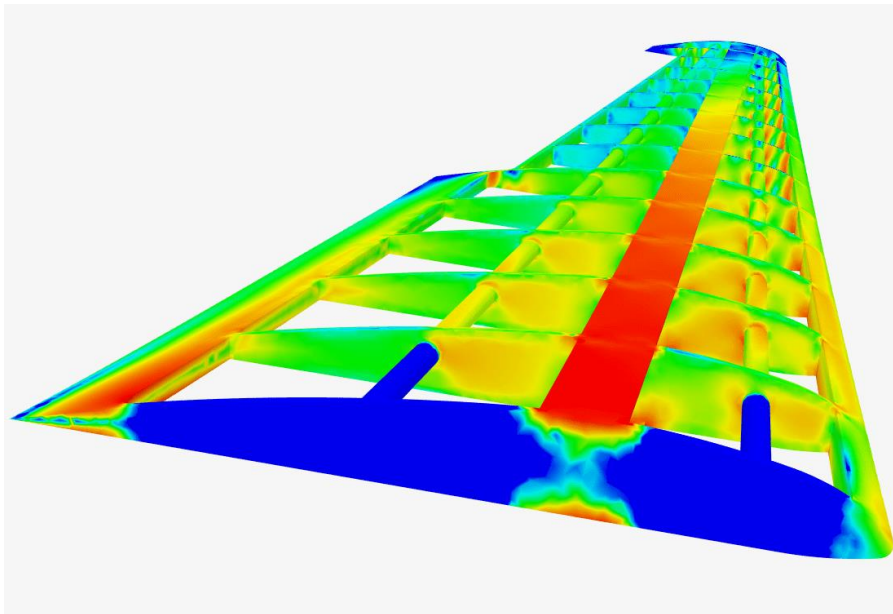


Figure 2: Structural simulation example (Credit (4))

Static or dynamic analyses can be carried out, depending on what you want to analyze. Over the years, structural analysis using software has developed a lot, resulting in a wide choice of specific programs to choose from.

### 3. Thermal analysis

Like the previous cases, computer graphics can be integrated with programs that use mathematical models to evaluate the thermal analysis of an object or system. Thermal Analysis is a branch of materials science that studies how changes in temperature affect a material's properties. Using 3D models, it is therefore possible to make an analysis of the geometries, going to evaluate the thermal flow values at each individual point of the mesh that makes up the model.

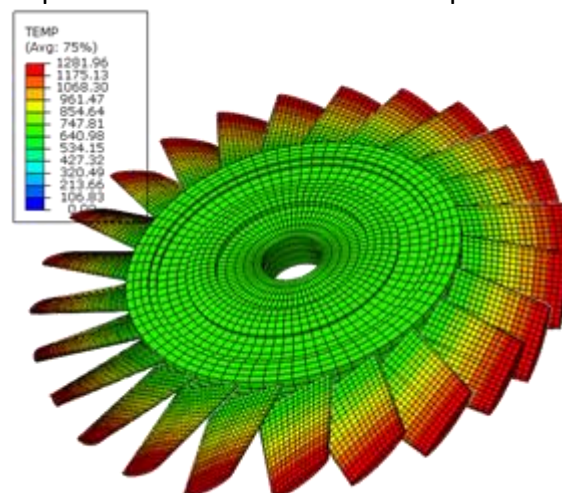


Figure 3: Thermal simulation example (Credit (5))

In the image above is presented the result of a thermal analysis of a turbine, graphically you can see how a color scale is used to represent the different temperature values on the body of the object

under examination. Also, in this area there are numerous programs to carry out different analyzes and usually there are tools within structural analysis programs that can also carry out thermal analysis.

#### 4. Radiometric and Photometric Analysis

Of greater interest for the project is the radiometric and photometric analysis of a scene. It can be carried out using computer graphics by creating truthful simulations of real-world scenes. In this way it is possible to obtain a physically accurate model, from which the desired information can be obtained.

In the following chapters the possibilities of computer graphics programs will be analyzed in detail, paying particular attention to the implementation of the light model and optical sensors that point at the scene created.

### 2.3 3D graphic design programs


In order to obtain 3D graphic simulations and consequently to be able to carry out analyzes it is necessary to use programs capable of creating and visualizing three-dimensional simulations of objects and scenes. On the market there are many programs of this type that are used in different types of different industries to obtain the desired results.






As mentioned above, computer graphics are used in areas such as cinema, video game production, medicine, architecture and engineering and consequently the needs of different sectors differ greatly. The great variety of programs, with different characteristics, strengths and weaknesses, allows each sector to make the most of the possibilities of this discipline.

The biggest difference between the different 3D graphics programs is given by their access; in fact, they are divided into two main categories: paid programs and open access programs. First ones are usually more performing and offer unique features to the user, allowing to perform very specific tasks with high precision and reliability, combining acceptable rendering times. Open access programs, on the other hand, are more generic and represent a solid base from which to enter the world of computer graphics; in recent years they have developed to such an extent, especially thanks to the contribution of users, that they have become the main programs used by some companies, as in the case of Blender, the software used for this project and described in detail later. Below is a list of the best computer graphics programs currently available on the market, both paid and free, with their main features. Only programs available in both Windows and Linux will be considered, according to (6).

#### 2.3.1 3D graphic design: paid-for programs





The following table describes the best paid 3D graphics programs available now.

Name	Logo	Description	Cost
Autodesk MAYA		Probably the best program available, from beginners to masters. Unrivalled set of features and tools. Very complete. Strengths: <ul style="list-style-type: none"> <li>• Incredibly powerful</li> <li>• Comprehensive toolset</li> </ul>	4590\$/year

<b>ZBrush</b>		<p>Incredibly powerful in creating bodies and organic sculptures, but difficult to learn. Able to use complex geometries.</p> <p>Strengths:</p> <ul style="list-style-type: none"> <li>• Incredibly sculpting</li> <li>• Handles millions of polys</li> <li>• Fast</li> </ul>	895\$
<b>Houdini</b>		<p>It uses a nodal approach to simulations, provides flexibility and control, and allows you to create incredible special effects (VFX).</p> <p>Strengths:</p> <ul style="list-style-type: none"> <li>• Powerful</li> <li>• Free version</li> <li>• Incredible VFX simulation</li> </ul>	4495\$/year
<b>Cinema 4D</b>		<p>It is recognized for its stability and for being the 3D modelling software with the easiest learning curve. Has a huge online library of tutorials.</p> <p>Strengths:</p> <ul style="list-style-type: none"> <li>• Huge amount of plugins</li> <li>• Easy to learn and accessible</li> </ul>	3495\$
<b>Modo</b>		<p>Very user-friendly, it is the best option to work with polygons and to create models.</p> <p>Strengths:</p> <ul style="list-style-type: none"> <li>• Incredible modeller</li> <li>• Easy workflow</li> </ul>	1596\$
<b>Lightwave 3D</b>		<p>Two apps in one. Modeller to create models and geometries, Layout to animate and rendering.</p> <p>Strengths:</p> <ul style="list-style-type: none"> <li>• Flexible</li> <li>• Easy to manage</li> <li>• Separate focus with two apps</li> </ul>	995\$

### 2.3.2 3D graphic design: free programs

The following table describes the best free 3D graphic programs available now. Blender is the chosen program for the project; it will be described deeply in the following section.

Name	Logo	Description
Blender		Best choice possible. Huge amount of addons and incredible precision in texturing, animating and rendering. Always in development. Strengths: <ul style="list-style-type: none"> <li>• Open Source</li> <li>• Powerful</li> <li>• Large use</li> <li>• Easy to learn</li> </ul>
Daz Studio		3D figure customisation, posing and animation tool that enables artists of all skill levels to create digital art. Strengths: <ul style="list-style-type: none"> <li>• Large amount of models</li> <li>• Accurate rendering</li> </ul>
SketchUp		Simplest 3D modeler available. Runnable on the web, with a related storage. Only usable for simple models. Strengths: <ul style="list-style-type: none"> <li>• Very easy to use</li> <li>• Models and textures available</li> </ul>
Hexagon		Under Daz Studio, it is focused on the creation of complex models, with a huge number of polygons and nodes. Strengths: <ul style="list-style-type: none"> <li>• Complex models</li> <li>• Works with other apps</li> </ul>
Fusion 360		Under the Autodesk family, it is able to simply create 3D models of any part, it needs a render to obtain images. Strengths: <ul style="list-style-type: none"> <li>• Autodesk family</li> <li>• Easy to use</li> </ul>
Wings 3D		Great choice for polygon models. It has a weird interface with respect of other programs, but it is very user friendly.

Most of the free software (except for Blender and Daz Studio) does not have an incorporate render, but they must be associated with an outside one.

## 2.4 Blender

Blender (7) is an open-source, free and cross-platform software that is used for modeling, animation, compositing and rendering of three-dimensional images; its history began in 1995 at the Dutch animation studio NeoGeo, and its release under the GNU GPL' license took place in 2002. To date, Blender boasts a very large community and can provide excellent support through the numerous forums dedicated to it and extensive documentation (written entirely by users). It is a very flexible program that can be easily understood and used by those approaching the world of 3D modeling for the first time, but which offers complex tools for the most experienced. It can represent different physical phenomena, including the gaseous and liquid states of matter and gravity. Blender offers geometric primitives to model the different objects of the scene, along with manipulation and editing tools that allow you to create any shape you want. It is also possible to add and work with curves. Within the scene it is essential to have one or more light sources, without which it would not be possible to render anything, and a camera that provides the framing for our rendering. It is good to emphasize that Blender remains a tool, and despite the wide variety of tools that provides a true masterpiece can only be obtained through practice and study, as well as the knowledge of subjects such as anatomy, the theory of lighting and colors, the principles of animation and in general every artistic, technological and scientific knowledge that is behind it that we want to represent.

Due to its open-source nature and its accessibility, Blender is one of the most used graphic design programs; moreover, Blender is constantly evolving and is constantly being improved by developers and its users, so that it has become increasingly competitive and reliable.

Blender boasts an impressive set of 3D modeling and sculpting tools and is nowadays considered a completely viable alternative to paid modeling programs. It is also becoming increasingly prevalent in the pipelines of major graphics studios, also because in the past Blender one was known for its non-standard way of working, but in recent years many of these problems have been solved and it has conformed to the most common graphic design programs, even if paid.



**Figure 4: Blender Logo**

One of the most useful tools of Blender is its python interface, which allows you to integrate the writing of a code with the possibilities given by a graphics engine.

Blender has a built-in Python interpreter that loads when Blender starts and remains active while Blender is running. This interpreter executes scripts to draw the user interface and is also used for some of Blender's internal tools.

Blender's built-in interpreter provides a typical Python environment, so tutorial code on how to write Python scripts can also be run with blender's interpreter. Blender provides its Python modules,

such as *bpy* and *mathutils*, to the embedded interpreter so that they can be imported into a script and give access to Blender's data, classes, and functions. Through the use of its specific modules, it is possible to perform any function allowed by Blender, it is therefore very convenient to write a script and launch it, going to modify small parts at each execution, rather than having to promptly modify parameters or simulated objects in the scene passing directly through the graphical interface.

The project was carried out entirely using python scripts, so that they were easily accessible to other users for possible future implementations. It was also decided to give continuity to the programming language used, going to realize in python also the scripts used outside Blender.

## 2.5 PANGU

For computer graphics engineering purposes, related to the world of aerospace, in recent years a software called PANGU (8) has been developed by the European Space Agency (ESA). The project was born in the Space Technology Centre of the University of Dundee.

The Planet and Asteroid Natural Scene Generation Utility (PANGU) is a set of software tools that provide a complete sensor modeling and simulation environment to support the development and testing of vision-based guidance systems as described in (9).

It includes a suite of 3D modeling functions to produce realistic planetary surfaces and asteroids, complete with morphological features such as craters and boulders, without neglecting the real effects determined by the materials that make up space objects.

PANGU also provides a radiometrically accurate environment, as light and scatter patterns were developed so that they were physically accurate in every simulable situation. This allows, through the implementation of an optical sensor with its parameters, to obtain results faithful to reality, thus being able to carry out analysis starting directly from the output of the software.

It also provides a visualization tool that can be used to generate an image from a simulated camera, RADAR and LIDAR measurements from any point of view, allowing PANGU to be used in closed-loop engineering simulations of complete CNG systems.

Recently, PANGU's modelling and rendering functions have been improved to include man-made materials and complete spacecraft models. This includes the ability and ability to import spacecraft models from CAD files, render highly non-Lambertian surfaces such as solar panels and OSR panels and include articulated joints and independent moving parts that can be dynamically configured.

These new tools are intended for use in dynamic scenarios, such as flight simulations or complete simulations of space missions in Earth orbit in which phases of rendezvous and docking in orbit appointment and docking in orbit are planned.

PANGU is therefore a program created ad hoc for graphic simulations of a spatial nature, thus resulting in the perfect software for radiometric analysis of spatial scenes. The software, however, is not open access but a license issued by ESA is required to use it.

Like most computer graphics programs, it has an interface from which you can change the parameters of the simulation, but most of the work is done using command line scripts or taking advantage of the different interfaces that allow you to use code written in different programming languages, such as C, C++, Java and Matlab. You can also leverage the TCP/IP interface to implement simulations.

You can get both images and videos of spatial scenes from the program and being the software designed for environments of this type, the rendering and waiting times are much shorter than generic programs such as Blender. Below is an example of Mars soil simulated in PANGU.

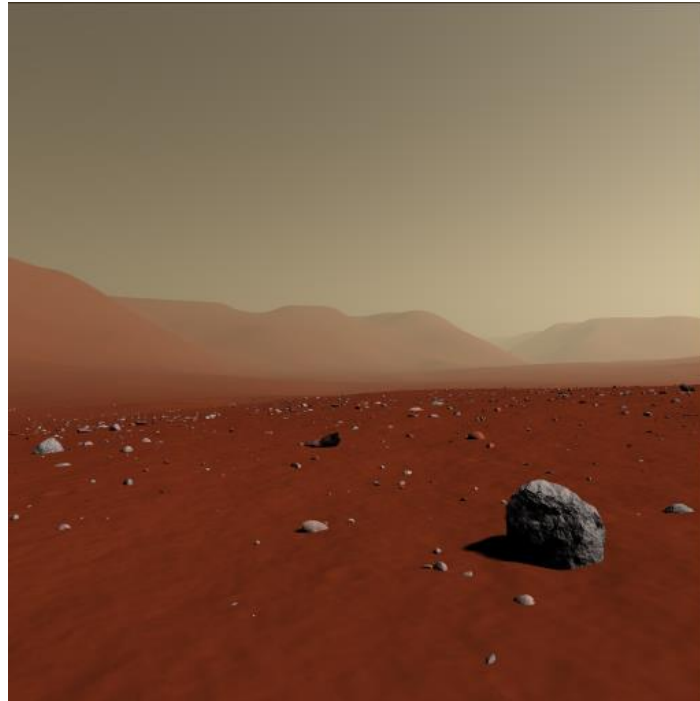


Figure 5: PANGU - Planet simulation (Credit (10))

The software is also suitable for the implementation of atmospheric models and evaluation of the presence of the atmosphere itself on a scene, as characteristic and physically accurate parameters can be easily implemented.

The software is therefore a good computer graphics solution when used in the field for which it is designed. The creation and development of a model in PANGU will be treated in detail in the dedicated section, also going to analyze and evaluate the results thus obtained.

## 2.6 Space Scene Simulation

As described in the abstract, the objective of the project under consideration is the study of the lighting conditions of a satellite orbiting in LEO (Low Earth Orbit). It is therefore necessary to create the graphical simulation environment to be able to carry out the analysis, which corresponds to a spatial scene. In order to simulate a 3D environment of a spatial scene accurately and realistically, certain conditions and accuracy must be met. In fact, the preparation of the simulation environment is fundamental, to be able to simulate all the physical phenomena of the space environment. We must also pay attention to the type of space scene desired, in fact there are significant differences between a planetary system and a satellite in low Earth orbit in terms of environmental conditions and behavior of the bodies.

Usually, a spatial scene is composed of the following objects:

1. **Celestial Bodies:** Like planets or asteroids, they are characterized by large dimensions. They are usually approximated with spherical shapes for simplicity, but the choice of surface finishes and materials is fundamental, in such a way as to give a good realism to the object. Very often the choice of the characteristics of the materials is complex especially regarding asteroids, as in the real world they have behaviors when hit by light rays.
2. **Orbiting bodies:** Spacecraft and satellites, bodies much smaller in size than the bodies previously described. Characterized by models usually very detailed and detailed. Also, in this case the choice of materials and surface finishes is fundamental, as the behavior of the spacecraft in the simulation environment must be simulated in the best possible way. Think, for example, of the enormous difference between the material of which a solar panel is composed, always present on the structure, and the material of which the main body of the satellite is composed. The difficulty of implementation is therefore linked to the large number of pieces and details from which they are composed, in addition to the enormous difference in size compared to the main bodies.
3. **Lighting Model:** Describes the presence of light on the scene. The model must be physically accurate, truthfully representing the lighting conditions that would be found in nature. It is defined by characteristic parameters that allow to obtain a result very close to reality.
4. **Atmospheric model:** Used near planets with the presence of atmosphere, it describes the environmental conditions in which a spacecraft is immersed, going to define the changes in density and temperature and influencing the animations both from the point of view of movement and from the point of view of lighting.
5. **Orbits:** describe the movement of simulated bodies. They are used for dynamic scenes and represent the paths that the different objects must follow, they are usually referred to a central body.

By correctly implementing these elements it is therefore possible to recreate a scene that represents the space, which can be used for advertising purposes of a mission or to carry out radiometric analysis, as in the present case.

The main problem is that usually computer graphics programs are not designed to simulate scenes of this type and there are great difficulties of rendering and functionality when you go to approach objects with extremely different dimensions, or perhaps at very large distances. To overcome this problem, the scale factor of the scene is exploited, going to modify and reduce the real size of all the objects present, so that the computer at the base of the program can better manage the interactions between the bodies and the graphic simulations. The following chapters describe the process for implementing the light model of a spatial scene.

## 2.7 Render Engines

To create and display a simulation environment on any computer graphics software, a renderer must be used. The different renders differ from each other in the graphic features they offer.

Rendering is the process behind a graphic simulation and consists of transforming a 3D scene into a 2D image. Most computer graphics programs have made renderings that can create images. PANGU has a physically accurate internal one that reflects its characteristics, while Blender, also thanks to its multifunctionality and adaptability, has several, depending on the characteristics you are looking for; it is also possible to implement others through tools or addons.



Blender (7) includes three main rendering engines with different strengths:

- Eevee is a physically based real-time renderer.
- Cycles is a physics-based light path tracker.
- Workbench is designed for layout, modeling, and previews.

The choice among different renderers from those mentioned above is determined by the wanted output characteristics of the rendering and using the software. These renderers allow, for example, to obtain a different diffusion of light, using different equations or to analyze a scene from a spectral point of view, as in the case of the Mitsuba2 render. In the following sections we will describe them in more detail, but they are rarely used due to the complexity of installation and the large time it takes to render images.

Rendering is essential because it allows users to obtain a result, usable as output, which represents the three-dimensional scene created. The render works by frame, then freezes a possible moving scene, obtaining the image relative to the single moment in time. Consequently, to obtain the video representation of a moving scene, the render will create a series of frames related to successive temporal moments and then put them in sequence in such a way as to give an idea of movement and animation to the scene.

In photography, the quality of a video is defined by a parameter called *fps* (frames per second), which represents the number of frames that are used to create every second of a video; the greater the number of *fps* and the better the quality and fluidity of the video, but negatively affecting the time it takes for the computer to produce the desired result.

Following (11), the basic objective of the render is therefore to represent the simulated scene, starting from the description of the energy flows, in the form of light, on the scene itself. In detail, it is a matter of solving an equation that allows to define the path and the amount of light radiation at each point of the image, consequently going to obtain the values for each individual unit of the image, called pixels. For the equation to be physically accurate and plausible, it must be subject to geometric and mathematical laws. In 1986 James Kajiya defined the rendering equation in (12), to which everything refers, which says: "In a certain position and direction, the outgoing light ( $L_o$ ) corresponds to the sum of the emitted light ( $L_e$ ) and the reflected light. Reflected light, in turn, is the sum of incoming light ( $L_i$ ) from all directions, multiplied by the reflective surface and angle of incidence. "

This definition represents an integral equation, which is linear and homogeneous in every point and direction of space.

A lighting model is then defined that allows to simulate the behavior and interactions of light rays with the objects of the scene. Currently the most used and most reliable is the path ray tracing model, which through mathematical models describes in a physically accurate way the path that a light beam follows it has been emitted by the light source. It will be explained in more detail for the individual renders, but Blender allows you to calculate the path of light up to over a thousand bounces after emission.

### 2.7.1 Render Engine: Cycles

The chosen render engine for the project is Cycles, described in (7). It was chosen because the render is already integrated in Blender and it uses real-world, physically based lighting and materials. As logic Cycles uses a path tracing integrator with direct light sampling where the rays are traced from the camera into the scene for each pixel in the output image and the rays continue until they find a light source.

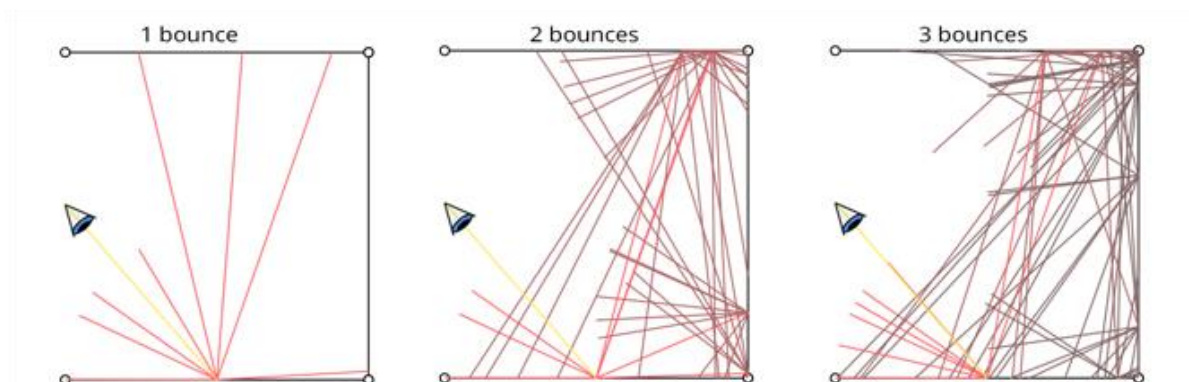


Figure 6 Light rays bounces (Credit (13))

Several parameters can be chosen in settings such as number of bounces permitted to light rays or parameters related to the background scene, to recreate the more realistic scene possible.

As we can see in Figure 6, the same light source can recreate scenarios with completely different lighting depending on the number of bounces allowed to its light rays, the greater they will be and the better illuminated the simulation environment will be. In particular, the render Cycles uses, for reasons of memory use, the backward path tracing technique that is opposed to the more expensive technique of forward path tracing, which consists in tracing the path of each individual photon that makes up the light rays, even those that at low energy or after a significant number of bounces no longer bring an obvious contribution to the illumination of the scene.

Backward ray tracing works in reverse; an eye ray is created to the eye that passes through the visual plane and continues into the world. The first object hit by the eye beam is the object that will be visible from that point of the visual plane. After the ray tracer has allowed that beam of light to bounce, it calculates the exact coloring and shading of that point in the visual plane and displays it on the corresponding pixel on the computer monitor screen. In this way, importance is given only to photons that effectively reach the optical sensor, drastically reducing the number of calculations necessary for rendering. Backward ray tracing is also known as eye ray tracing.

There are several types of lights sources available in Cycles and these will be described in further detail below. The render is characterized by discrete waiting times, as a lot of work is done of overlapping the data before returning the required output; multiple samples are required for each rendered image and selecting an appropriate sample size will result in a convergence and a representative image of the overall illumination of the scene.

Using Cycles different light sources can be modeled, in our case study the “Sun lamp” has been used to simulate the Sun, parameters are defined in following chapters.

Regarding the output image, Cycles can render to a high dynamic range image format (Radiance HDR, Open EXR, etc.), where every single pixel is encoded with real world luminance values and thus the HDR image includes the entire dynamic range of the scene. The dynamic range is the ratio between the maximum and the minimum of the tonal value in an image, so the greater it is, the better. In fact, an HDR image will allow us to obtain information on a real-world scene, including the brightest and the darkest parts, that in a normal image format would result as total white or total black. Studying the characteristics of the output images is possible to obtain the photometric and radiometric values of the scene as seen through a space camera.

### 2.7.2 Render Engine: Eevee

Eevee is Blender's real-time rendering engine built using OpenGL focused on speed and interactivity achieving the goal of making materials PBR. Eevee can be used interactively in the 3D Viewport but also produces high-quality final renderings.

Materials with the Eevee engine are created using the same shader nodes as Cycles, making it easy to render existing scenes. For Cycles users, this makes Eevee perfect for previewing materials in real time.

Unlike Cycles, Eevee is not a raytracing rendering engine. Instead of calculating each beam of light, Eevee uses a process called rasterization. Rasterization estimates how light interacts with objects and materials using numerous algorithms. Although Eevee is designed to use principles based on real physics, it is not perfect and Cycles will always provide more physically accurate renderings.

The rasterization process therefore determines major limitations in Eevee and for this reason it cannot be used in the current project; in addition to the already described physical inaccuracy in light tracking, Eevee produces renderings with particular illuminations. In addition, some functions present in Cycles for the implementation of physical or luminous models are not present with this graphics engine. In the specific case of the creation of the light source of the scene, with Eevee it will not be possible to correlate the values with real-world quantities and it will not be possible to give a physically accurate color to the light rays, given the absence of the node blackbody shader.

Eevee is therefore a good render in terms of applicability and production speed, but unfortunately completely inadequate for the purposes of the project.

### 2.7.3 Render Engine: Mitsuba2

Mitsuba 2 is a search-oriented rendering system written in portable C++17, according to (14). It consists of a small set of basic libraries and a wide variety of plug-ins that implement features ranging from materials and light sources to full rendering algorithms. Mitsuba 2 strives to maintain scene compatibility with its predecessor Mitsuba 0.6. However, in many other respects, it is a completely new system that follows a different set of goals.

The most significant change in Mitsuba 2 is that it is a retargetable renderer: this means that the underlying implementations and data structures are specified in a generic way that can be transformed to perform several different tasks.

The main advantage of this render engine is the fact that Mitsuba can be used as a monochrome renderer, an RGB-based renderer, or as a spectral renderer. Each variant can optionally consider the effects of polarization if desired. This allows you to visualize and analyze a scene from a completely different point of view, to get more information.

In addition, Mitsuba is a physically accurate render and simplifies the work of users thanks to a python interface, from which you can program and impose features and commands on the graphics engine.

Due to the difficulty of installing the render and its difficult use it was not taken into account to be used in the project; moreover, being combined with Blender it is difficult to bring it back to an accurate radiometric model, with the risk of obtaining unusable results for the purposes of analysis.

### 3 Radiometrically accurate Model

The aim of this part of the project is to create a radiometrically accurate model of the light conditions related to a space environment in the proximity of an object (Space Rider) in low Earth orbit. For the implementation of the model, Blender was used, an open-source computer graphics software, with which it was possible to create a simulation of a spatial environment near the Earth with its boundary conditions. In previous works the model of the Earth, the atmosphere, Space Rider and the SROC CubeSat have already been implemented, starting from the work contained in (15). Below are reported the fundamental steps of the development of the radiometric model, starting from an overview among the physical quantities that will be involved in the analysis.

#### 3.1 Radiometry vs. Photometry

Before starting with the definition of the model, it is important to understand which physical quantities must be used. According to (16), light can be measured in different ways and two different branches of science study it:

- Radiometry: the study of the electromagnetic radiation, it represents the physic of the light.
- Photometry: the study of light in terms of its perceived brightness to the human eye.

Each light is defined by radiometric and the correlated photometric parameters; they are defined as follow:

- Radiometric Parameters:
  - Radiant flux (or power),  $\Phi_e$ , expressed in Watts (W), is the radiant energy emitted, reflected, transmitted, or received, per unit time.
  - Radiant intensity,  $I_e$ , expressed in Watts per steradian (W/sr), is the radiant flux emitted, reflected, transmitted, or received, per unit solid angle.
  - Radiance,  $L_e$ , expressed in Watts per steradian per square meter (W/sr/m<sup>2</sup>), is the radiant flux emitted, reflected, transmitted, or received by a given surface, per unit solid angle per unit projected area.
  - Irradiance (or radiant flux density),  $E_e$ , expressed in Watts per square meter (Watt/m), is the radiant flux received by a surface per unit area
- Photometric Parameters:
  - Luminous flux (or power),  $\Phi_v$ , expressed in lumen (lm), is the measure of the perceived power of light and is a measure of the total amount of light a lamp emits.
  - Luminous intensity,  $I_v$ , expressed in candela (cd), is the luminous flux emitted by a light source in a particular direction per unit solid angle and is a measure of how bright the light is in a particular direction.
  - Luminance,  $L_v$ , expressed in candela per square meter (cd/m<sup>2</sup>), is the luminous flux per unit solid angle per unit projected source area, and is an indicator of how “bright” an illuminated surface or light source will appear to an observer.
  - Illuminance (or luminous flux density),  $E_v$ , expressed in lumens per square meter (lm/m<sup>2</sup> or lux), is the luminous flux incident on a surface, and is an indicator of how much the incident light will illuminate the surface.

As we can notice, there is a strong connection between the two branches of study; the radiometric parameters will represent the input for the description of the light, the photometric instead will be the output from the rendered image.

### 3.2 Creation of light source

Due to the enormous distance between bodies in the Solar System it is difficult to create the Sun cannot as a “Point Light” in Blender using python code, but it is simpler to use the “Sun Light” type of light implemented in the program. A “Sun Light” is characterized by various parameters such as strength, angle, color, and direction; in our project they are defined in a Python script and imported in Blender thanks to its Python API. Generally, the “Sun” in Blender is a uniform field of light which invest the whole scene from a defined direction. Below the choice of parameters is described, to obtain a radiometrically accurate model of the Sun. Attempts and preliminary evaluations were made on the 3D model of the ATV module for simplicity of calculation and rendering.

#### 3.2.1 Parameters: Strength

The parameter “Strength” in Blender is defined as the “Sunlight strength” and it is measured in Watt per squared meters [ $W/m^2$ ]; then it represented a physically accurate measure of the light. In fact, we can measure the average intensity of the Sun seen from Earth. The solar flux represents the amount of energy, measured in Watt, emitted by Sun in the outer space as an electromagnetic radiation, basically divided in visible light and infrared. This flux travels through the space since it reaches the Earth, then the average Solar flux density that hits our planet can be calculated. We can talk about “average” because the intensity is not constant during the motion of revolution of the Earth around the Sun, in fact, the Earth's orbit is elliptical and therefore the distance of the planet from the Sun varies depending on the time of year. In this part of the project, we can consider the solar flux density (or Irradiance if we would use radiometric quantities) as a constant value and it is:

$$E_{SUN} = 1367 \left[ \frac{W}{m^2} \right]$$

This value represents the whole quantity of energy received by the Earth. It will be reduced and modified by the atmosphere model implemented in previous parts of this work to create a totally physically accurate model of intensity of the light source.

Future work can be based on understanding the spectral composition of Sunlight, defining the amount of visible, infrared and UV fluxes from the Sun.

It is important to set the correct value of Irradiance to be able to create a physically accurate ambient of work with proper light and illumination conditions.

This parameter is defined in the script python *sun\_final.py* as Irradiance (in a section below the whole code is described).

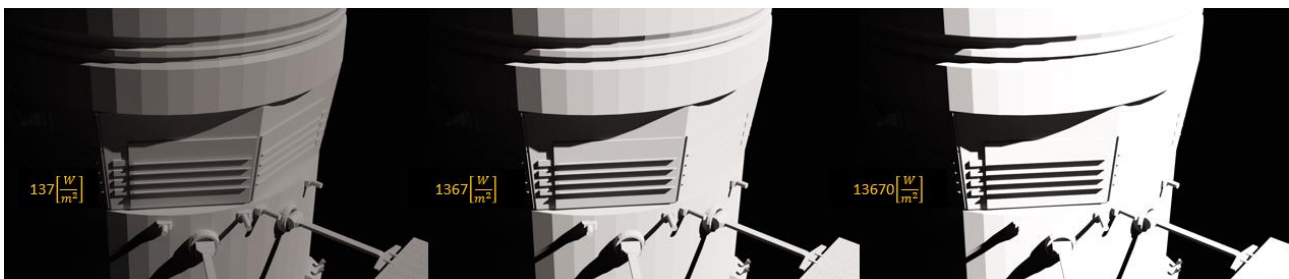


Figure 7: Light Strength variation

### 3.2.1.1 Irradiance and Illuminance

Starting from the value of solar flux density, a radiometric parameter, we can correlate a photometric quantity. The power received by a surface per unit area measured in Watt per squared meters can be converted in illuminance, a parameter that describes the amount of light perceived by human eye per area unit. It is measured in Lux or lumens per squared meters, and it can be calculated with the following conversion:

$$1 \left[ \frac{W}{m^2} \right] = 683 \left[ \frac{lumens}{m^2} \right]$$

This conversion factor is only an estimation because it is referred to only a single average wavelength of the electromagnetic spectrum. Light in fact is defined by wavelength and this calculation is related to the average wavelength of the visual spectrum of light, equal to 555 [nm]. Since the human eye can see only a short range of wavelength, this conversion could be acceptably accurate.

A wrong value of irradiance will give a scene not physically accurate, and some problems related to brightness of the whole rendering.

### 3.2.2 Solid Angle

The parameter “Angle” in Blender is measured in radians and it represents the angular diameter of the Sun as seen from Earth. From this input value the solid angle of the Sun in steradians can be calculated and then values of quantities such as Radiance or Luminance can be obtained.

The Sun has an average diameter of:

$$D = 1.39 * 10^9 [m]$$

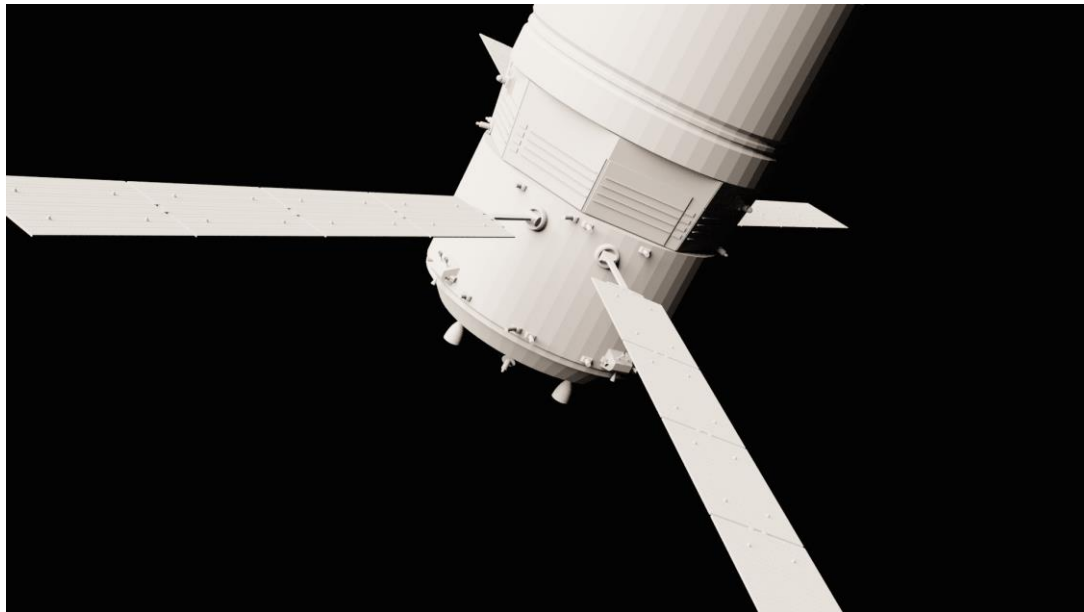
that at a distance from Earth of about:

$$distance = 1 [UA] = 1.5 * 10^{11} [m]$$

will give us an angle of:

$$angle = 0.526^\circ.$$

In Blender the angle determines the width of each single ray of the uniform field of light created to approximate the Sun. As we can see in the following pictures representing the ATV module from NASA, a wrong value of the angle will have incredible effects on the casting of shadows on the scene, with the result of a not physically true image as output. With a huge angle the scene will result over-lightened, and a lot of small particulars will be lost.



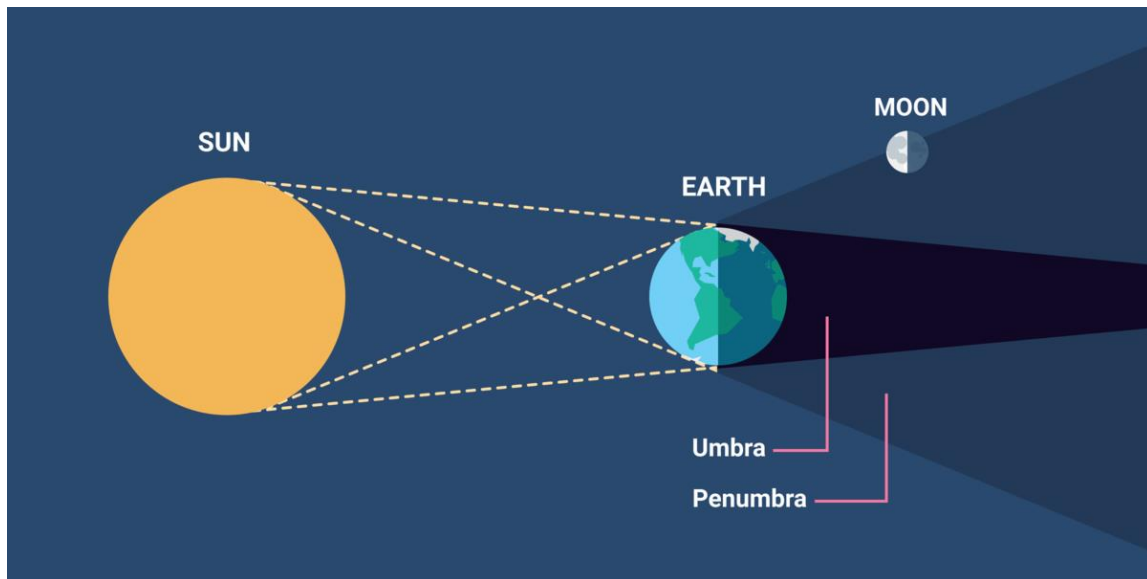
**Figure 8: ATV render with 60° Sun angle**



**Figure 9: ATV render with correct Sun angle**

The size of the solid angle of the Sun obviously implies the physical size of the Sun, even if it is not physically simulated in the program; a wrong value of this value therefore implies a star of incorrect size. Considering the Sun as a light source from which the light rays that hit the illuminated objects start, it is easy to understand that with a wrong size there are great effects regarding the implementation of shadows on surfaces. As can be seen from the previous images, in the case of a not physically accurate angle of the Sun, lighting conditions and shadows totally distant from the real ones are obtained; this problem transposed into a simulation of a scene with several bodies, as it will be in the final simulator, with the coexistence of Earth, atmosphere and satellites, will be

deleterious, as the interactions between lights and shadows generated by the different bodies are fundamental for a good physical accuracy of the entire simulation.



**Figure 10 Earth-Sun light system (Credit (17))**

In the image above the Earth-Sun system is schematized, the areas of total shadow and penumbra are clearly noticed and it is therefore easy to guess that a too large value of angle greatly expands the penumbra area to the detriment of the shadow, while vice versa a value too small goes to eliminate the penumbra, going to affect the entire scene. Possible situations are represented in the image below.



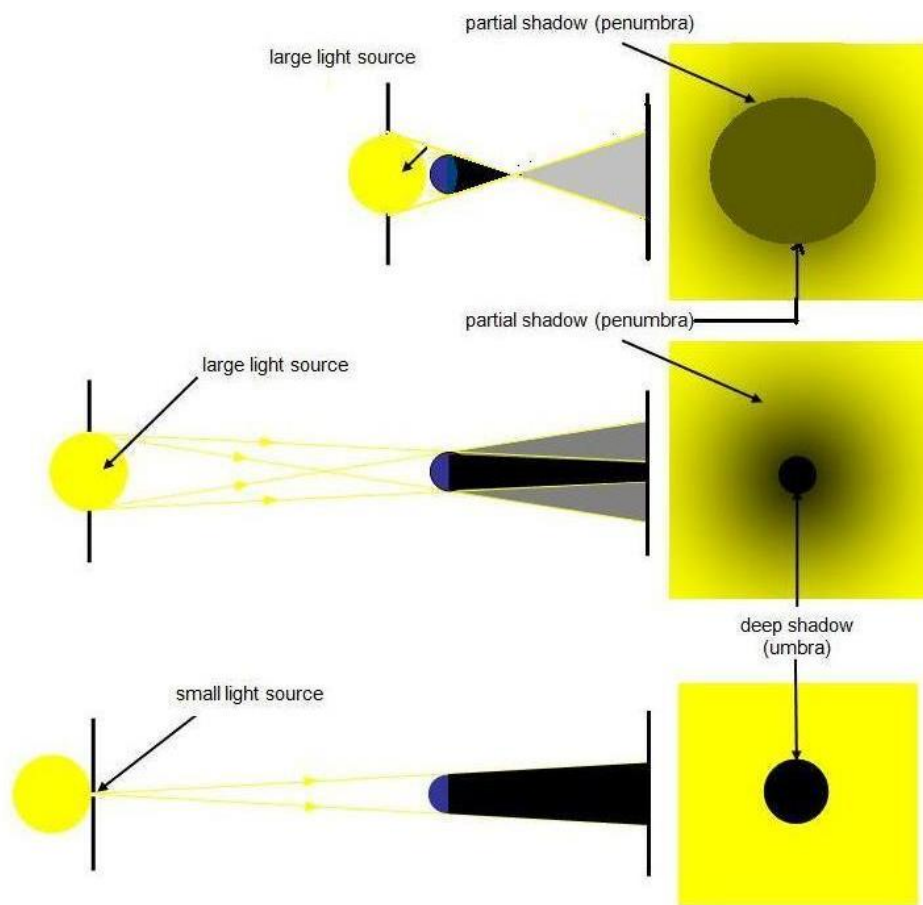


Figure 11 Sun Solid angle effect (Credit (18))

The physically correct value of the Sun angle must be scaled in order to uniform it to the scale of the whole scene; in fact, a correct value combined with objects with a different scale and other than the actual dimensions will cause the same effects of inaccuracy due to a totally wrong angle, with the result of a scene apparently extremely close to the Sun.

### 3.2.3 Rotation

With the parameter “Rotation”, an array with 3 components, the angular direction of the light rays is defined. It is related to the reciprocal position between the Earth and the Sun, which varies depending on the time of year in which it is in the simulation. For a short simulation, in the order of hours, the angular direction of the rays, divided into 3 axes, can be considered constant. In *sun\_final.py* the vector “rotation” contains the 3 Euler angle of light rays with respect with Earth expressed in radians.

The possibility to change the rotation vector of solar radiation at will allows you to simulate all possible lighting conditions in which a satellite in low Earth orbit can be found. Through a python code it is therefore possible to modify the position of the satellite around the earth's central body and the rotation of the sun's rays to simulate the lighting conditions, the periods of exposure and eclipse along the entire orbit of the space mission.

### 3.2.4 Color

Due to its physical characteristics such as dimensions, chemical composition and temperature, the Sun is classified as a "yellow dwarf" of spectral type G2 V, where "G2" indicates that the star has a surface temperature of 5777 [K]. The chemical-physical processes that take place inside the star generate a huge amount of energy that is emitted in space in the form of electromagnetic waves, which have frequencies mainly in the visible and infrared fields.

To evaluate the color of the visible radiation emitted by the Sun, and therefore to model the star physically accurately, it is necessary to make some approximations and considerations. The Sun is considered as a blackbody, that is, an object that absorbs all the electromagnetic radiation to which it is exposed, which is then re-emitted into the external environment.

The radiation emitted by a black body is called blackbody radiation and the radiated energy density blackbody spectrum. The spectrum (intensity or density of the radiation emitted as a function of the wavelength or frequency) of a black body is a spectrum with a characteristic bell shape dependent exclusively on its temperature  $T$  and not on the matter that composes it.

The correlation between temperature and color is determined by the Correlated Color Temperature.

It is possible to determine the distribution of radiation emitted by a black body at a given temperature using Planck's radiation law, from which the bell graph at a given temperature is obtained and the emission peak of the body is obtained, corresponding to the main color that will assume the electromagnetic radiation emitted.

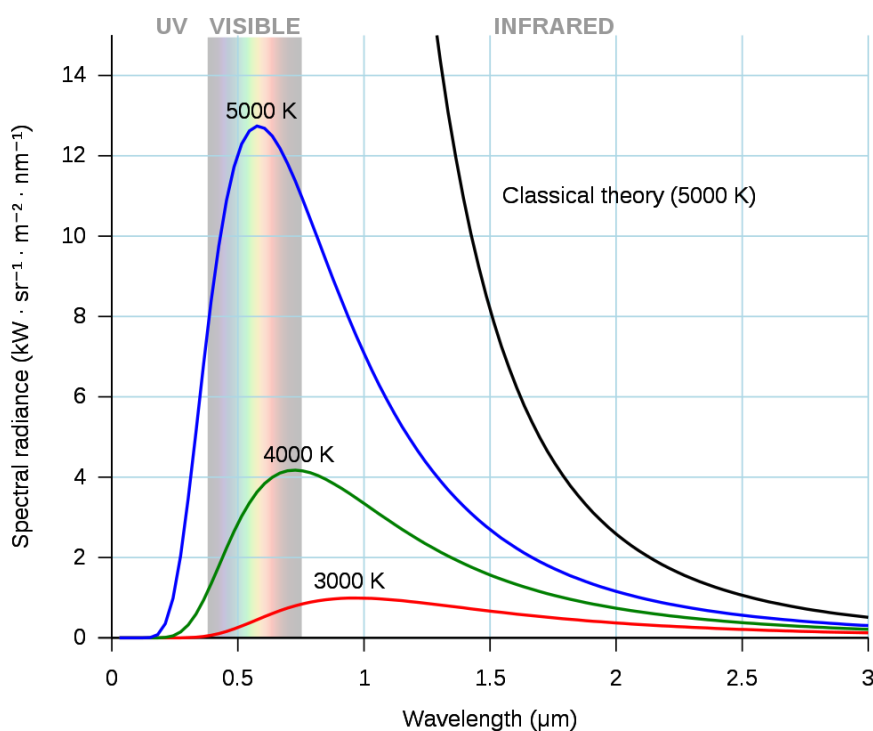


Figure 12: Planck's radiation law Graph (Credit: Wikipedia)

As we can see from the graph reported above, as the temperature increases, the peak moves to shorter wavelengths in the visible field, tending to ultraviolet.

It is therefore necessary to attribute to Sun model on Blender the right color, this is possible in several ways, which will be explained below.

### 3.2.4.1 BlackBody Shader Node

Using the Render Cycles, the color of the light field that represents a "Sun" model using a function in the Blender code called "BlackBody" can be chosen. In this way, by entering the surface temperature of the Sun as input, the program will return the corresponding color as output, assigning it to the light rays of the source.

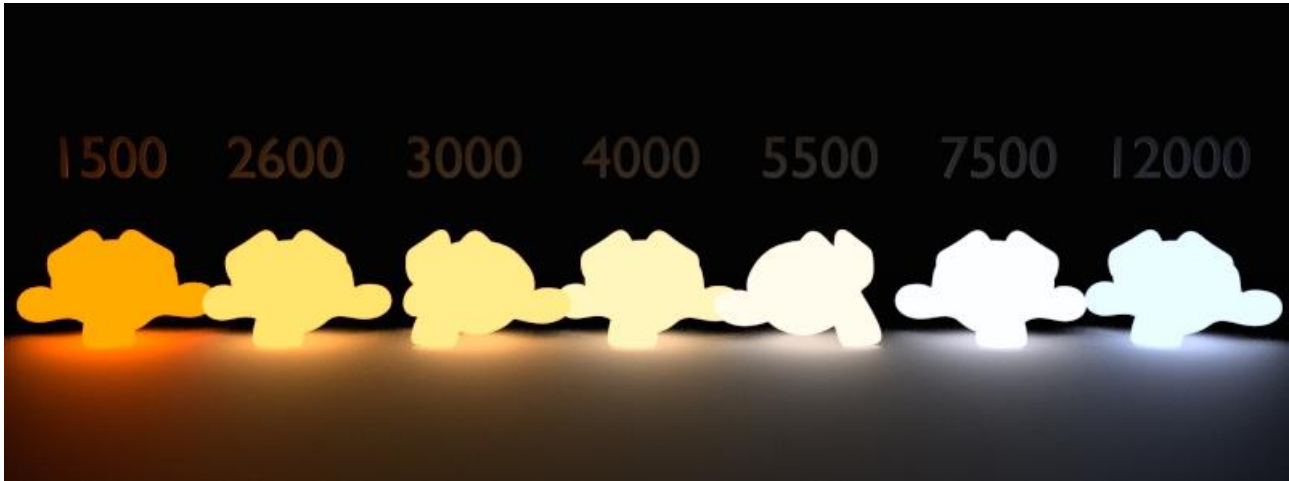


Figure 13 Blackbody emission: temperature variation (Credit (7))

Above there is a representation of the effect of temperature on blackbody shader node, applied to a light model implemented on the famous monkey-head of Blender. As we can notice, with a relatively low temperature the light will be warm, instead, increasing temperature will shift the light color to a cooler one.

In the following images an example of changing temperature of light incident on the ATV module is represented. The light is created with a Sun model using the Cycle Render. An "Emission" node is used in order to create a surface property in which the "BlackBody" shader node could be implemented, giving the light the correct color. The light has the following parameters:

- Strength: 1367 [W/m<sup>2</sup>]
- Angle: 0.526°
- Temperature: changing between 500°K to 12000°K

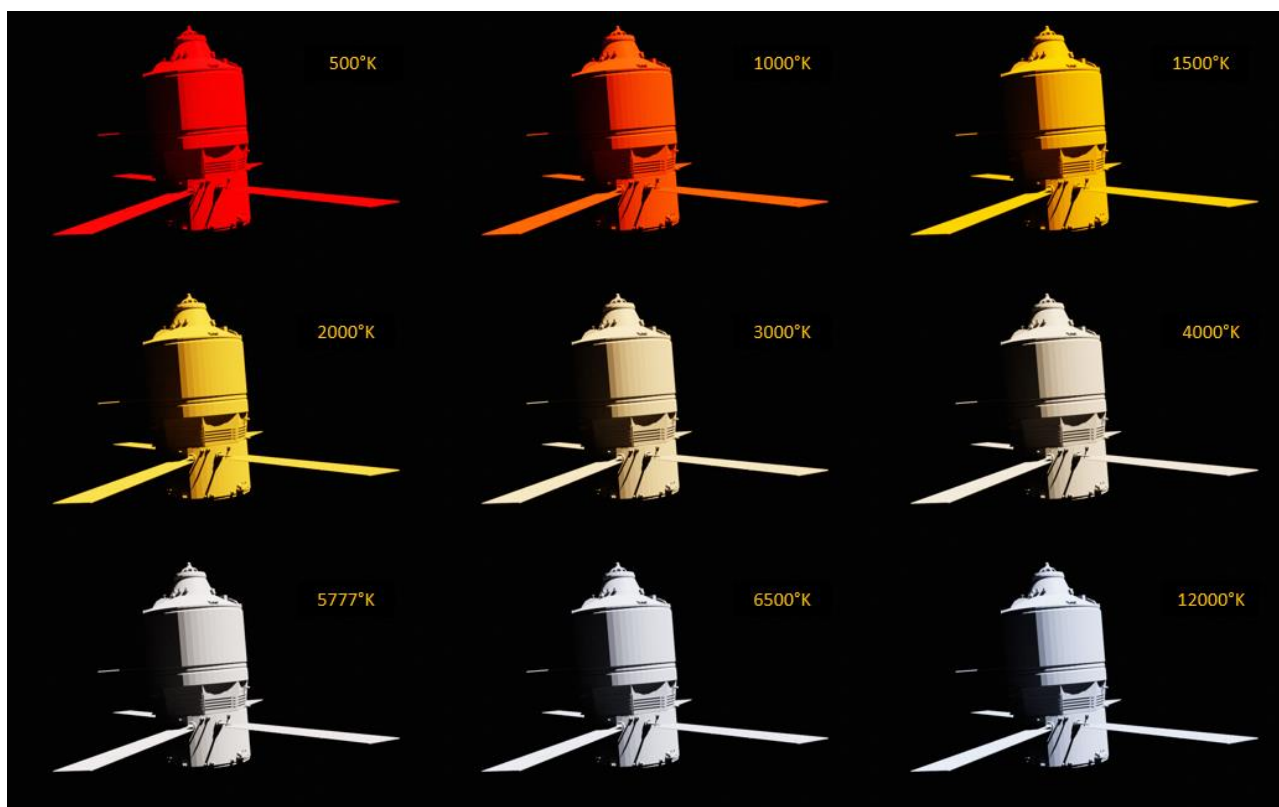


Figure 14 Light temperature effect on ATV Module

As expected, the result switches from warm colors to cold colors as the temperature increases, noting in particular how the temperature value of 6500 ° K is the total white color, as expected. As a result, the Sun, which has an average surface temperature of about 5777°K, produces light slightly warmer than white, as can be seen in the image at the bottom left.

A physically accurate color is obtained for the light rays of the light model. Given the simplicity of implementation and the accuracy of this method, it will then be chosen and used in the final simulator.

### 3.2.4.2 Python .colour library

Another way to determine the correct color for a light source at a given temperature is related to the use of the functions present within the Python ".colour" library. This library contains functions related to the science of colors and allows to transform a given color into the different encodings possible through mathematical methods. It also represents a verification for the BlackBody Shader Node, as the correspondence between the colors obtained with the two methods can be verified.

The Correlated Color Temperature (CCT) is defined as a the temperature of an ideal black-body radiator that radiates light of a color comparable to that of the light source. The radiation is in the visible field and the CCT is expressed in Kelvin (K). Blackbodies are divided into two categories of color emission depending on their temperature: bodies with a temperature above 5000 [K] generate "cool colors", below instead we speak of "warm colors". Since the Surface Temperature of the Sun is about 5777 [K], it will have a cold light, with a color between yellow and white.

To transform CCT into a color that Blender understands, following (19), we need to introduce the concept of color coding and color spaces. A color space represents a combination of a color model and a mapping function, where the color model represents a mathematical model that can uniquely

describe all visible colors using combinations of numbers. The most commonly known spaces are those based on RGB models, where with the use of 3 values on a scale of 8 bytes (256 bits) it is possible to define the percentages of Red, Green and Blue related to a specific color. Using mathematical functions and conversion models developed over the years it is possible to represent the same color in different color spaces.

The figure below shows the graph for the XYZ color space, developed in 1931 by the Commission Internationale de l'Eclairage (CIE) (20) in such a way that it included all color tints visible to the human eye, regardless of the luminance conditions of the color. In fact, adding luminance as a Z coordinate to the two-dimensional space XY results in the complete 3D model. The model called CIE 1931 is based on the use and mixing of 3 primary colors and represents the reference for all other color spaces due to its accuracy. The parameters of the color space are called tristimulus values and represent respectively: Y brightness, Z blue stimulation and X similar to the red sensitivity of the curve of the cones.

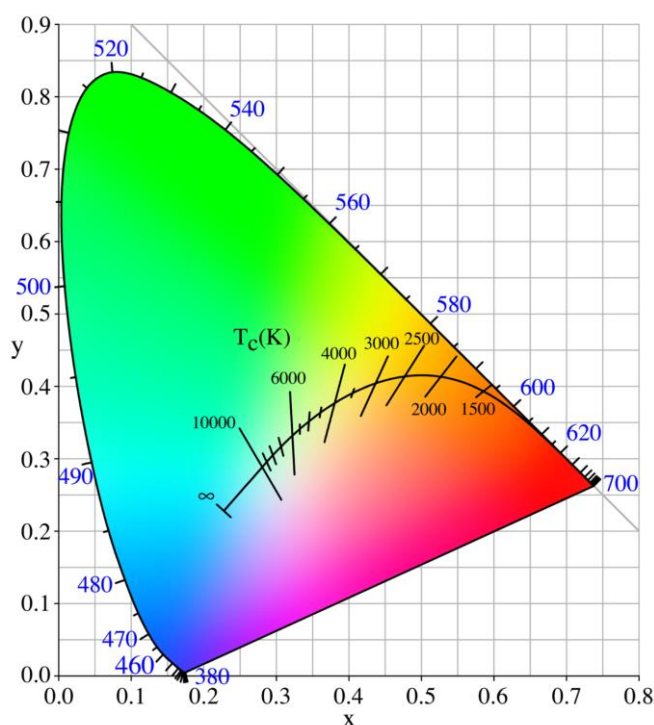


Figure 15: XYZ color space (CIE 1931)

In the graph above is represented the curve  $T_c(K)$  which represents the trend of emission colors of a black body as its temperature changes. Over the years, numerous models of conversion from CCT to more common color spaces have been developed; for the project was chosen the one developed by Hernandez in 1999, described in (21), which through a mathematical model allows to switch from CCT to XYZ encoding, which can then be used to switch to the standard RGB format and be implemented in Blender.

These methods are implemented in a version of the light model in Blender, in order to obtain the physically correct color for the Sunlight rays. The conversion follows the following scheme:

1. From CCT to XYZ color Space using the Hernandez 1999 method.
2. From XYZ to sRGB (standard RGB color space, not linear).
3. From sRGB to RGB using a linearization.

RGB color will be used as input for the "Sun" type light, and it will give the wanted color to light rays.

This solution, although it is in the first approximation, offers a lower precision than the implementation and use of the Blender Blackbody node, as there are problems of numerical approximation due to the different conversions necessary to move from one color space to another. This method is therefore a confirmation of the validity of the Blackbody node, since, comparing the results obtained previously with those expected from the temperature curve in the color space of the previous image, a complete and accurate correspondence is noted.

### 3.3 Scene Exposure

Once the light source of the model is created, so that it is radiometrically accurate, an additional step must be taken before rendering the scene. In fact, it is necessary to set a correct exposure value of the scene, so that it is as accurate as possible to reality. The exposure does not represent a radiometric quantity that must be imposed a priori, but it is a parameter that is necessary to obtain good results. For rendering spatial scenes, the exposure values recommended in the literature are between -7 and -5; for this phase of the project a value of -6.25 was therefore chosen. An incorrect exposure value will make the scene extremely illuminated or extremely dark. Using the Filmic Color Management in Blender it is therefore possible to solve this problem, without losing the physical and radiometric accuracy of the light source. In the images below, the same scene is represented, with the same light source, but with different exposure values.



Figure 16: ATV Module with different exposure

As we can see in the image on the left, an under-exposed rendering will give us a dark image, where colours and shadows will not seem real. On the other hand, a very high exposure, as the picture on the right, will give us an image where small details and shadows will be lost due to the incredible brightness of the rendering.

To evaluate exposure choice, the False Color Tool in Color Management could be used. With this tool the user can obtain a visual representation of the exposure of the scene using a scale of colours, where red will represent the major exposure and blue the darkest parts. The image below shows the rendering in False Color of the three scenes above.

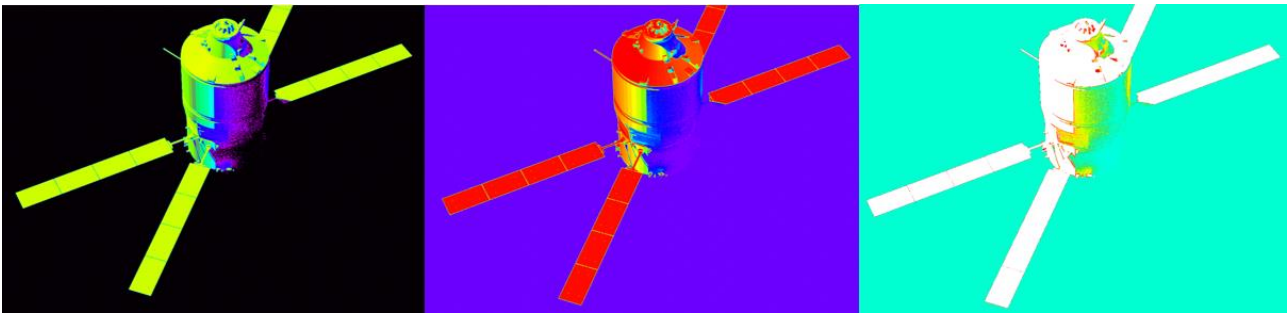


Figure 17: ATV module in False Color Rendering

### 3.4 *Light Model*

The light model is then implemented in Blender with the characteristics described above, so that it is physically accurate and traceable to a real situation. From a practical point of view, the model is created in Blender by means of a python script that is then run from the dedicated interface in the program itself.

The script contains in a dictionary variable all the information and parameters that uniquely define the light source, such as intensity, angle, direction, color and name. These parameters are then read by the script, which through Blender's *bpy* library imposes them on the light model that is created. The convenience of this method is related to the fact that it is enough to simply change the parameters contained in the dictionary type variable to change the lighting pattern.

## 4 Scene Calibration: Exposure

In this section we analyze the fundamental steps of calibrating the exposure levels of the scene created in Blender, to obtain data from the rendering as accurately as possible and without losing information due to the lighting conditions of the scene itself.

The evaluations were carried out on a test environment, in which only the Space Rider model in its latest version available at full size and sunlight, reproduced with maximum physical accuracy both in terms of intensity and color of light radiation, are implemented. The presence of the Earth and the atmosphere, present in the complete simulation, is therefore neglected.

Below are proposed different possibilities of values for the exposure of the scene, each with its own characteristics.

### 4.1 Calibration Method

To evaluate the lighting conditions of the scene due to the exposure, Blender's Color Management was used, which allows you to change the display parameters of the scene.

For the "real" visualization of the image, the View Filmic was used, which allows to:

1. To compress the scene referred linear radiometric energy values down to the display / output referred range. This aspect is known as a transfer function or tone mapping. The shape of the Filmic Base Log with a contrast aesthetic roughly emulates a photographic film curve.
2. To compress the gamut for high intensity values. As colour ratios increase in intensity, highly saturated ratios tend to be resistant to transfer function compression, which results in peculiar feeling imagery with some regions feeling appropriately over-exposed and others "lingering" behind. Filmic considers all colour values fair game and attempts to blend colours into a consistent output that matches our learned expectations from film emulsion-like media.

The image obtained in Filmic was then compared with the image obtained in the False Color view. False Color is a Tool useful to evaluate images in terms of the dynamic range and the latitude and it is basically a color coded heat map, where each colour is referred to a Linear Value and to an EV value. So, it is possible to obtain information about luminous conditions all over the rendered scene. The aim of the calibration is to completely avoid total white and total black zones on False Color image, because they indicate areas of the image that are outside the supported dynamic range and therefore correspond to areas from which it is not possible to obtain radiometric information. Black refers to areas that are too dark and total white to areas that are too light, seen as a single spot unique by the program.



False color legend (Credit (7)):

Value	Colour	Scene Referred Value
Low Clip	Black	Scene Referred Linear value below 0.0001762728758.
-7.5 EV	Purple	Scene Referred Linear value 0.00099436891.
-5.0 EV	Blue	Scene Referred Linear value 0.005625.
-2.5 EV	Cyan	Scene Linear value 0.005625.
-0.01 EV	Green-Cyan	Scene Referred Linear value ~0.178.
0 EV	Grey	Scene Referred Linear value 0.18009142.
+0.01 EV	Green-Yellow	Scene Referred Linear value ~0.181.
+2.17 EV	Yellow	Scene Referred Linear value 0.80779930598.
+4.33 EV	Red	Scene Referred Linear value 3.62857262286.
High Clip	White	Scene Referred Linear value above 16.29174024.

Below are the borderline cases of exposure, each with its pros and cons. An intermediate case is also presented, which can represent a trade-off between a very wide dynamic range and a plausible image. All simulations are addressed in two borderline cases:

1. In full illumination, with the sun's rays forming an angle of 90 degrees with SR.
2. In complete darkness, with the SR completely shielding the sun's rays and the camera placed on the eclipse side of the satellite.

For all the simulations, an average scene contrast was chosen, which can be modified a posteriori as needed.

#### 4.1.1 Totally Dark Sky

The scene represents the highest limit value that allows the user to obtain a completely dark sky, referred to light conditions.

Exposure: -8.17

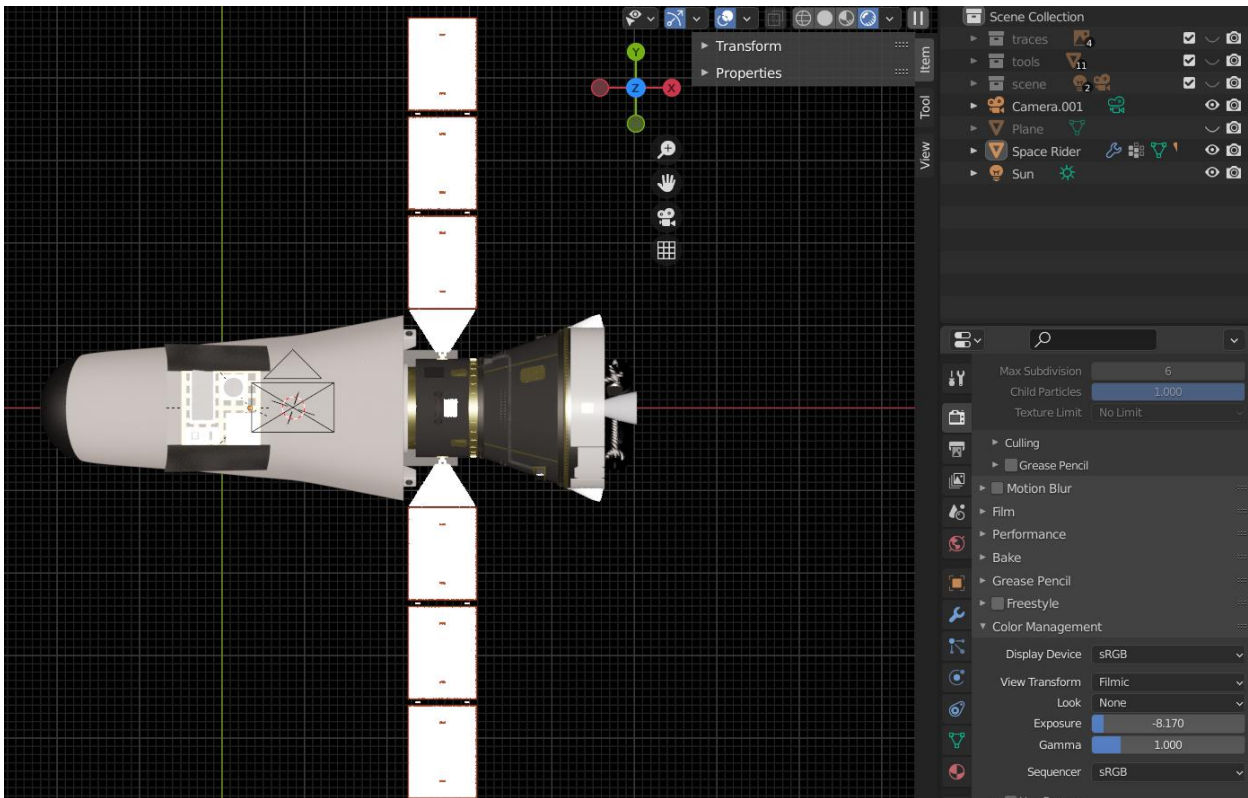


Figure 18: Filmic Top view (EV -8.17)

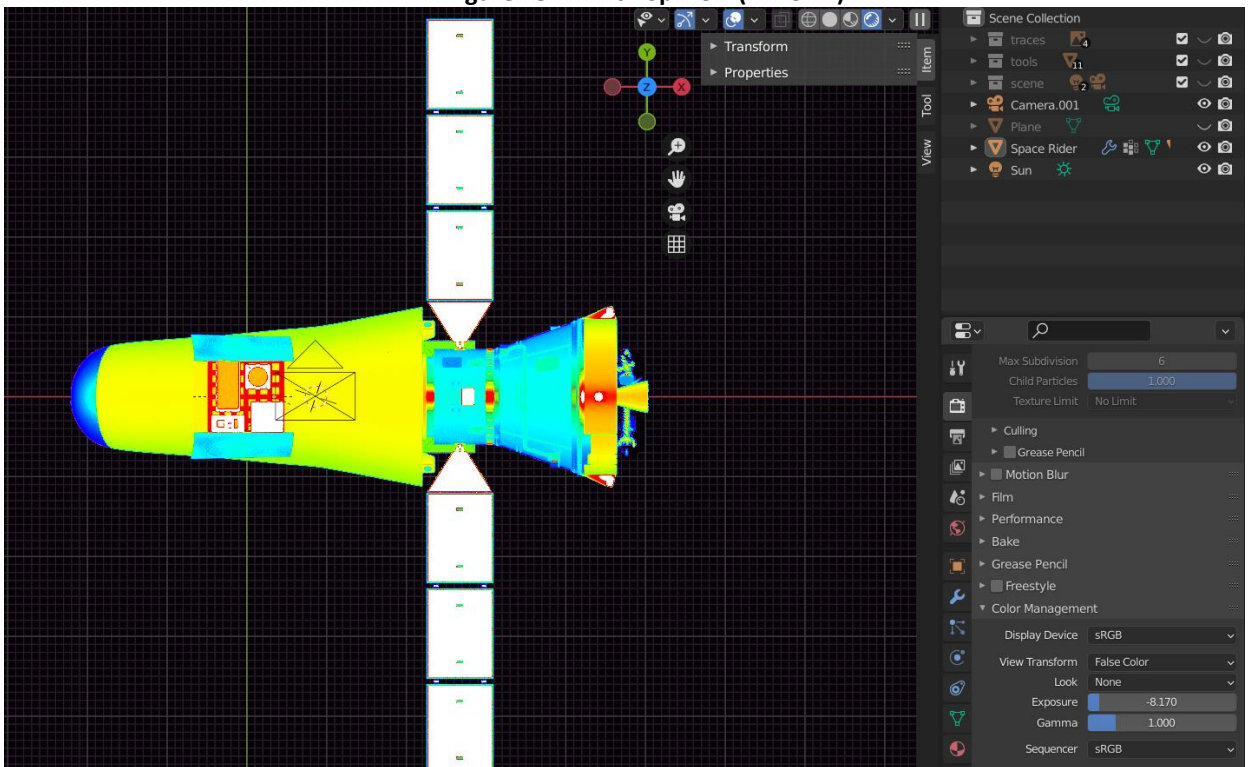


Figure 19: False color top view (EV -8.17)

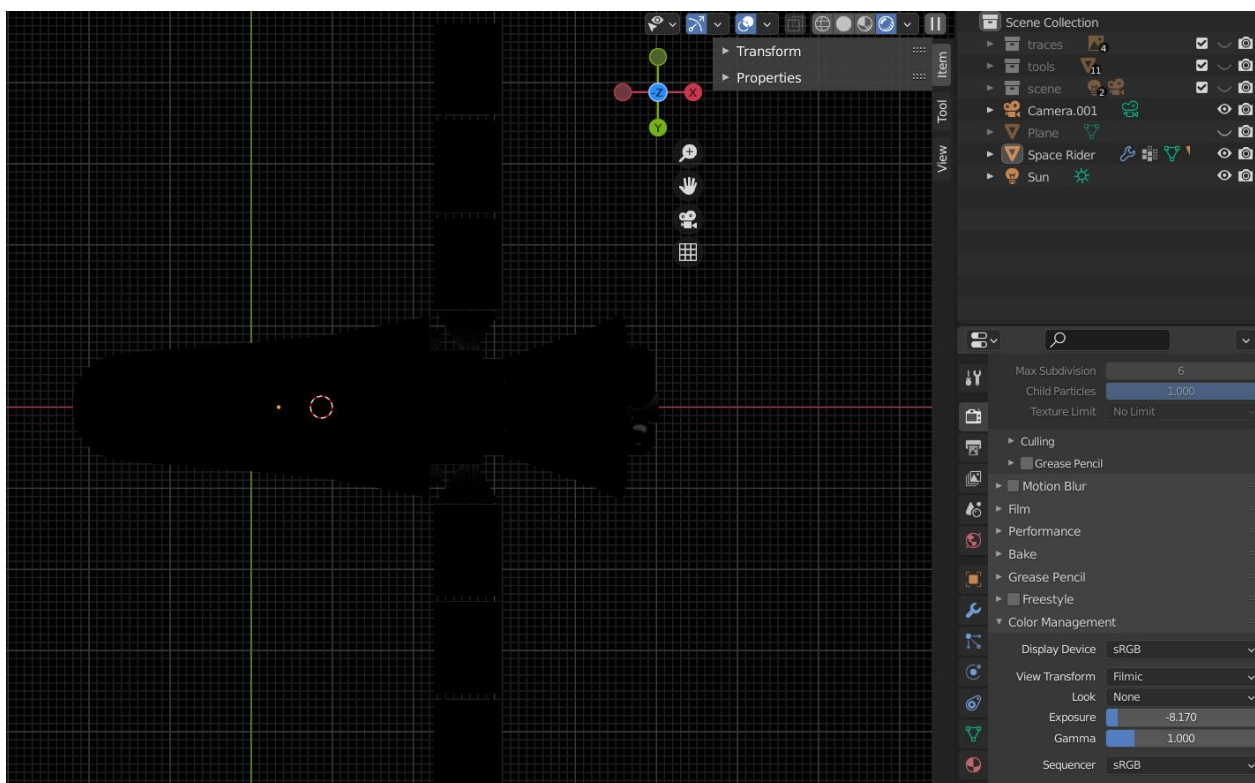


Figure 20: Filmic Bottom view (EV -8.17)

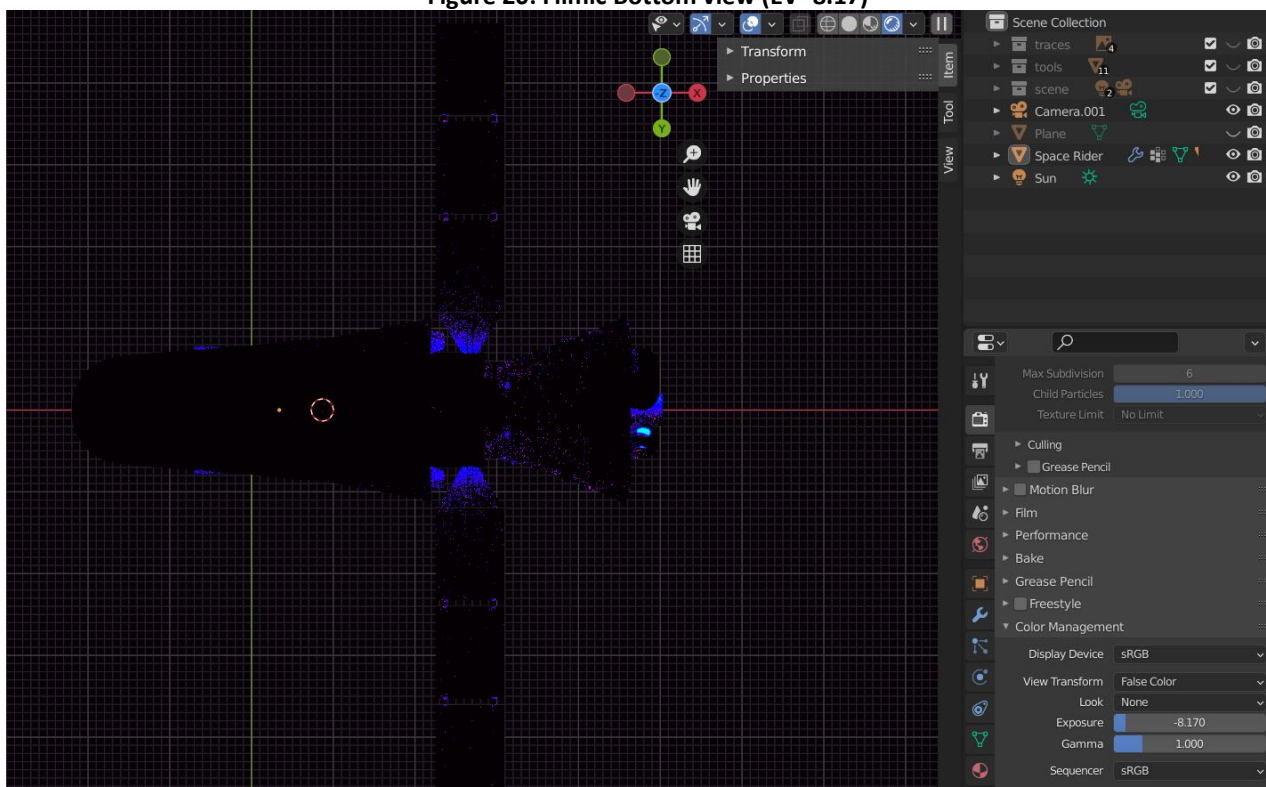


Figure 21: False color Bottom view (EV -8.17)

The exposure value -8.17 determines a completely dark external environment with SR, from which no information can be obtained. The body of the satellite, on the other hand, is well detailed at the lighting level, as only a few areas are completely white in the False Color image, so information is

lost in the drawer containing the optical payloads, on some sensors perpendicular to solar radiation and on all solar panels, which are highly reflective. The back, on the other hand, thanks to the light diffusion phenomena implemented in Blender, is not completely dark, but rather it is possible to obtain information on the lighting conditions of the shape and small parts, even if most of the body is completely black. The filmic image of the belly is well readable, but rather dark at first glance.

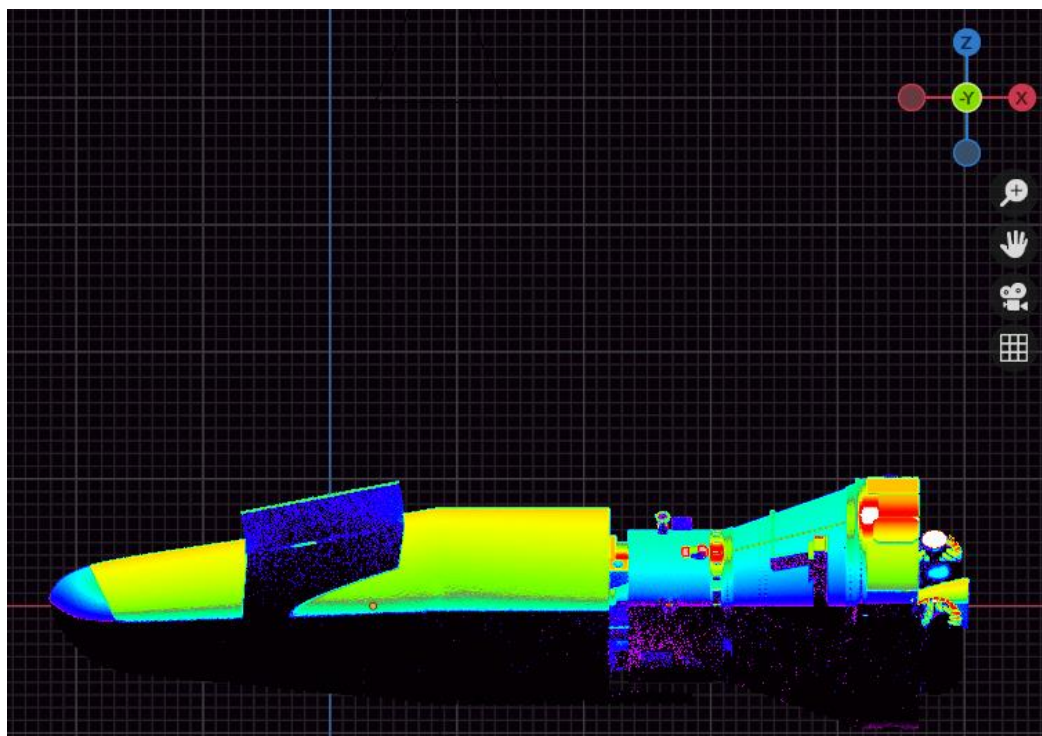


Figure 22: False color Side View (EV -8.17)

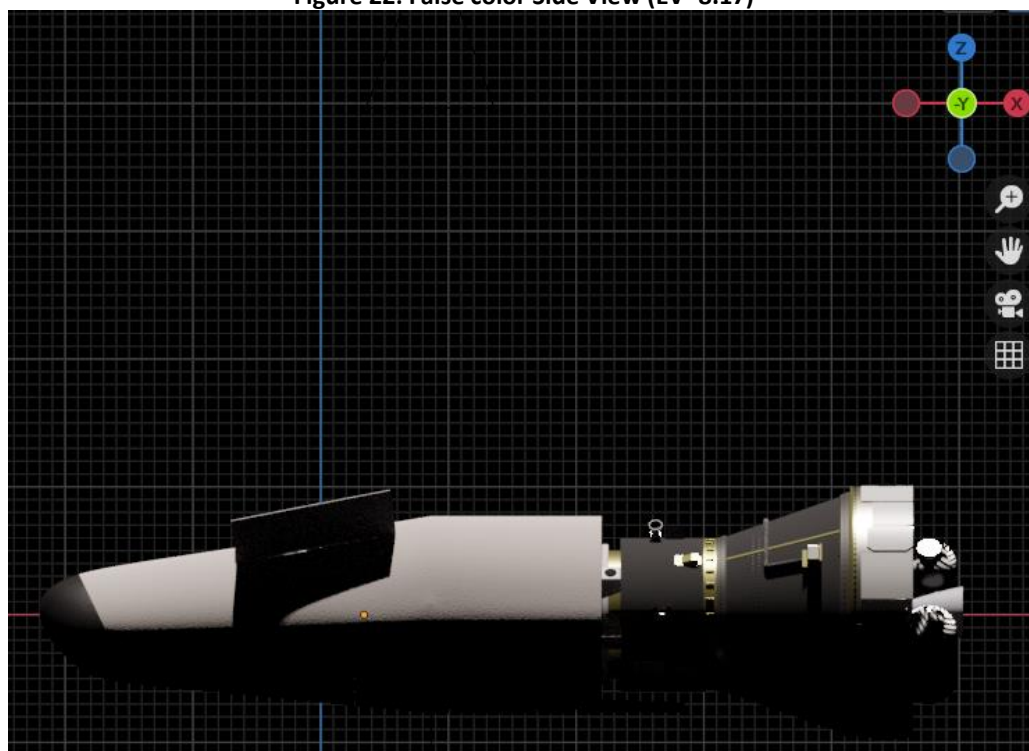


Figure 23: Filmic Side View (EV -8.17)

### 4.1.2 No White

The scene represented below is referred to upper limit exposure condition where there are no white areas on SR body (we get information from every pixel of the rendering), except for the solar panels. Exposure: -9.68.

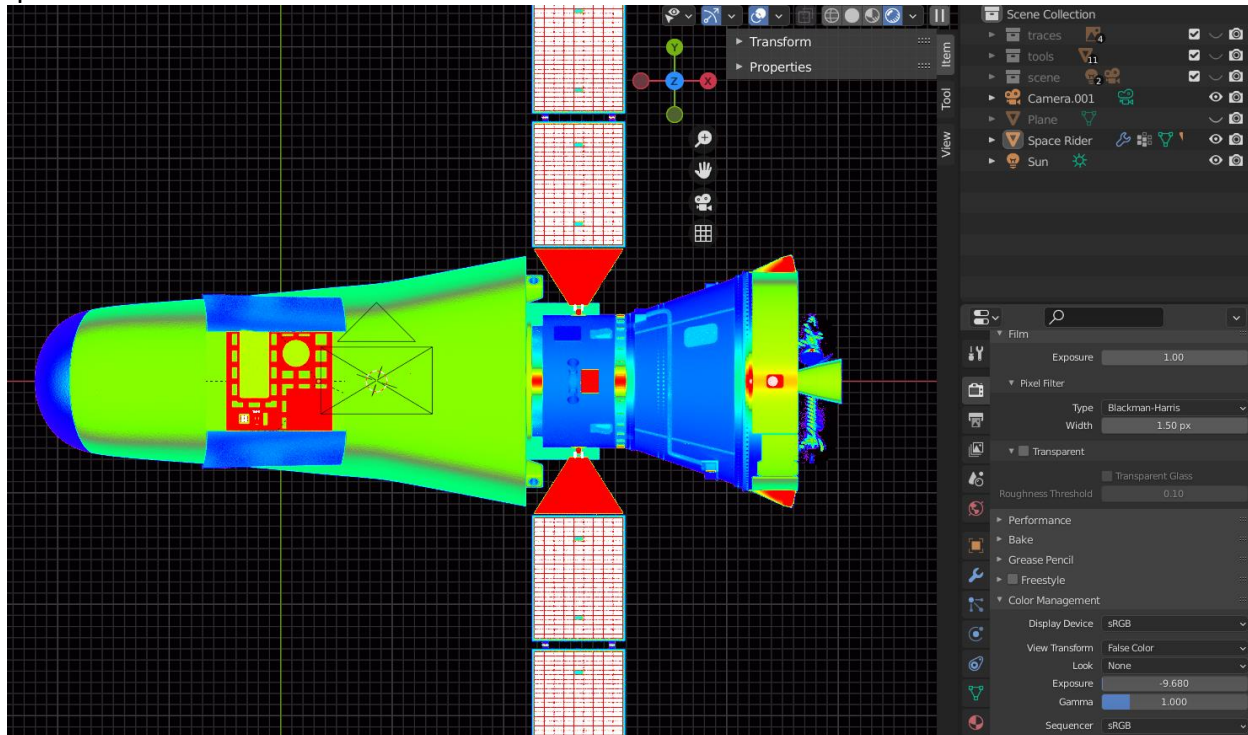


Figure 24: False color Top View (EV -9.68)

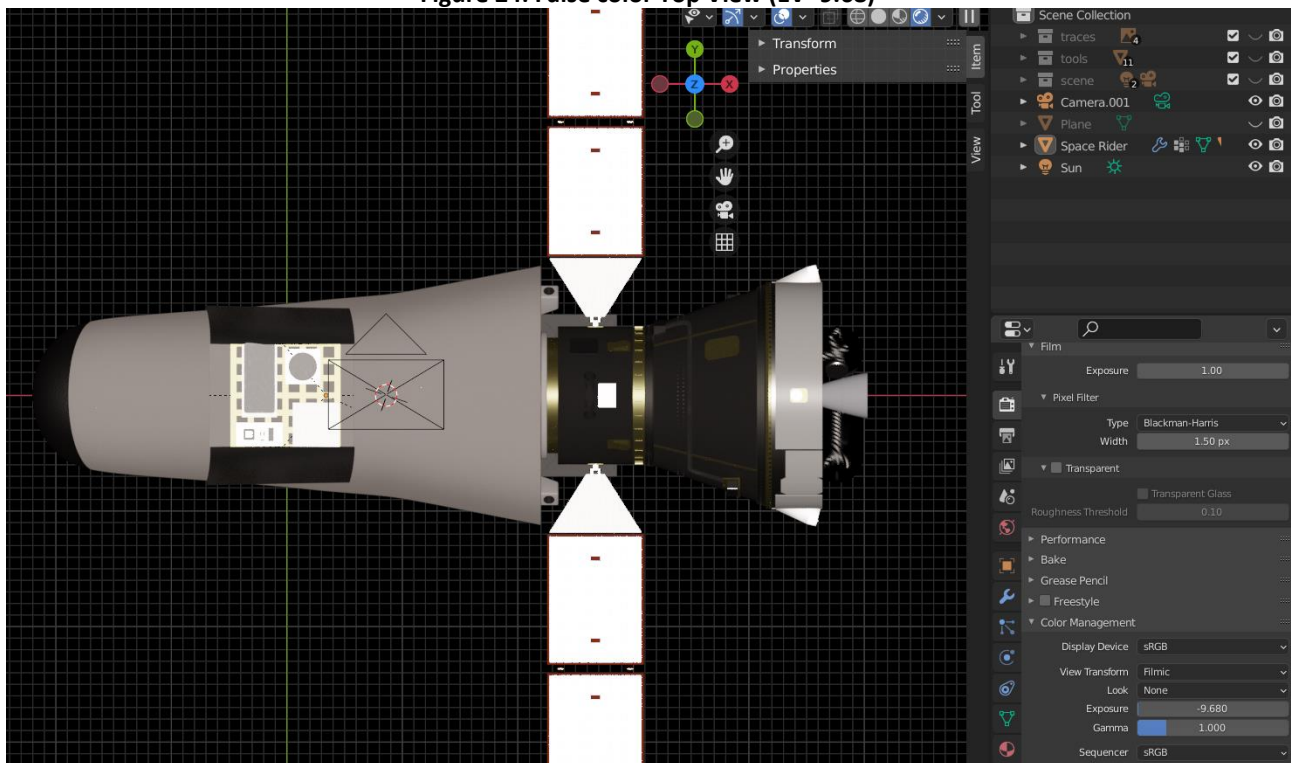
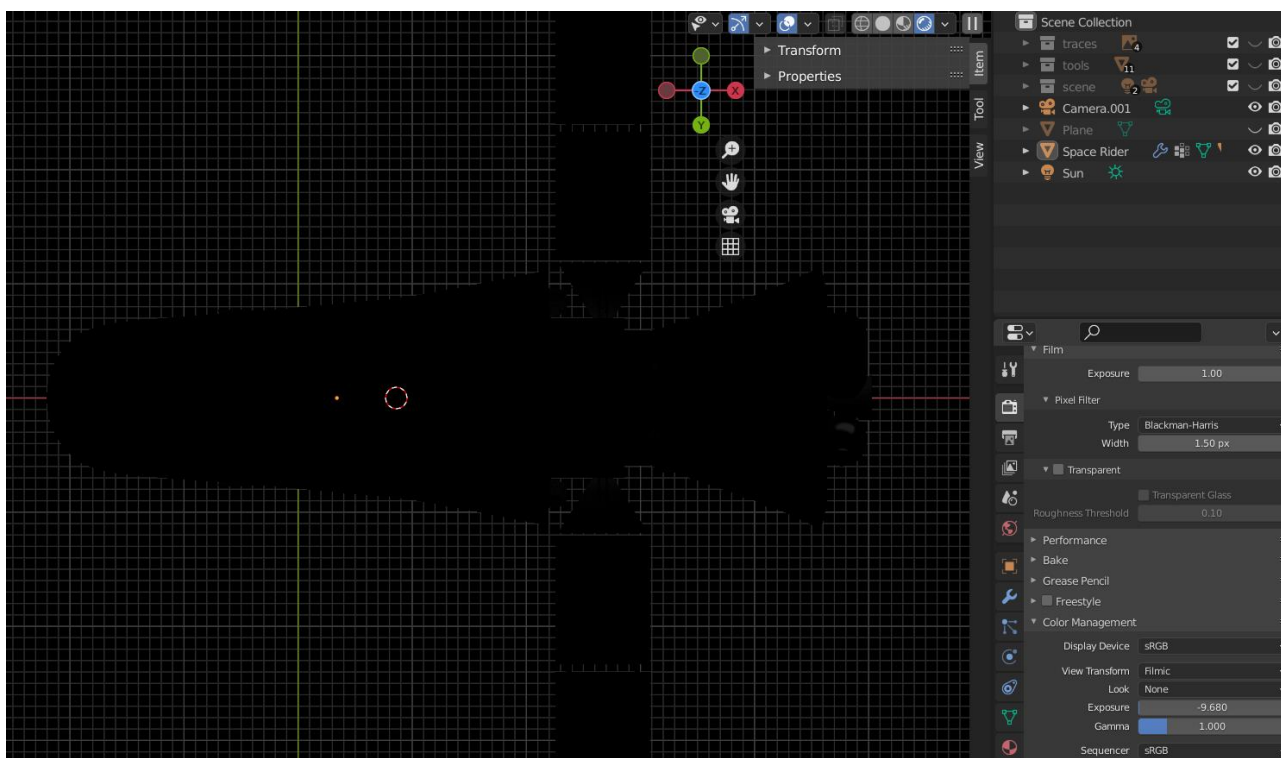
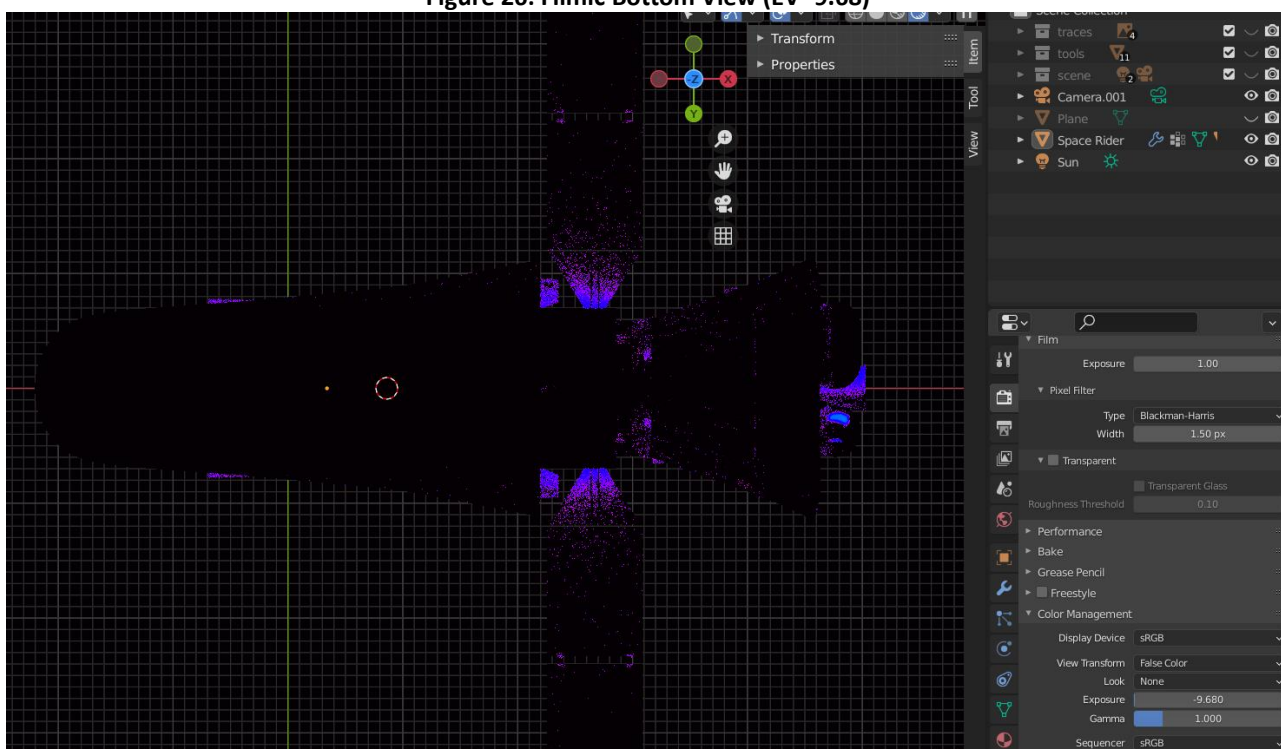


Figure 25: Filmic Top View (EV -9.68)



**Figure 26: Filmic Bottom View (EV -9.68)**



**Figure 27: False color Bottom View (EV -9.68)**

With an exposure value less than or equal to -9.68 there is no presence of completely white areas on the body of SR, thus resulting with each individual pixel within the expected dynamic range. Like the previous case, the image in Filmic is dark and dull, but detailed in all its parts.

The back also has characteristics like the previous result, with details appreciable only in a narrow portion of the edges.

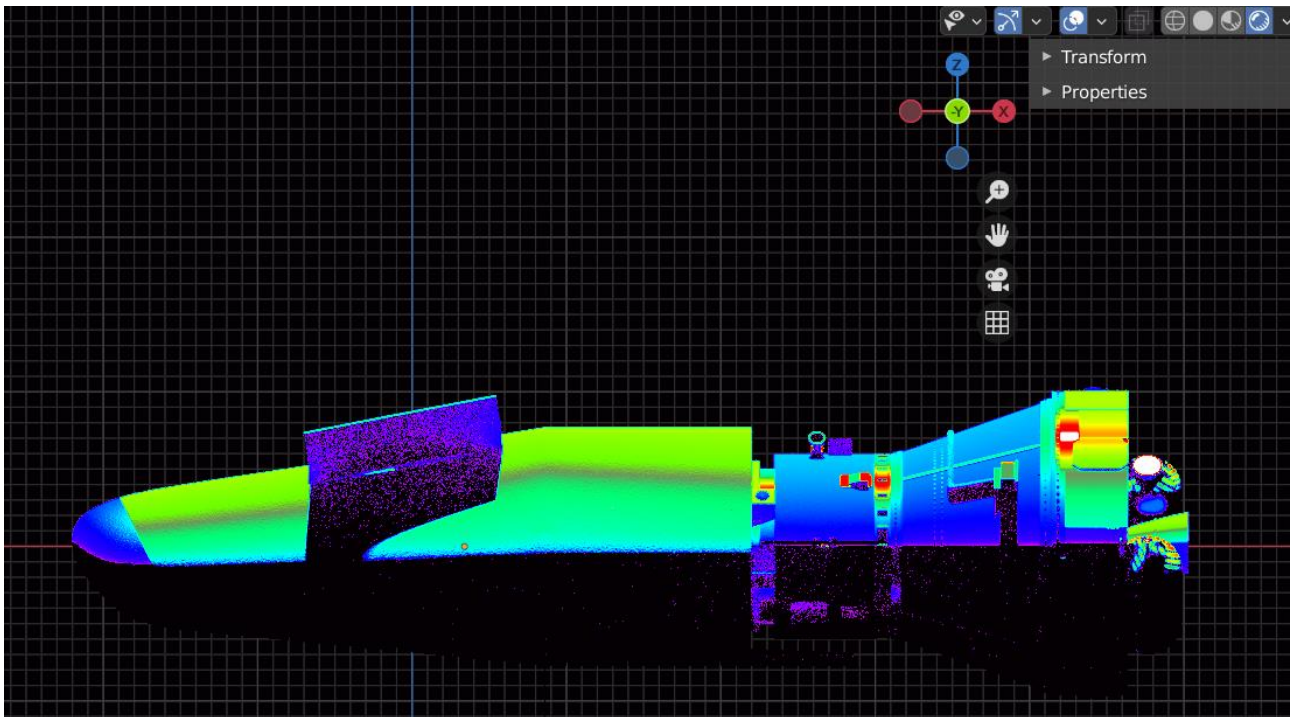


Figure 28: False Color Side View (EV -9.68)

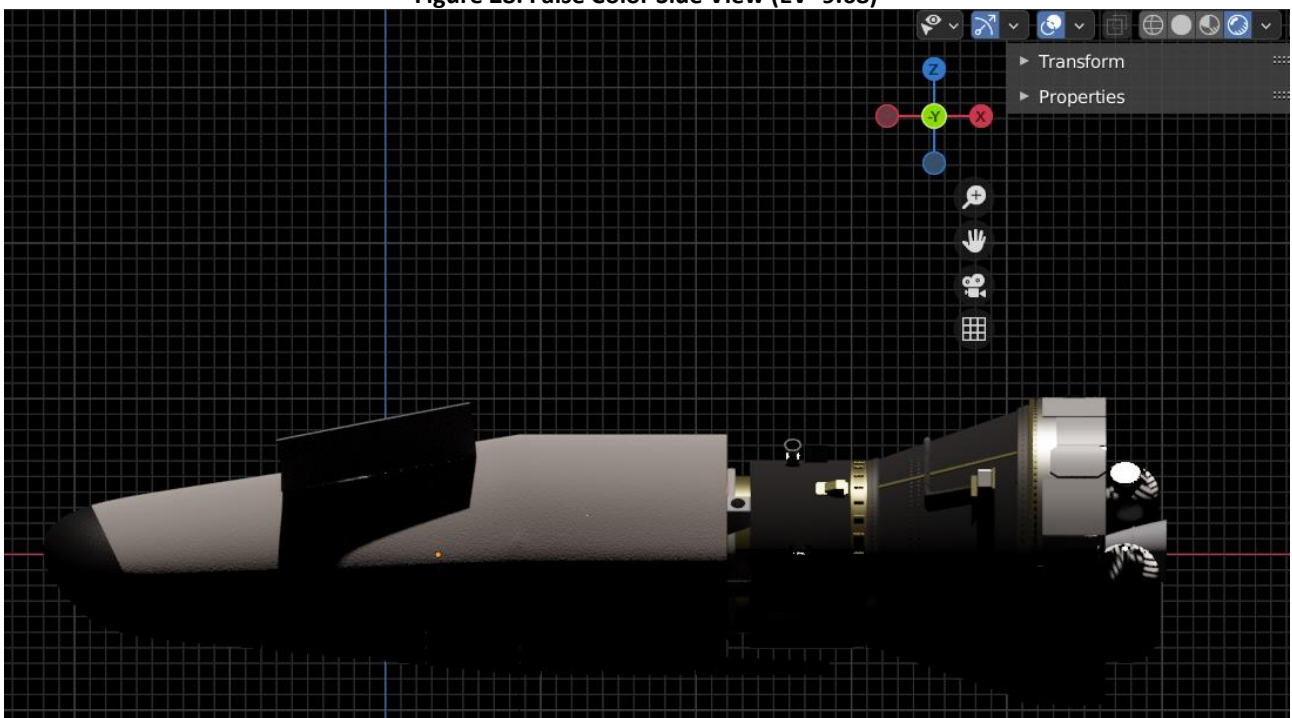


Figure 29: Filmic Side View (EV -9.68)

#### 4.1.3 Back: total information

This simulation is created to obtain the minimum value of exposure of the scene to have the possibility to get information about luminous conditions all over the surfaces in eclipse of SR. Exposure: -3.58.

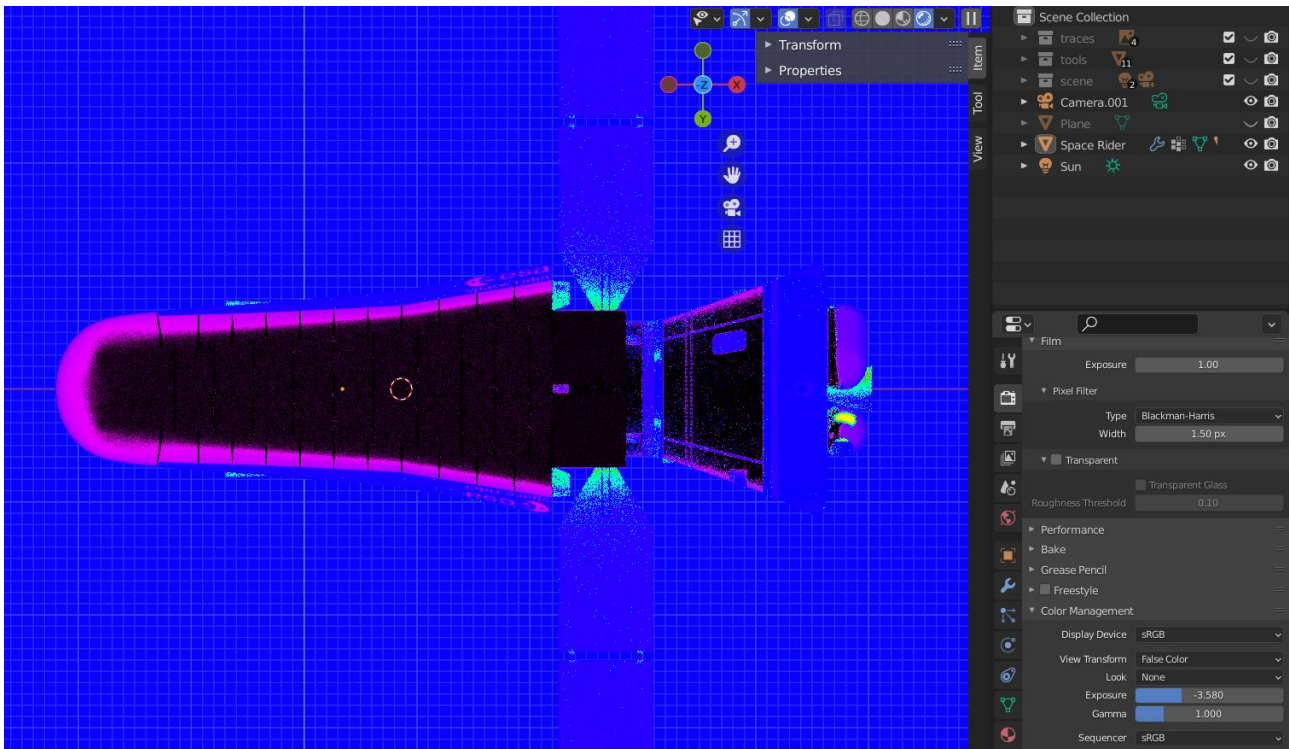


Figure 30: False Color Bottom View (EV -3.58)

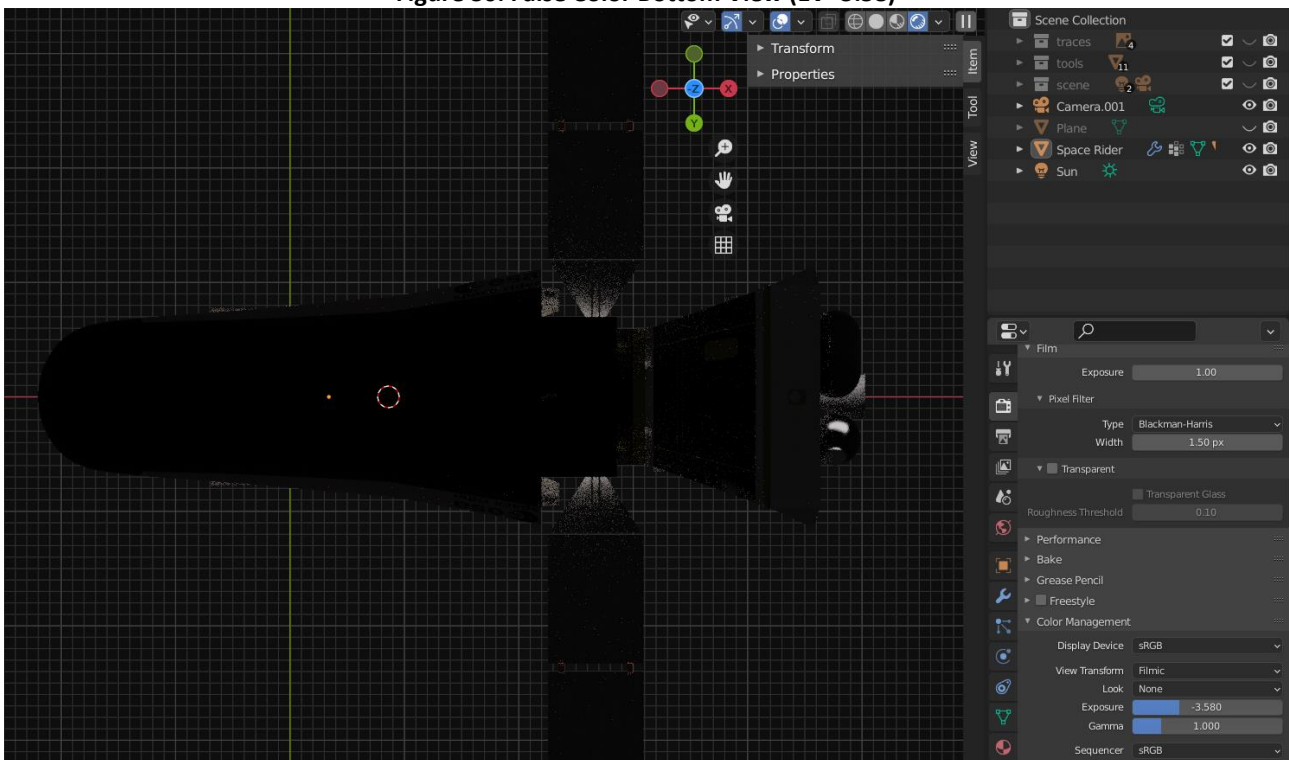


Figure 31: Filmic Bottom View (EV -3.58)



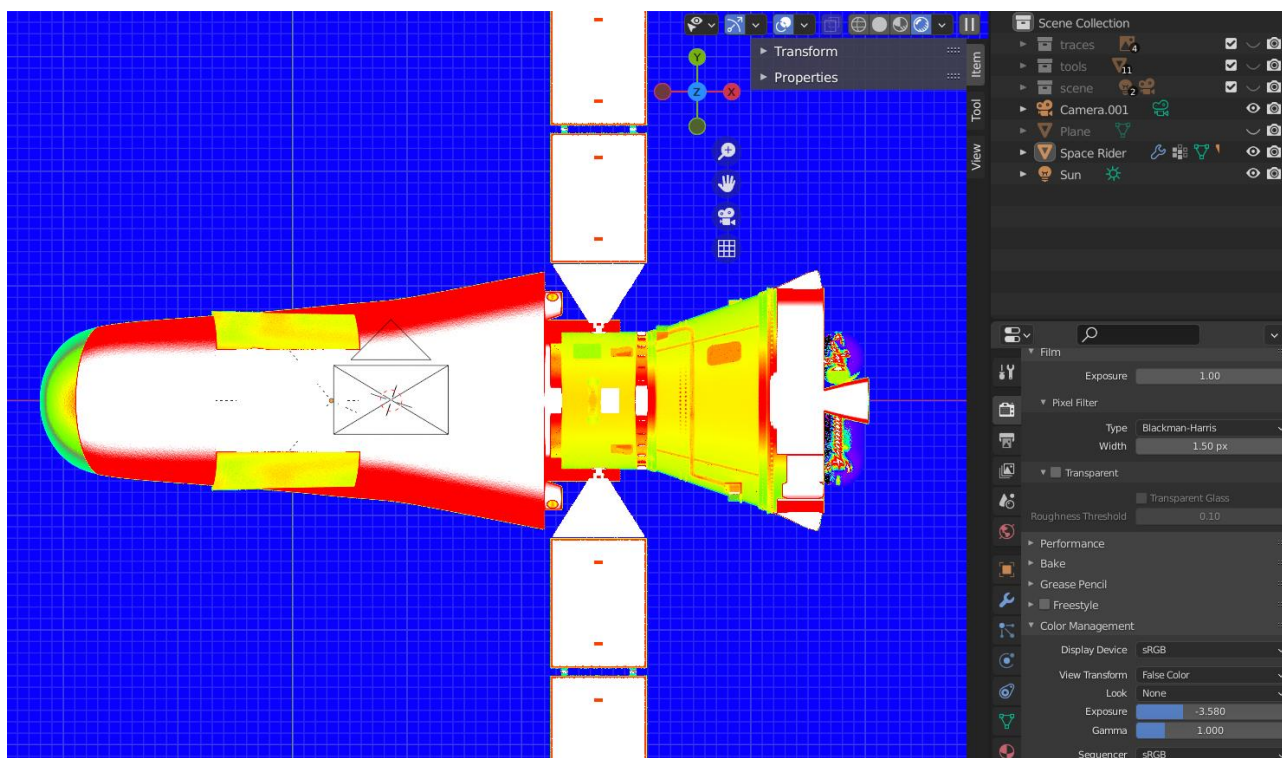


Figure 32: False color Top View (EV -3.58)

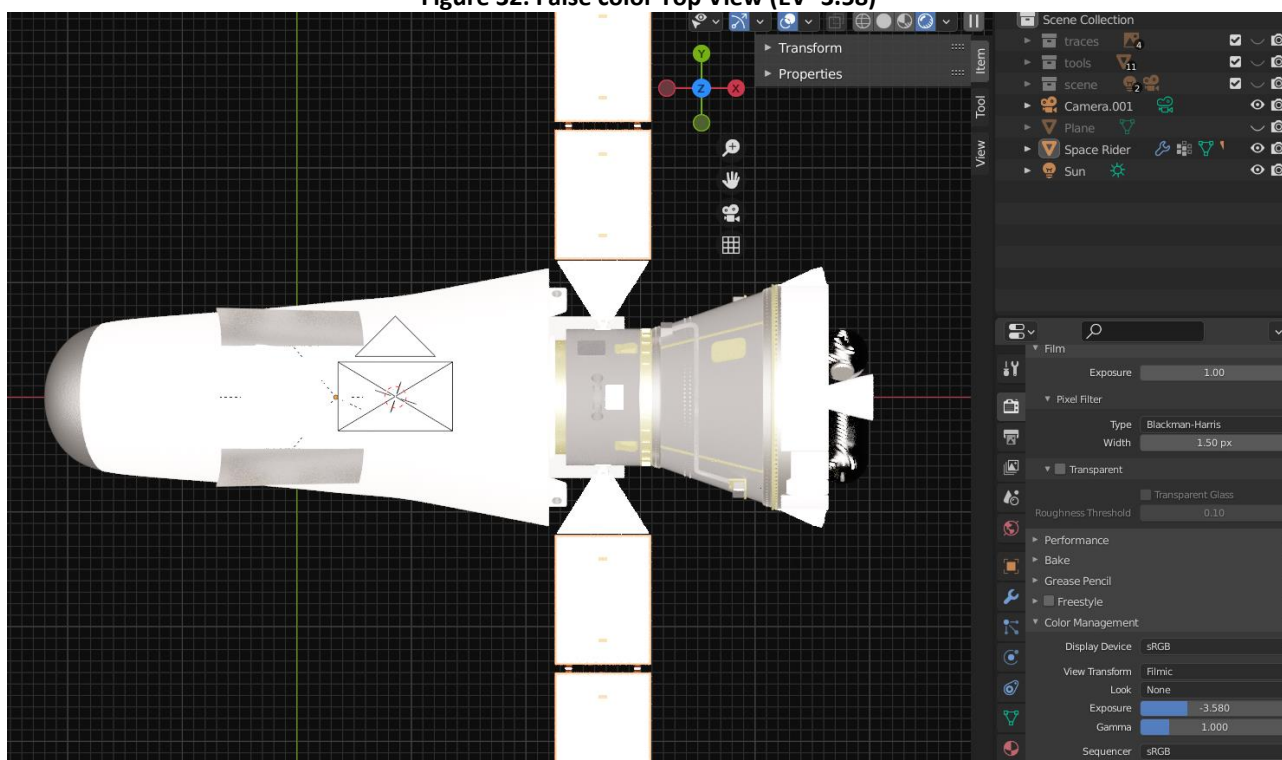


Figure 33: Filmic Top View (EV -3.58)

As can be seen from the images above, with an exposure value above -3.58 you get an image that can provide information on the entire surface in eclipse, keeping each part within the limits of the dynamic range. On the other hand, however, the illuminated part is extremely clear and overexposed, thus being impossible to analyze. The background is also not defined by a completely

dark color, but falls within the dynamic range, with the possibility of contaminating subsequent analyzes.

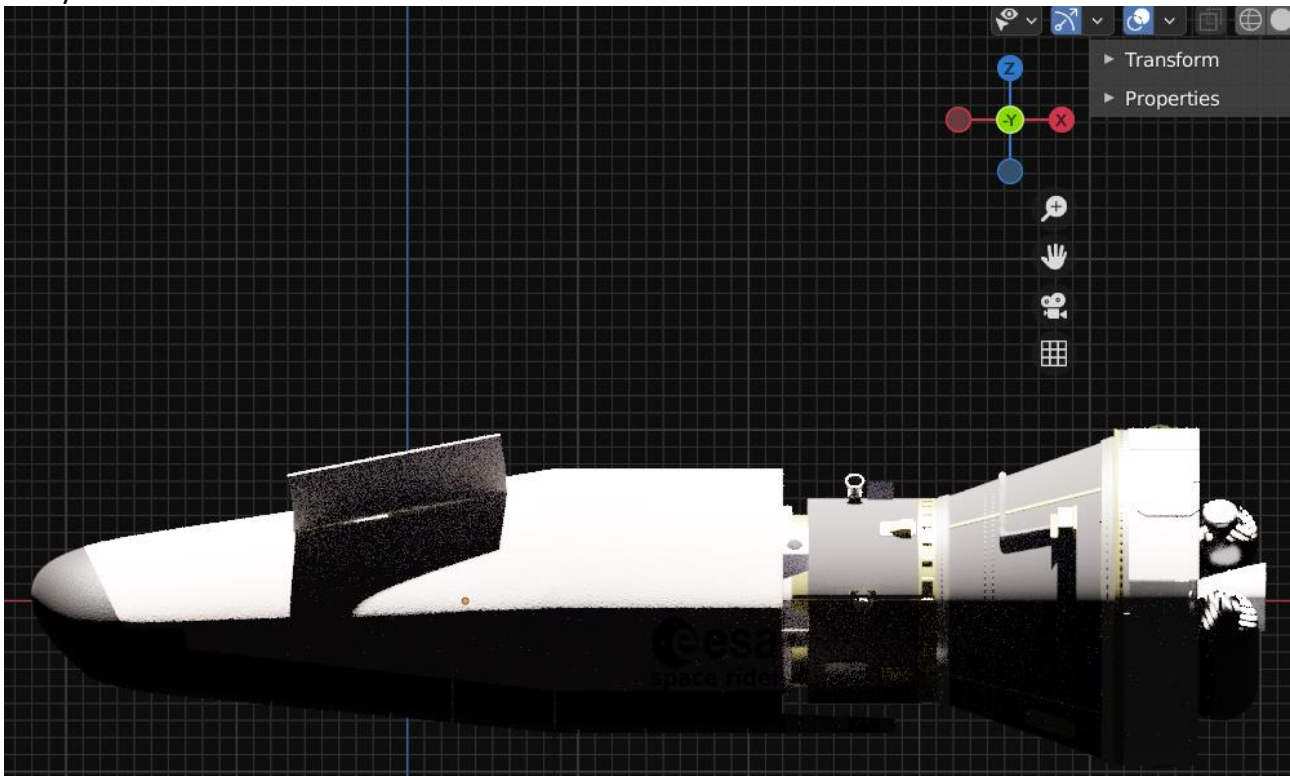


Figure 34: Filmic Side View (EV -3.58)

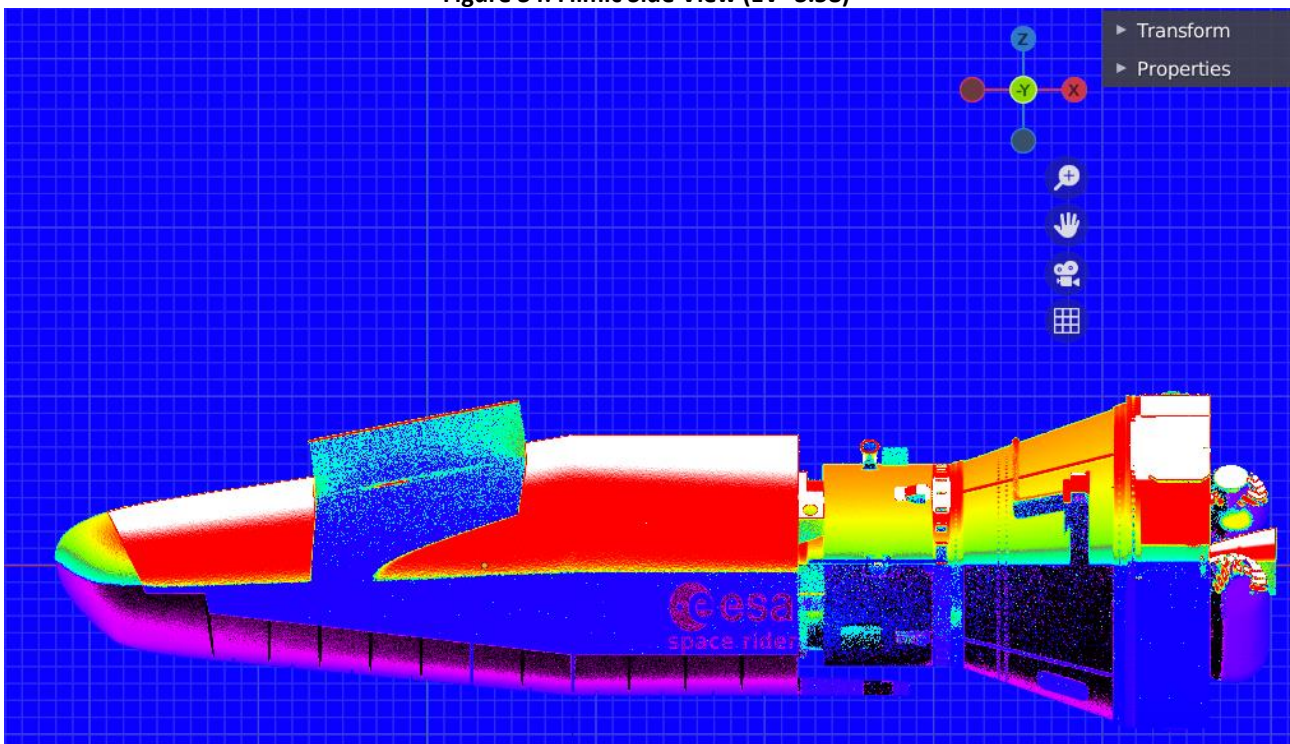


Figure 35: False Color Side View (EV -3.58)

#### 4.1.4 Literature mean value

The following simulation is obtained using a scene exposure value of -6.5. Consulting the literature available for simulations in computer graphics programs of cases similar to the one analyzed, it is found that the recommended and most used exposure varies in a range of -6.5/-7.

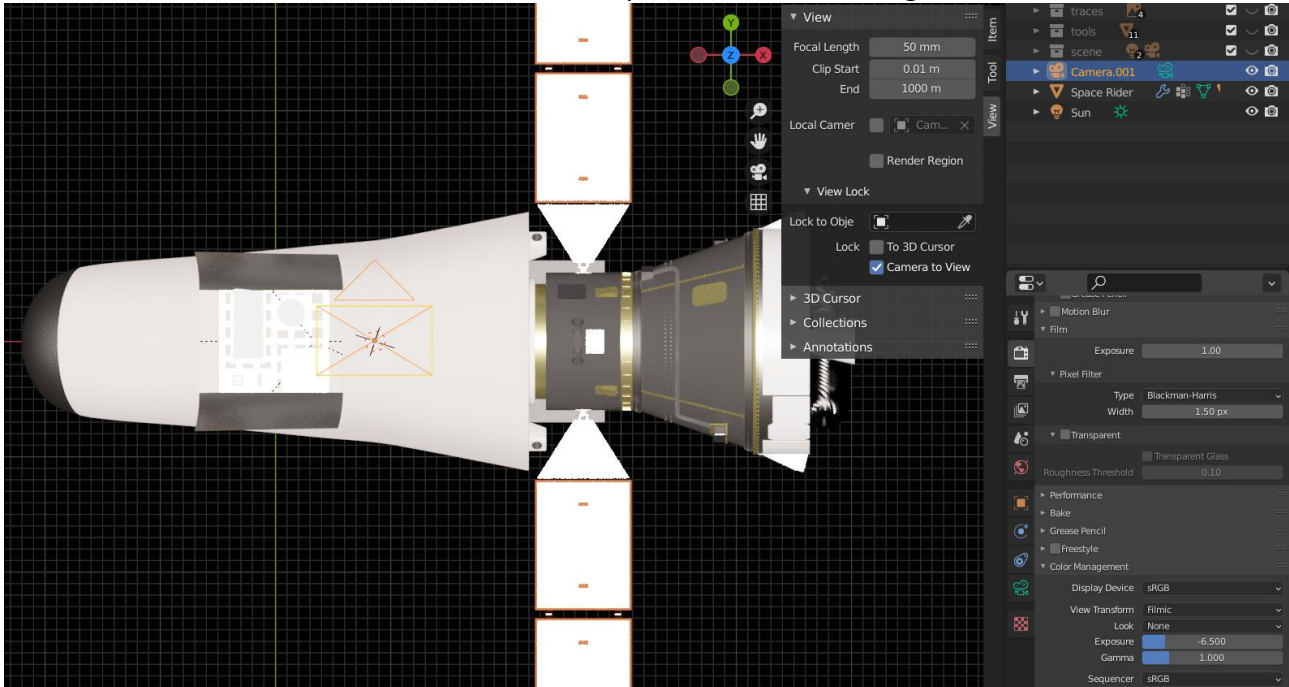


Figure 36: Filmic Top view (EV -6.5)

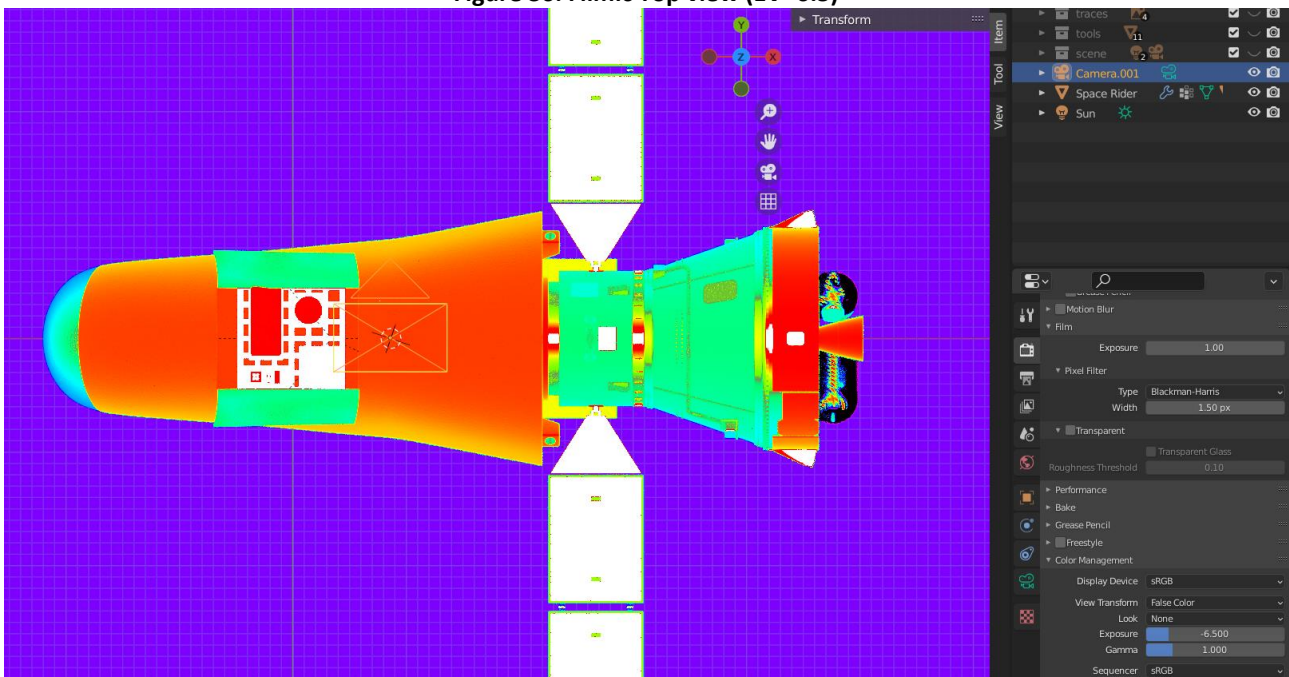
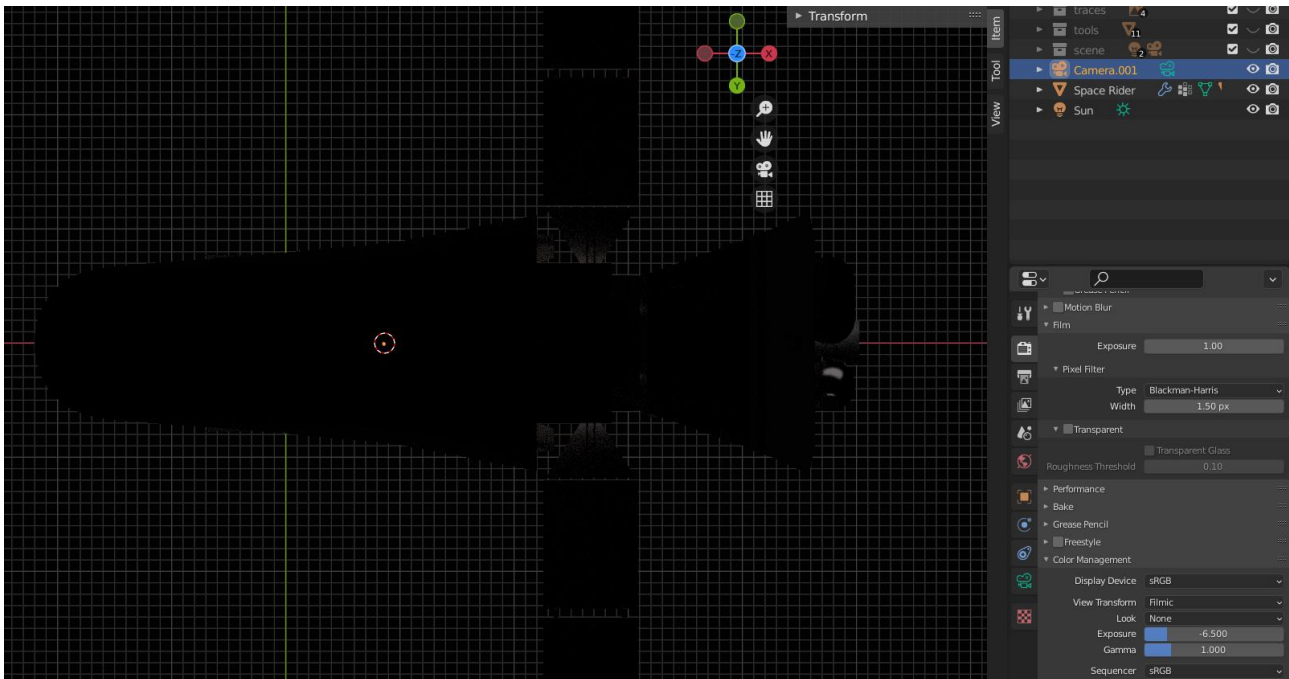
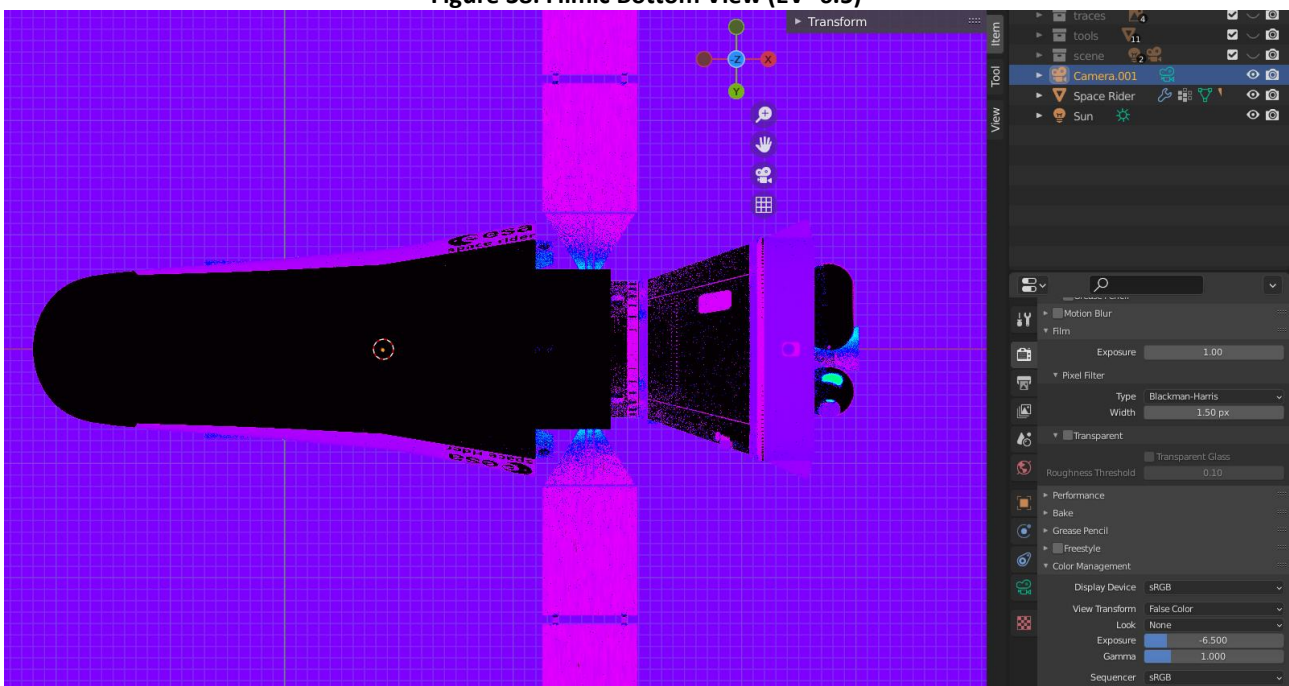


Figure 37: False Color Top View (EV -6.5)



**Figure 38: Filmic Bottom View (EV -6.5)**



**Figure 39: False Color Bottom View (EV -6.5)**

As can be seen from the images above, we are in an intermediate situation compared to the previous cases. It is therefore possible to obtain a good amount of data regarding both the belly and the back but having areas in which the analyzable limits of the dynamic range are exceeded; you also have the surrounding environment within the dynamic range itself, thus not being completely dark.

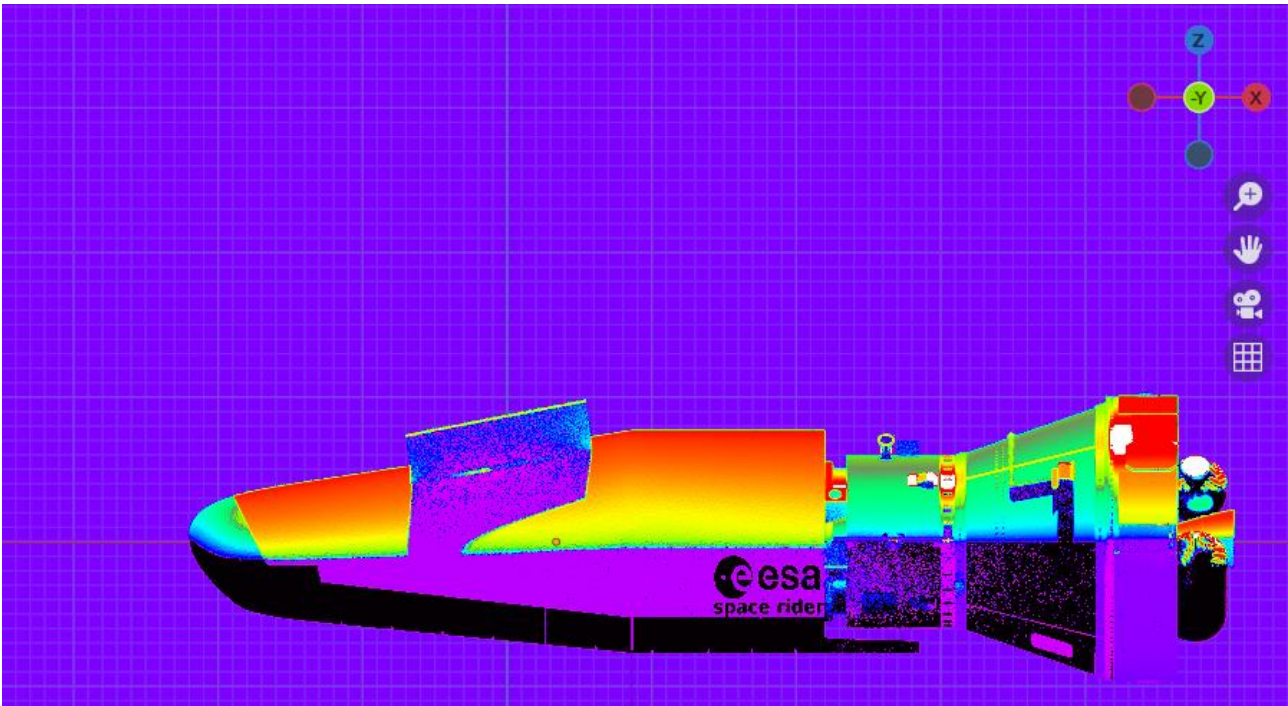


Figure 40: False Color Side View (EV -6.5)

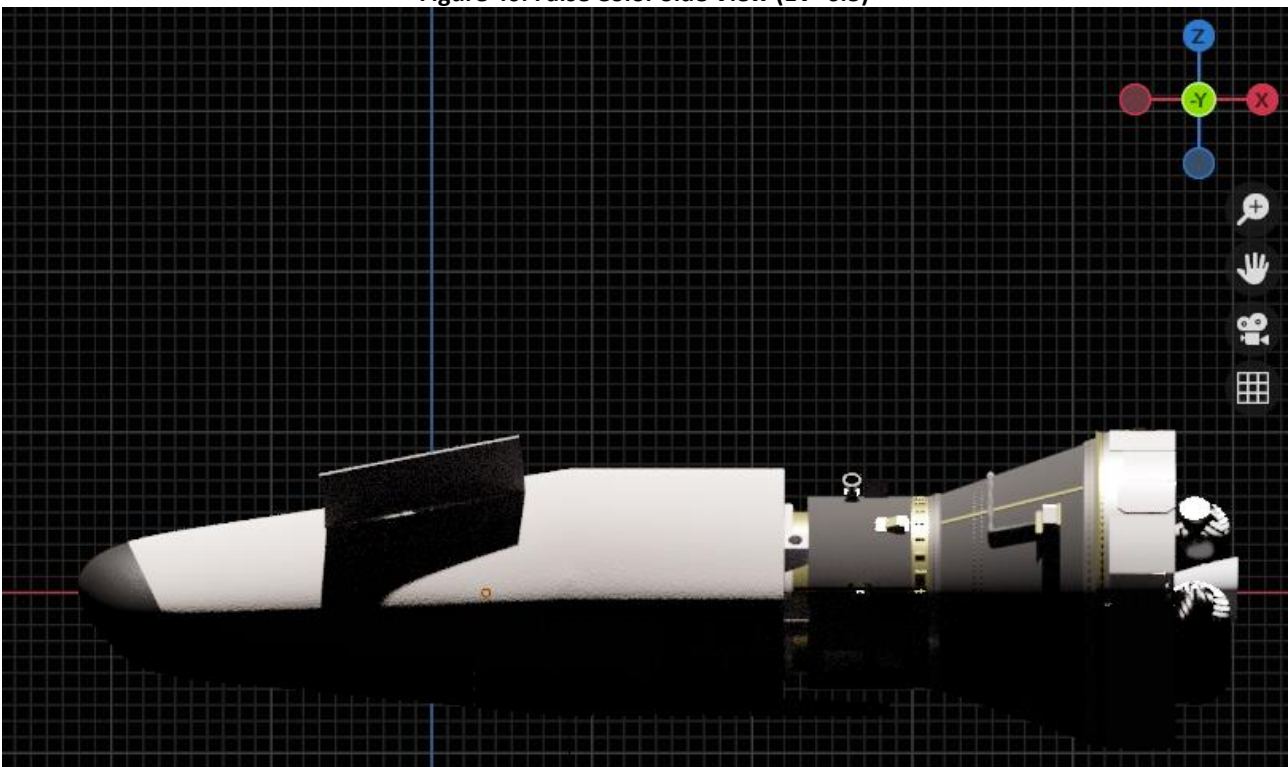


Figure 41: Filmic Side View (EV -6.5)

## 4.2 Calibration with a totally reflective plane

To uniquely evaluate the exposure parameter of the simulated scene, a fully reflective plane was taken as a reference to assess up to which exposure value it was outside the available dynamic range. The plan was created from a material with the following parameters:

1. RGB color = [1, 1, 1]
2. Metallic = 1.000
3. Specular = 1.000

The result, from different angles, is presented in the images below; it looks like a completely white plane if hit perpendicularly by light radiation, otherwise it only reflects the light source to infinity. Clearer images of the mirror are presented lower down when it is related to the SR model.

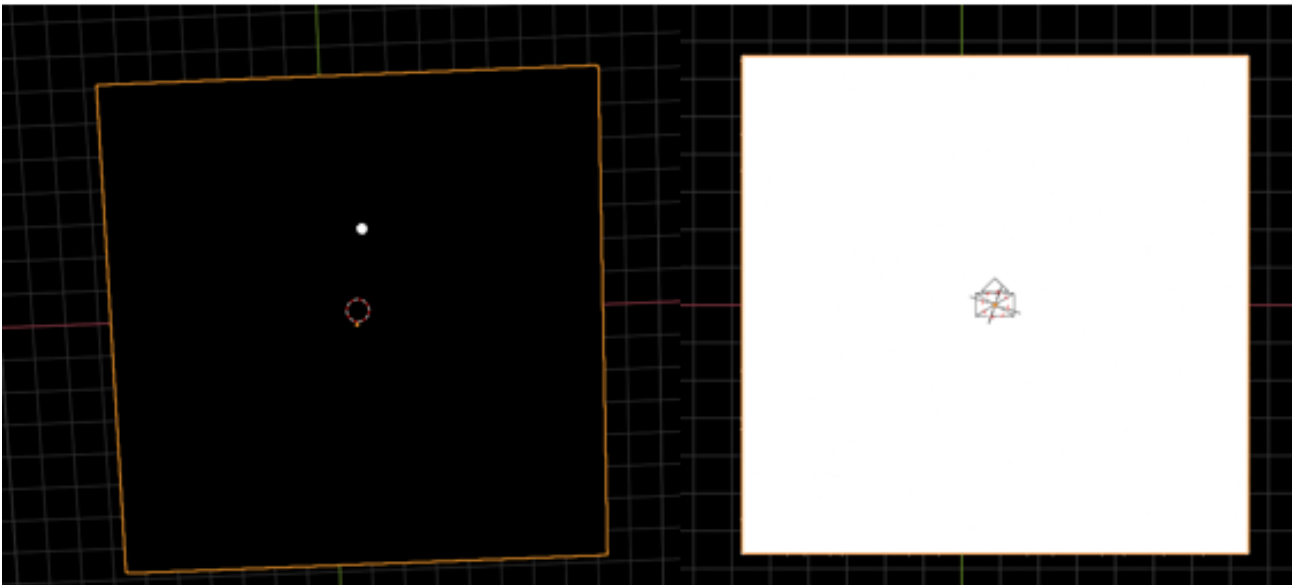


Figure 42: Reflective plane: angle variation

Going to add in the simulator also the Space Rider model you can appreciate more the mirror nature of the plane created for calibration. In the image below you can appreciate a side view of the SR-Mirror system, in which you can clearly see the reflection of the spacecraft on the plane below. Remember that the lighting conditions are true to the physically accurate model described in the previous chapters.



Figure 43: SpaceRider & Mirror

Analyzing the rendered image, we can see that the reflection of Space Rider is free of imperfections and grains, as the mirror has ideal characteristics, going to remove all the real effects due to roughness of the material or dispersion of light rays.

It is also noted how a mirror maintains the colors of the spacecraft, thus perfectly reproducing the image generated in the simulated conditions.

#### 4.2.1 Exposure analysis and evaluation

Going to evaluate the exposure and the dynamic range thanks to the use of the falsecolor visualization, the exposure value of -11.98 was identified as the maximum value for which the reflected light radiation falls within the available dynamic range. The value was identified following a procedure similar to what was reported for previous cases.

Below are the results obtained with an exposure of -11.98 in FalseColor and filmic visualization both with regard to the mirror and SR. Note how solar panels have a similar behavior to the mirror, as they are extremely bright when invested perpendicularly by the luminous flux and vice versa very dark with a different angle of incidence. The graphical display of the results, with the comparison between Filmic view and Falsecolor view on display are shown below.

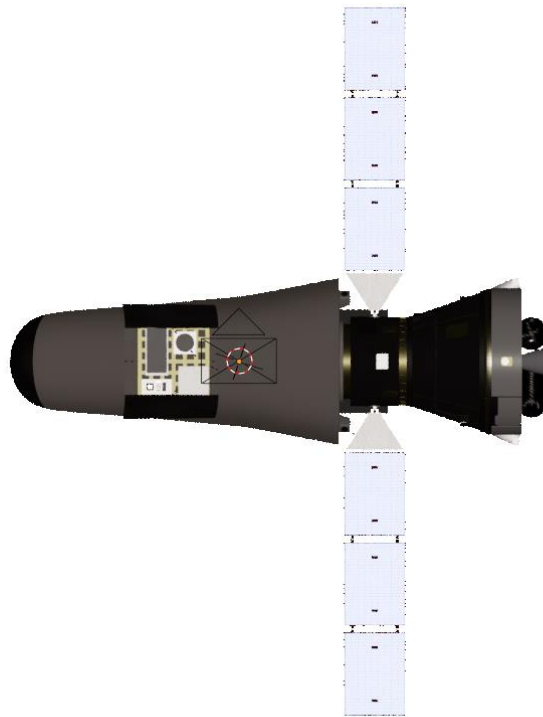


Figure 44: Filmic Top View (EV -11.98)

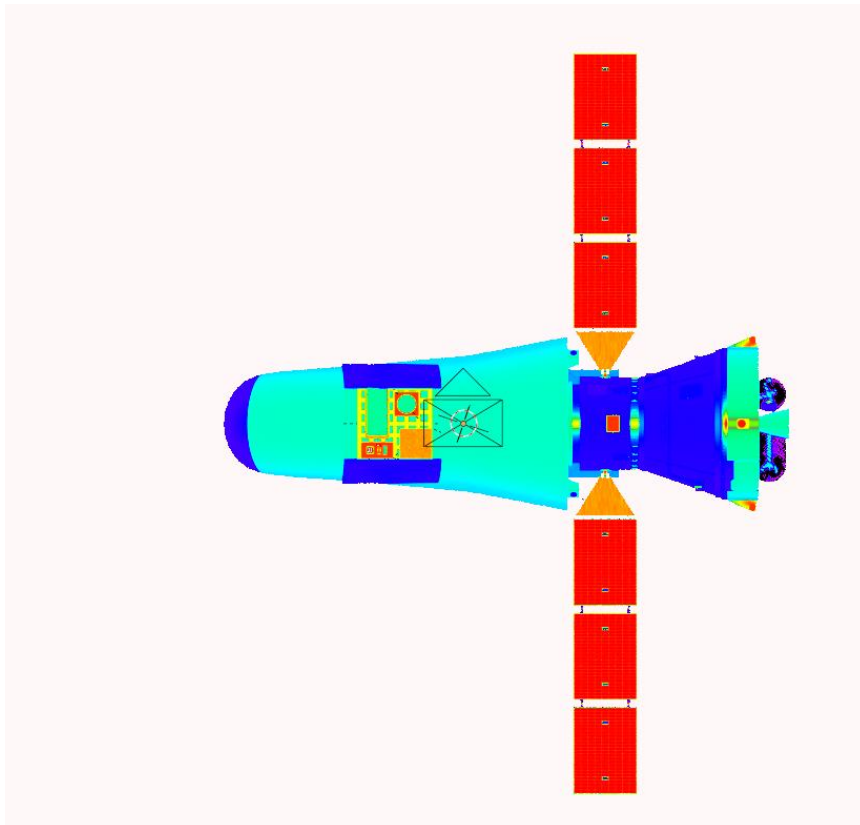


Figure 45: Falsecolor Top View (EV -11.98)



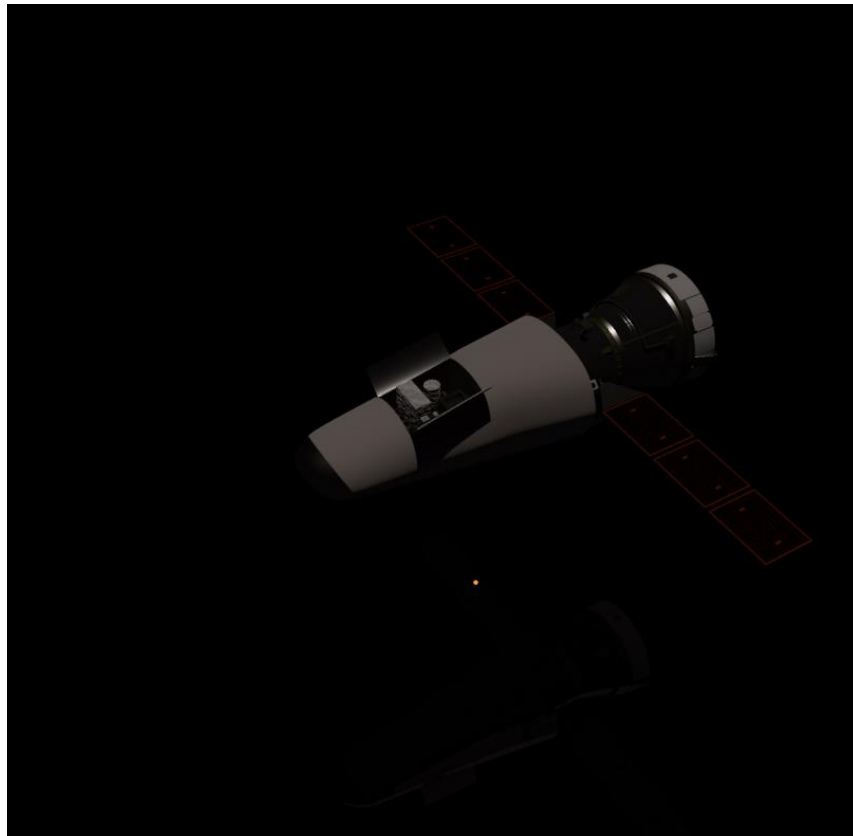


Figure 46: Filmic Side View (EV -11.98)

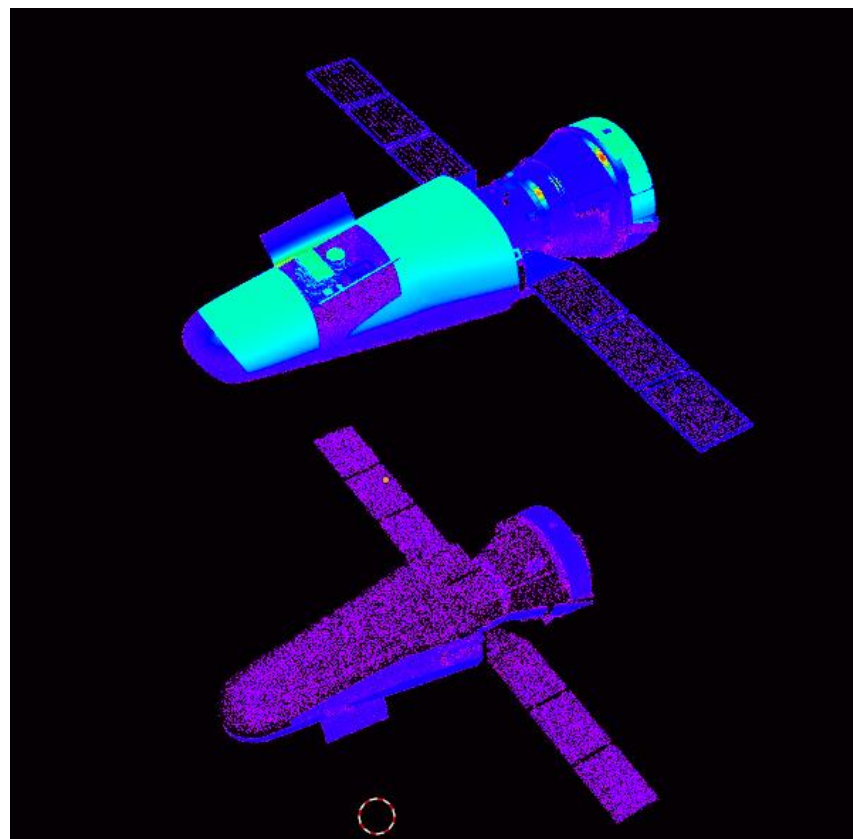


Figure 47: False Color View (EV -11.98)

Analyzing the view from above of the calibration scene, we notice how in the Filmic visualization the reflective plane seems completely white, but going to analyze the vision in Falsecolor, you get a different color from pure white, thus indicating a brightness of the plane within the dynamic range allowed by the scene and the program.

However, the exposure value thus obtained of -11.98 is low, going to make the whole scene very dark. This can be easily appreciated from the visualizations in Filmic: the body of SpaceRider that should have a color on white given the material with which it is implemented and the optical characteristics of the paint with which it is covered has instead a very dark gray color, thus going to move far from the expected result and what is expected in reality.

Only surfaces perfectly perpendicular to light radiation, such as the solar panels in Figure 44, have a color close to white while the other parts are extremely underexposed. This problem is also reflected in subsequent analyses; some tests were done and a maximum value of radiation reflected by SpaceRider of about 200 [W/m<sup>2</sup>] was obtained, much too lower than acceptable results.

We therefore tried, through the study of the choices underlying the creation of Blender, to identify a correct exposure value to be able to carry out the analyzes in the first approximation, even at the cost of losing some information due to the limited dynamic range.

### 4.3 Exposure in Blender

The main problem in the evaluation of the exposure parameter in Blender is that the modifiable parameter is not equivalent and related to the physical exposure parameter. In the real world, exposure is a photographic value that determines the conditions under which a camera captures an image. According to (22), the parameter is commonly referred to as Exposure Value (EV) and is defined by the following formula:

$$EV = \log_2 \left( \frac{A^2}{T} \right)$$

Where:

- A represents the opening of the diaphragm of the chamber.
- T represents the exposure time, which is the time in which the camera acquires light from the scene.

Being a value that depends on two different parameters, it is easy to understand that different combinations of focal aperture and exposure time can provide the same exposure values.

The measurement exposure to levels, called "STOPS", which determine the lighting conditions of the captured scene. The reference value is 16 EV, which corresponds to the standard conditions in which a scene can be observed on a sunny day and with good air transparency. The next image shows the trend of the exposure value as the focal aperture and exposure time vary, on the two axes respectively.

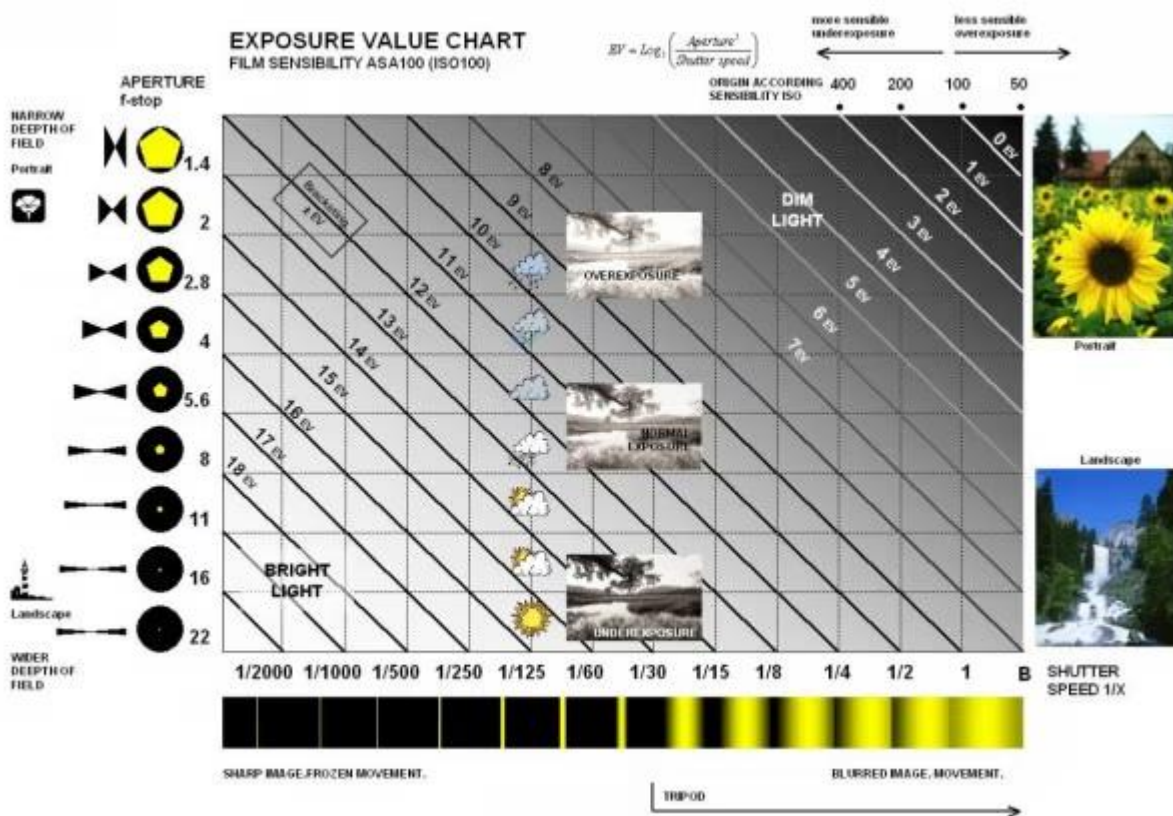


Figure 48: Exposure value Chart (Credit (22))

As you can see for EV values below 8 you get an extremely overexposed scene, vice versa for values above 16 you go to underexpose the scene. These limit values are found in very special situations when the light conditions are too light or too dark.

Considering the LEO environment similar to a sunny day in the absence of clouds, as the sunlight is direct and the atmosphere is not even present as a light attenuation filter, we try to identify the value of EV 16 also in the Blender simulation.

The problem is related to the fact that the exposure parameter modifiable in the color management of Blender varies differently from the real one and in particular the formula applies:

$$EV_{Blender} = -1 * (EV_{Real} - 9.416)$$

This implies that with an exposure parameter set to 0 you get an EV value of 9,416. This value is not random but represents the calibration choice of the software developers. In fact, going to replace this value in the EV formula, we get:

$$\left( \frac{A^2}{T} \right) = 2^{EV} = 2^{9.416} \approx 683$$

Where the value 683 represents the maximum value of luminous efficacy in the conversion between Watts and lumens at a wavelength of 555 [nm]. The choice is therefore justified for simplicity of calculation and implementation, allowing maximum effectiveness and light transmission with the parameter set by default.

It is therefore easy to obtain the desired calibration value of 16 EV:

$$EV_{Blender} = -1 * (16 - 9.416) = -6.584$$

With this value there is a loss of information due to the dynamic range and the image format, but you have a correct calibration and getting some plausible results.

Below is an example of rendering the full simulator obtained from this exposure parameter value.



**Figure 49: Correct exposure rendering**

The simulated image represents the SROC satellite's vision on SpaceRider during one of the phases of the mission around the Earth. In this rendering a previous SR model was used but you can still distinguish the different parts that compose it and that behave differently in lighting conditions. As expected, you get a plausible scene, in which the terrestrial planet and the atmosphere also have a realistic color and appearance.

You then choose to use the exposure parameter value of -6,584 for the analysis.

## 5 Image Format

A fundamental decision for the analysis to be carried out is linked to the choice of the image format to be used to obtain the best results. In Blender it is in fact possible to save the rendered images related to the simulated three-dimensional scene in numerous image formats, also being able to choose the channels that make up the saved image. In fact, each image obtained is divided into pixels, as explained in previous sections, and each pixel contains information that characterizes it. Usually, the information present is related to the color, divided into the channels related to the 3 main RGB coding colors, but channels related to the chromaticity, or the alpha value can be added. The different image formats therefore differ in the channels they use, but above all in the type of encoding and compression through which they save images. In fact, there are image formats that buy rendering data to reduce the weight in terms of memory of the image itself and others that retain all the information. Below are described the two image formats most considered within the project, with their pros and cons and with the motivation of the final choice.

### 5.1 Image Export Format: OpenEXR

Among the different formats available for the image output from rendering, in a first time it was chosen to use the OpenEXR format, as it allows to carry out the necessary analyzes more accurately. In fact, the format OpenEXR (23) is a deep raster format developed by ILM and broadly used in the computer-graphics industry, it can store arbitrary channel such as specular, diffuse, alpha, RGB, normal, and various other type in one single file and then it takes away the need to store this information in separate files. Moreover, OpenEXR stores images in a linear color space and guarantees a very wide dynamic range, necessary to be able to carry out radiometric analyzes in an accurate and differentiated way.

In Blender the OpenEXR format is not immediately recognized as a render, presenting itself as a partial result that maintains the characteristics of the scene without considering the changes to the parameters of the Color Management and the scene. You must then specify the following settings in the save:

- **View as Render**  
Display the image data-block (not only renders) with view transform, exposure, gamma, RGB curves applied. Useful for viewing rendered frames in linear **OpenEXR** files the same as when rendering them directly.
- **Save as Render**  
Option in the image save operator to apply the view transform, exposure, gamma, RGB curves. This is useful for saving linear **OpenEXR** to e.g. PNG or JPEG files in display space.

OpenEXR therefore represents an excellent image format to carry out analyzes at the base of the image itself, as it contains linear and convenient values to use and allows you to vary over a very wide dynamic range, including both very dark and very light areas of an image. However, in this phase of the project its use was not possible because this image format saves the information of the simulated scene regardless of the modification of the parameters present in the Blender Color Management and being the fundamental exposure parameter and calibrated differently from reality, all the images obtained starting from a physically accurate lighting model, as described

above, in this format they will be extremely clear and overexposed, thus implying inaccurate radiometric analysis results. It was therefore decided to use less complete formats and with a less wide dynamic range, but with more reliability with regard to lighting conditions.

## 5.2 Image Export Format: PNG

The ".png" format is one of the most used for saving photographs and digital images. PNG stands for "Portable Network Graphics" and is an image format created in 1995. It allows you to save images in lossless mode, that is, without loss of information.

It is characterized by 3 main channels for each pixel, related to RGB colors, with the possibility of saving an additional channel called Alpha, which represents the transparency of the single pixel. Each channel varies in a range between 0 and 255, thus allowing to represent a total number of different colors equal to:

$$N_{Colors} = 256^3 = 16777216$$

This value is much lower than in more complex formats such as the OpenEXR.

However, we chose to use this format for the analysis for its simplicity and accessibility, in addition to the non-negligible fact of being able to obtain and save images while maintaining the features implemented in the Color Management on Blender.

The colors are also saved starting from the sRGB color space, referring to the display used and therefore influenced by the boundary conditions. To carry out a photometric and radiometric analysis, as described in the next chapter, it is necessary to obtain linear values of the color channels of the image, so that they are absolute and not related to the simulation conditions. The values are also normalized in a range between 0 and 1 in order to be used more effectively.

Below is graphically represented the range of the sRGB color space on a CIE 1931 space, it is noted that this space can effectively represent only the colors present within the triangle that characterizes it, going to lose information regarding the colors outside. The vertices represent the primary colors in this type of coding and the point D65 represents pure white, which is not at 6500 ° K on the temperature curve of the black body, but slightly displaced, going to represent in white the condition of daylight filtered by the atmosphere.



Figure 50: CIE 1931 xy chromaticity diagram and sRGB color space (Credit: Wikipedia)

As mentioned, you want to get the image in the RGB linear color space, the transformation that must be done starting from the RGB color space is as follows:

$$C_{linear} = \begin{cases} \frac{C_{sRGB}}{12.92} & \text{if } C_{sRGB} \leq 0.04045 \\ \left( \frac{C_{sRGB} + 0.055}{1 + 0.055} \right)^{2.4} & \text{if } C_{sRGB} > 0.04045 \end{cases}$$

The linear format of the colors is therefore obtained as desired, in a similar way to what happened for the output of the OpenEXR format but included in a much wider dynamic range.

## 6 From Scene to Simulated Image

This section describes the main steps that compose the acquisition of scene data by the simulated sensor and the transformation of the data, adding the losses and physical effects present in the real sensor, up to a simulated image, according to (24) and (25).

The process can be schematically divided into the following steps:

- SCENE: creates a radiometric description of the scene.
- OPTICS: converts scene radiance data into an irradiance image at the sensor surface.
- SENSOR: converts the irradiance image into electrons.
- PROCESSOR: converts the digital values in the two-dimensional sensor array into a three-dimensional (RGB) image that can be rendered on a specified display.
- DISPLAY: generates a radiometric description of the final image as it appears on an LCD display.

### 6.1 SCENE

The scene represents the starting point for radiometric analysis. It is rendered from a 3D model containing the Earth, SpaceRider, a camera that simulates the presence of SROC, an atmosphere and a lighting model. The 3D environment is then transferred and captured in a two-dimensional image in the format described above, retaining all the characteristics imposed on the scene itself.

During this discussion, the initial image is considered the starting point from which to obtain a sensor output simulation and not the final result.

#### 6.1.1 SCENE: Analysis Workflow

The logic process that allows to obtain the desired results for radiometric analysis is composed of numerous steps, which transform the simulated scene in Blender into a matrix that can be analyzed through an optical sensor, which is installed on SROC. The output of one step represents the input for the next step; The workflow is defined as follows:

1. Blender Render: OpenEXR/PNG Image.
2. OpenEXR/PNG Image: RGB linear values for each pixel.
3. RGB Map: Relative Luminance Map.
4. Relative Luminance Map: Light Flux Map.
5. Light Flux Map: Numbers of Photons Map.

Once the number of photons that hit each individual pixel under ideal conditions has been obtained, the radiometric analysis can be considered concluded. In the following chapters the process that leads to the creation of a simulated image starting from radiometric analysis is discussed, considering all the real effects that occur in space and within the optical system.



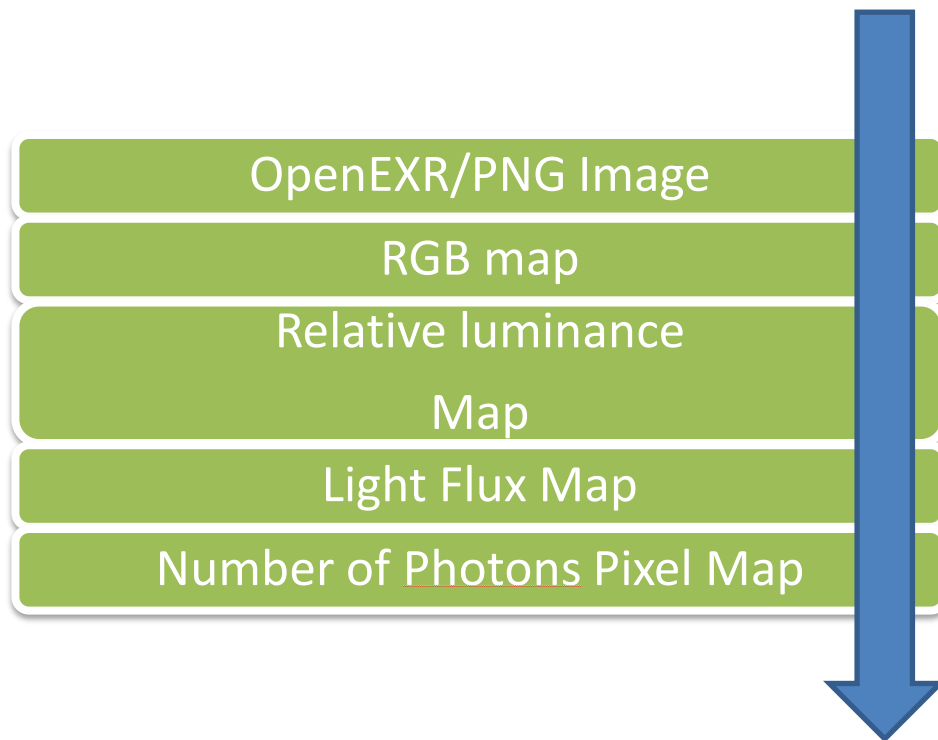


Figure 51: Scene workflow

### 6.1.2 From OpenEXR/PNG to RGB map

Once the simulated scene on Blender has been fully developed, we proceed with the rendering, going to obtain a frame of the spatial scene, which must be saved in the OpenEXR or PNG format as described before, which will then be processed as described below.

Using the script python *PostProcessing.py* it is possible to obtain information from the OpenEXR image file. Thanks to the OpenEXR library in python it is possible to obtain color values from every pixel of the image. The color Data are stored in a linear RGB format, which allows us to have no data loss and can be easily used for subsequent analysis. To achieve this result, conversions between the different color spaces are necessary and a linearization is carried out between the sRGB model and the linearized RGB model. This conversion is described in detail in the main body of the thesis. This first part of the code provides a vector containing the RGB values for each pixel of the source image, seen as a *width\*height* size matrix.

### 6.1.3 RGB values to Relative Luminance

Starting from the RGB values of the pixels, it is possible to calculate the Relative Luminance for each. Relative luminance is usually indicated as Y and it follows the photometric definition of luminance L including spectral weighting for human vision, but while luminance L is a measure of light in units such as  $\text{cd}/\text{m}^2$ . Relative luminance Y values are normalized as 0.0 to 1.0, with 1.0 being a theoretical perfect reflector of 100% reference white. Like the photometric definition, it is related

to the luminous flux density in a particular direction, which is radiant flux density weighted by the luminous efficiency function  $\gamma(\lambda)$  of the CIE Standard Observer.

Relative Luminance relative luminance can be calculated from **linear** RGB components in the following way:

$$Luminance_{REL} = 0,2126 * R_{lin} + 0,7152 * G_{lin} + 0,0722 * B_{lin}$$

A relative luminance map is then obtained that provides information from each pixel on its ability to reflect and transmit to the sensor the solar flux that invests it.

#### 6.1.4 Relative Luminance to Light Flux

For the analysis we considered a value of Relative Luminance equal to 1 corresponding to a pixel that represents a surface completely reflecting the incident light radiation. Starting from this, the value of the Luminous Flux was scaled according to the value of Relative Luminance; this results in a map that indicates a value of Luminous Flux emitted by each pixel, calculated in Watts per square meter.

Further considerations will have to be made on the solid angle and the portion of the area represented by the pixel in order to determine the amount of Watts that reaches the optical sensor. As the starting light radiation, the average value of the solar flux that reaches the top of the Earth's atmosphere was used, without considering any fluctuations and losses due to the presence of solar phenomena and the atmosphere itself. The flow value is calculated as follows:

$$Flux_{pixel} = Luminance_{REL} * Flux_{Sun}$$

Where Solar Flux = 1367 [W/m<sup>2</sup>].

We then obtain a map in which each pixel is associated with a value of luminous flux measured in Watts per square meter.

#### 6.1.5 Photons Number Calculation

The objective of the radiometric analysis is to provide as a final output the number of photons that reach the optical sensor of the camera used on the SROC CubeSat in such a way as to be able to evaluate the behavior of the latter in the different conditions in which it will be in relation to SR during the orbital period.

It is possible to calculate the number of photons from the energy formula of a single photon:

$$E_{photon} = h * \frac{c}{\lambda}$$

Where h is Plank's constant; c is the speed of light in a vacuum and  $\lambda$  is the wavelength expressed in nanometers of light radiation.

Multiplying the energy of the single photon by the total number of photons yields the total power emitted by a light source. The number of photons can then be calculated as:

$$n = \frac{P}{E_{photon}}$$

Which in the case of the single pixel it becomes:

$$n_{\text{photon}} = \frac{P_{\text{pyxel}}}{E_{\text{photon}}} = \Phi_{\text{pyxel}} * \frac{\text{Area}}{E_{\text{photon}}}$$

Wavelength can be considered in two ways:

- As a single average wavelength, corresponding to the peak of solar radiation.
- A different wavelength for each pixel, treating each part of the image as a black body emitting peak light radiation relative to the color of the pixel itself.

In the first case, a peak radiation of about:

$$\lambda_{a_{SUN}} = 500 \text{ [nm]}$$

In the second case, the concept of Photopic Response is introduced, dependent on the wavelength of the light radiation, and this value is compared with the relative Luminance associated with the single pixel, thus managing to obtain through a logarithmic function the wavelength associated with each different light radiation that reaches the optical sensor.

### 6.1.6 Output and Results

Once the map of photons reaching the sensor in each of its pixels has been obtained, the preliminary radiometric analysis can be considered concluded. Note that the values obtained represent an extremely ideal case, in which no real effect is considered both at the level of sensor geometry and about disturbances and losses due to environmental or physical factors of the sensor itself. The next chapter describes the process of obtaining a simulated image in output from the number values of photons obtained now.

Below are the results obtained so far; for the graphic representation a heatmap was used, which allows to appreciate the differences, at the level of luminous flux, between the different parts of the simulated scene.

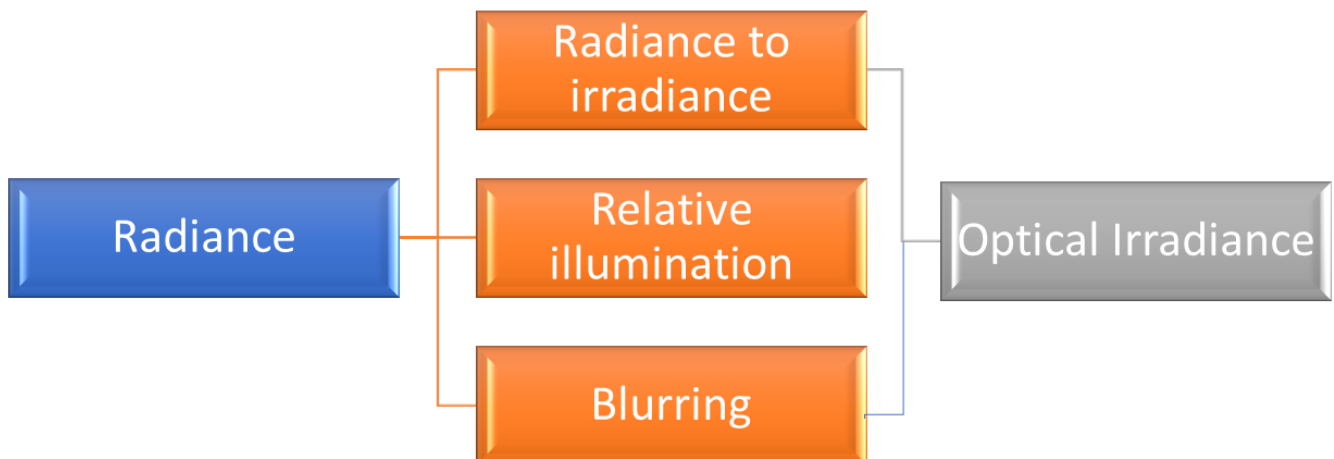
In particular, the results are related to a test simulation, composed only of the Space Rider model, as realistic as possible in all its parts and with the correct materials applied to the surfaces, and the light radiation model described above. Comparing the maps obtained with the simulated scene in Blender, we notice a partial loss of information that results in a lower resolution and sharpness of the shapes, due to the process of linearization and adaptation to the RGB linear color field.

## 6.2 OPTICS

Optics represents the characteristics of the camera sensor. In this part scene radiance data is converted into an irradiance image (photons/s/nm/m<sup>2</sup>) at the sensor surface. The conversion from radiance to irradiance is determined by the properties of the optics, described in (24) and (26), which gather the diverging rays from a point in the scene and focus them onto the image sensor itself.

Irradiance image is called optical irradiance and it can be obtained through the following steps:

- Conversion from Radiance to Ideal optical irradiance.
- Relative Illumination calculation due to geometry of single pixel.
- Blurring effect, calculated with different model.



### 6.2.1 OPTICS: Irradiance

Using the camera equation, we are able to convert scene radiance to an ideal optical irradiance value, dependent to vertical and longitudinal coordinates  $(x, y)$  and to wavelength  $(\lambda)$ . The equation is:

$$I(x, y, \lambda) = \frac{\pi * T(\lambda)}{4 * \left(\frac{f}{\#}\right)} * L_{scene} \left(\frac{x}{m}, \frac{y}{m}, \lambda\right)$$

Where:

- $T(\lambda)$  is the lens transmissivity.
- $f$  is the focal length.
- $\#$  is the effective aperture of the camera.
- $\frac{f}{\#}$  is F-stop value.
- $m$  is lens magnification.

This law is totally accurate for the pixels near the center of the camera, which receive light in a perpendicular way; for other location, far from the center, an off-axis correction must be used.

### 6.2.2 OPTICS: Relative illumination

The previously obtained value must be mitigated taking into account the position of the individual pixels with respect to the central axis of the sensor. In fact, a shading effect is created due to the imperfect perpendicularity of the light on the surface of the pixels outside the main axis of illumination. Shading is described by the following law, called cosine-fourth law:

$$R(x, y, \lambda) = \cos^4(\theta) \approx \left(\frac{d}{S}\right)^4$$

Where:

- $S$  is the image field height (distance from on-axis).
- $d$  is the distance from the lens from image plane.

You then get the map of the Relative Illumination. This map does not consider the irregularities of the chamber, the lenses and the various possible effects of geometric distortion.

### 6.2.3 OPTICS: Blurring

The irradiance image cannot be a perfect replica of the scene radiance. Imperfections in the lens material or shape, as well as fundamental physical limitations (diffraction), limit the precision of the reproduction. Several types of blurring effects can be implemented in order to represent the imperfections of camera sensors.

Below are 3 different models of different complexity:

- Diffraction-limited optics: related only to f-number of the lens and wavelength of light radiation. This model defines the blurring transfer function.
- Shift-invariant image formation: for isoplanatic model (shifting of a point does not affect irradiance distribution); from Diffraction model we obtain real Irradiance on the sensor.
- Shift-variant image formation, raytracing: complete and complex model.

#### 6.2.3.1 *Blurring: Diffraction-limited optics*

The model describes the blurring caused by a perfect lens with a finite aperture. The Fourier transform is used to describe the point spread function (PSF), which determines how the diffraction-limited system behaves.

Then the optical transfer function is obtained, defined as follow:

$$OTF = \begin{cases} \frac{2}{\pi} a \cos(\rho) - (\rho\sqrt{1 + \rho^2}) & \text{if } \rho < 1 \\ 0 & \text{if } \rho \geq 1 \end{cases}$$

Where:

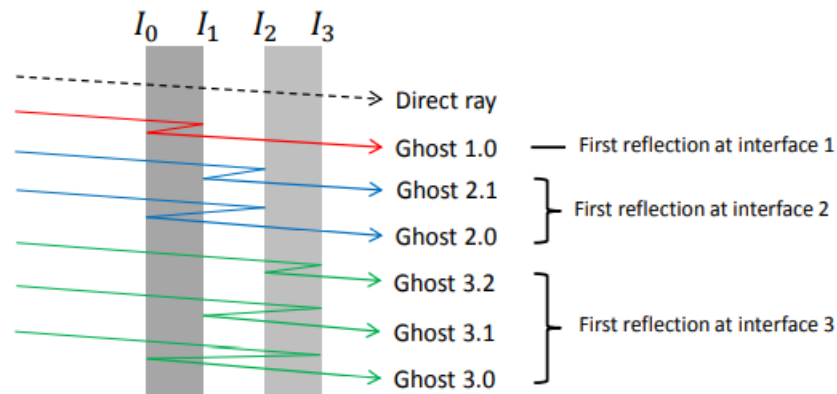
- $\rho = f \left( \frac{A}{d} \right)$  is the normalized frequency.
  - $f = \text{frequency} \left[ \frac{\text{cycles}}{m} \right]$
  - $A = \text{aperture diameter} [m]$
  - $\lambda = \text{wavelength} [nm]$
  - $d = \text{distance between lens aperture and detector} [m]$

This model define how the real Irradiance map is different with respect to the ideal one.

#### 6.2.3.2 *Blurring: Lens Flare from the Sun*

Physically optics segment of a camera is composed of many different lenses. When light passes through materials with different refractive index a part of light will be refracted and another part will be reflected, producing a stray light. Stray light represents the losses of the optical segment, in fact part of the reflected light will be absorbed by lenses or will exit through camera aperture.

Instead, part of reflected light will be reflected back to the sensor surface, generating on the sensor “ghost” images, proportional to number of lenses and distance between them. The phenomena are schematically represented below (Credit (26)).



### 6.2.3.3 Blurring: Shift-Invariant image formation

To greatly simplify the calculation of the blurring effects on the irradiance values obtained by the camera sensor, the isoplanatic system can be used, for which it is possible to use a shift-invariant model.

An isoplanatic patch in an optical system is a region in which the aberrations are constant; experimentally, a patch is isoplanatic if translation of a point in the object plane causes no change in the irradiance distribution of the PSF except its location in the image plane.

The image formation and the PSF are converted in the spatial frequency domain, thanks to Fourier transformation the spatial distribution of irradiance image is obtained. The formula is defined as follow:

$$I_{image}(x, y, \lambda) = FT^{-1}\{OTF * FT\{I_{ideal}(x, y, \lambda)\}\}$$

Where:

- FT is the Fourier transform operator
- OTF is defined in previous slide

For more complex systems, in which there is non-constant aberration (anisoplanatic), additional considerations must be carried out and consequently the calculation of the irradiance map is greatly complicated.

## 6.3 SENSOR

The role of the sensor inside an optical camera is to transform the optical irradiance map, coming from the optical section of the camera, into a two-dimensional array of voltage samples, one sample for every pixel.

The part dedicated to this transformation is a device called photodetector, usually made of silicon, whose response to the photon beam is linear and consequently the current value for each pixel increases as the number of incident photons increases.

### 6.3.1 SENSOR: Photodiode

Before being able to evaluate all the effects related to the sensor of a digital camera, it is necessary to define its internal structure and in particular to understand the operation of its photodetector. A single pixel is usually formed by a series of Microlenses and filters that convey the light beam, composed of photons, onto a photodiode, which is responsible for the acquisition of light. The photodiode is therefore the element underlying pixel functioning and is a semiconductor device with a P-N junction that converts photons (or light) into electric current. The P layer has an abundance of gaps (positive), and the N layer has an abundance of electrons (negative). Photodiodes can be produced from a variety of materials including, but not limited to, silicon, germanium and indium gallium arsenide as described in (27). Each material uses different properties for cost advantages, increased sensitivity, wavelength range, low noise levels, or even response speeds.

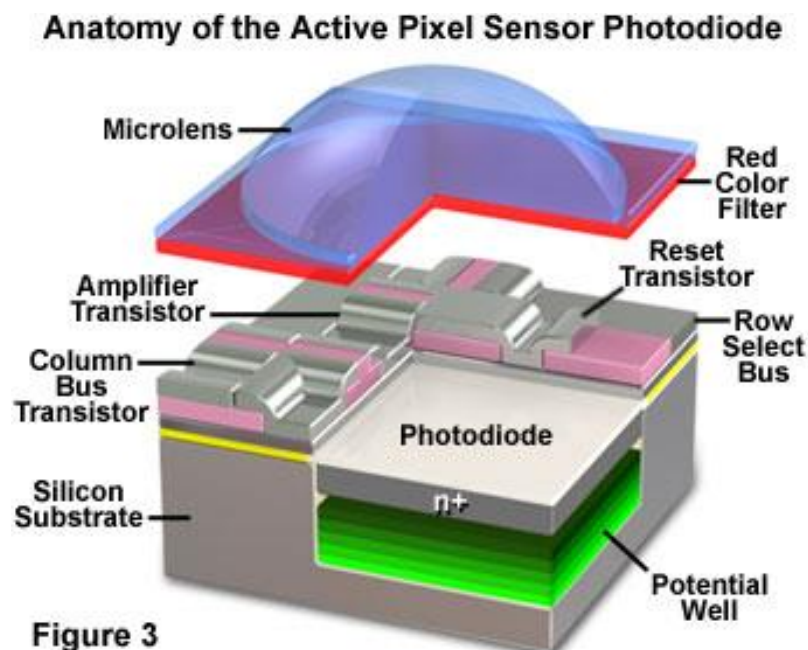


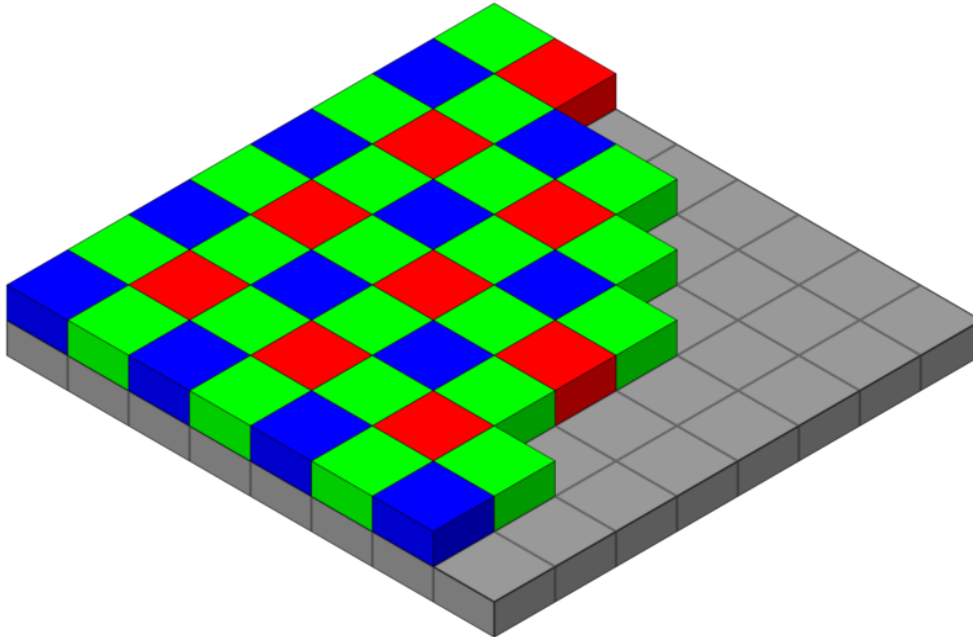
Figure 52: Photodiode model (Credit (27))

On a physical level, a photodiode exploits the photoelectric effect to produce electricity. When a photon of sufficient energy hits the diode, it creates an electron-gap pair. If absorption occurs in the region of emptying of the junction, or at a diffusion distance from it, these vectors are swept away from the junction by the built-in electric field of the emptying region. So, the holes move towards the anode and the electrons towards the cathode and a photocurrent is produced. The total current through the photodiode is the sum of the darkness current (current that is generated in the absence of light) and the photocurrent, so the darkness current must be minimized to maximize the sensitivity of the device. It is also necessary the presence of a transistor and an amplifier for the correct transmission of the electrical signal.

First, for a given spectral distribution, the photocurrent is linearly proportional to the irradiance, calculated as a result of the previous phases.

Each pixel is represented by a CMOS (complementary metal-oxide-semiconductor) sensor which, however, is not formed by a single photodiode, but by four photodiodes associated with filters

based on the three primary colors red, green and blue. A representation of the distribution of photodiodes with different filters is given below.



**Figure 53: pixel photodiodes distribution**

Each photodiode is masked by a red, green, or blue filter in low-end chips, but higher-resolution CMOS devices often use a teal (blue-green) filter instead of one of the green filters. Together, the four elements shown in the figure above comprise the photosensitive portion of the pixel. Analyzing a single portion of four photodiodes, two green filter masks are used because visible light has an average wavelength of 550 nanometers, which is located in the green region. Each pixel element is controlled by a set of three transistors and an amplifier that work together to collect and organize the distribution of optical information. The three transistors then define the current value for each RGB channel, uniquely describing the color of the pixel itself. Alternative solutions imply the presence of overlapping filters or other particular geometries.

### 6.3.2 SENSOR: Numerical Integration

Starting from Signal Current Density Image we can obtain the number of electrons accumulated at each pixel. It is necessary a double integration:

- Spatial: based on pixel dimensions (height and width).
- Temporal: exposure time is considered.

For an ideal sensor, which does not consider noise, imperfections and dissipative effects, the number of electrons per pixel is obtained by the following formula:

$$e_i = T \iint_{\lambda x}^i S_i(\lambda) A_i(x) I(\lambda, y) d\lambda dx$$

Where:



- $I(\lambda, x)$  is the irradiance per pixel
- $S_i(\lambda)$  is the sensor spectral quantum efficiency  $\left[\frac{e^-}{\text{photon}}\right]$
- $A_i(x)$  is the aperture function across space  $[m^2]$
- $T$  is the exposure time [s]

We then obtain the value of the current that is generated on the sensor of each pixel, calculated in number of electrons; it is essential to determine the correct exposure time of the sensor in such a way as not to receive too much or too little light, which will therefore imply unacceptable current values.

The value thus obtained takes into account the optical segment and the effects of the sensor, but does not represent the final result as at this stage the noise effects that are generated on the sensor segment due to unevenness and discontinuity are not yet considered. These effects will be described below.

### 6.3.3 SENSOR: Real effects

The number of electrons per pixel previously calculated represents the ideal number, which considers only the effects of optics seen above. However, it is necessary to evaluate the optical effects of the sensor and individual pixels, in order to have a more accurate estimation of the number of electrons and to obtain the Signal Current Density Image.

The effects to be evaluated are the following:

- Presence of infrared filters.
- Presence of color filters.
- Pixel geometric structure (pixel as a tunnel, with the detector at the bottom surface, in silicon substrate).
- Microlenses array to redirect photons flux across the pixel.
- Materials refractive index (light scattering).

Considering all these effects the optical efficiency of each pixel is obtained.

### 6.3.4 SENSOR: Noise

A further effect that must be considered and that greatly influences the final result of the image obtained by the sensor is the noise, which comes in different forms in the analyzed system and is due to the physical and electrical imperfections present on the sensor.

The main sources of noise, which will be analyzed individually later, are:

- Photon Shot Noise
- Electrical noise at pixel
- Inhomogeneities across the sensor
- Dark currents
- Fill factor
- Fixed pattern non-uniformity
  - Photo response non-uniformity (PRNU)
  - Dark signal non-uniformity (DSNU)

### 6.3.4.1 Photon shot noise

Photon shot noise is a type of disturbance related to the particle nature of photons. It is probabilistic in nature and by means of a Poisson distribution describes the fluctuations in the number of photons reaching the single pixel.

Below is an example of the variation of the encoding of an image as the Photon shot noise changes. The higher the noise value, the lower the comprehensibility of the image.

### 6.3.4.2 Dark current

The dark current is a small current that is generated due to thermal fluctuations in the material of which the pixel is composed. The current that is generated increases the number of electrons accumulated on the sensor, going to change the value digitized by the converter and consequently going to change, albeit slightly, the color of the pixel concerned.

The fluctuation current value is related to the exposure time, the material of which the sensor is composed and its temperature. Optical sensors must therefore be kept at very low temperatures to mitigate this disturbance.

It is called dark current as it is independent of the incidence of light on the sensor and occurs even in complete dark conditions.

It can be modelled as follow:

$$I_{dark}(T) = b * I_d * T^{\frac{3}{2}} * e^{-\frac{E(T)}{2*k*T}}$$

Where:

$$E_{Si}(T) = 1,1557 - \left[ \frac{7,021 * 10^{-4} * T^2}{1108 + T} \right]$$

$$b = 411915,824603584 \quad \text{scale factor}$$

$$I_d = \text{dark current at } 300 \text{ [}^\circ\text{K]};$$

$$k = \text{Boltzmann constant};$$

$$E(T) = \text{Bandgap [eV] at } T$$

### 6.3.5 SENSOR: Pixel fill factor

In some detectors, only part of the area of a pixel is capable of detecting the photons that call on it. The fractional area is the fill factor: an ideal pixel has a fill factor of 1, resulting in every photon that is incident on it will have the chance to be converted into a photoelectron.

Real pixels instead have some parts dedicated to circuitry that reduce the photosensitive area, with the result of a fill factor between 0 and 1.

## Fraction of pixel area that integrates incoming light.

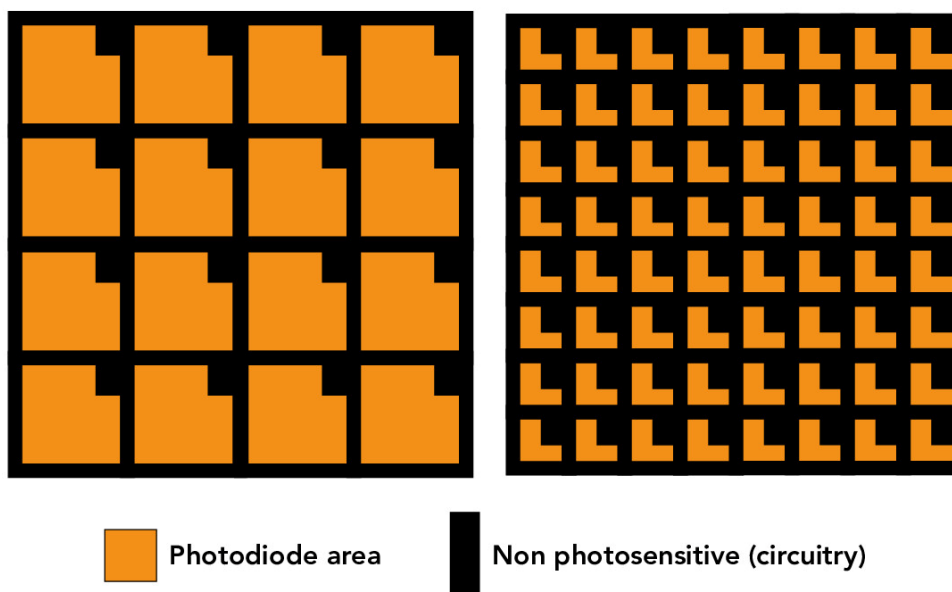


Figure 54: Pixel Fill Factor (Credit (28))

### 6.3.6 SENSOR: Radiative Environment

Finally, it is necessary to evaluate the signal losses and instabilities due to the external environment in which the sensor is immersed; the environment characterizing the orbit on which Space Rider moves, called radiative environment, is therefore considered. The radiation environment in space consists mainly of high-energy protons, electrons, heavy ions and photons. These particles are in continuous motion and their continuous bombardment of an optical sensor by causes a gradual accumulation of damage in the semiconductor lattice that reduces the efficiency of charge transfer and affects the behavior of the pixels, with the risk of obtaining unusable images.

The effects of damage due to the radiative environment increase with increasing duration of a space mission, but they are quite small and in the first approximation can be considered constant. On short time scales, high-energy particles can ionize the semiconductor and deposit electrons on the wells of the pixels. Transient events induced by these impacts may appear as bright isolated pixels, or as short trails if the particle passes through several pixels with a low angle of incidence.

Charged particles in the atmosphere move following a spherical distribution, but free particles with different paths are also present. At the level of the noise model, the number of charged particles (protons, electrons, ions) that hit the sensor from the radiative environment is defined as:

$$N = F * W * H * A_{pixel} * t_{exposure}$$

Where:

- $F$  is the specific flux of particles per unit area.
- $W$  and  $H$  are the dimension of the sensor itself.
- $A_{pixel}$  is the section of the pixel.
- $t_{exposure}$  is the time of exposure.

The obtained value must be added to the previous calculations.

## 6.4 CONVERTER

Once all the effects related to the sensor have been considered, both as regards the acquisition of the light beam and the numerous disturbances present, it is necessary to convert the current value present at the tail of each pixel into a digital value that can be interpreted and used by a computer. This operation is carried out by an analog-to-digital converter (ADC) which is an electronic circuit capable of converting an analog signal with continuous trend, such as an electrical voltage in our case, into a series of discrete values. Each converter is defined by an amplification parameter called gain. In the case where the input parameter is defined by a voltage whose electron flow is known, the corresponding digital discrete number is obtained by the following formula:

$$DN = \text{floor} \left( \frac{N_e}{G} \right)$$

Where:

- $N_e$  is the number of electrons accumulated on the pixel.
- $G$  is the gain of the converter.
- *floor* indicate the rounding in order to have an integer value.

For ease of use and calculation the value is then normalized by the system to obtain a number between 0 and 1, with the following formula:

$$DN_N = \frac{DN}{2^{bits}}$$

Where *bits* indicates the number of bits reserved for the allocation of digital information.

A two-dimensional map of the sensor size is obtained, which contains digital values that represent the information acquired by the sensor from the scene.

## 6.5 PROCESSOR and DISPLAY

Once you reach the sensor you have a two-dimensional map that contains the information of the scene; it is therefore necessary to report the data obtained in a format that is easy to understand and visualize. These tasks are entrusted to a processor and a display, which through the use of mathematical functions and conversions, can transform a matrix of numbers into an image that can be viewed on a device.

In particular, the processor will transform the digital matrix into a 3D space containing the values of the RGB channels of the simulated image and the display will allow you to interpret these values in a color space and display them on the screen. The process behind these steps is very broad and complex and will likely involve future work.

## 7 Photometric and Radiometric Analysis

Starting from the simulation test environment in Blender, containing only the SR model and the sunlight reproduced in a physically accurate way, a rendering is obtained to be able to evaluate the lighting conditions of the scene. Once the boundary conditions are fixed (position and direction of the sun's rays, exposure of the scene, graphic engine used) the rendering of the scene takes place, which produces an image of what is seen and captured by the optical sensor. This image is saved in a specific format and then through analysis and mathematical steps described in previous chapters it is possible to obtain radiometric and photometric information relating to the simulated scene. The tool created for the realization of the analyzes can be used both with a Windows operating system and with Linux, the codes and procedures will differ slightly, but you will get the same result.

### 7.1 Simulator description

The entire project represents the extension of a previously created simulation environment, which has been modified and improved so that radiometric analysis can be carried out on the scene. The simulator is mainly developed in Blender, in which using a python script are added the models related to the bodies necessary for the creation of the scene, contained in external files. The spatial coordinates with which SpaceRider must move in reference to the Earth are also described and implemented. These coordinates are enclosed in .csv files (comma separated values) and are provided by a research group that has carried out navigation studies on the STK software. In input are also made the 3D models of Space Rider, deriving from an external CAD, and the model of the globe, from a Blender file. There is also a model of clouds and atmosphere to make the visualization of the entire scene better. The image below schematically represents the logical process carried out by the simulator.

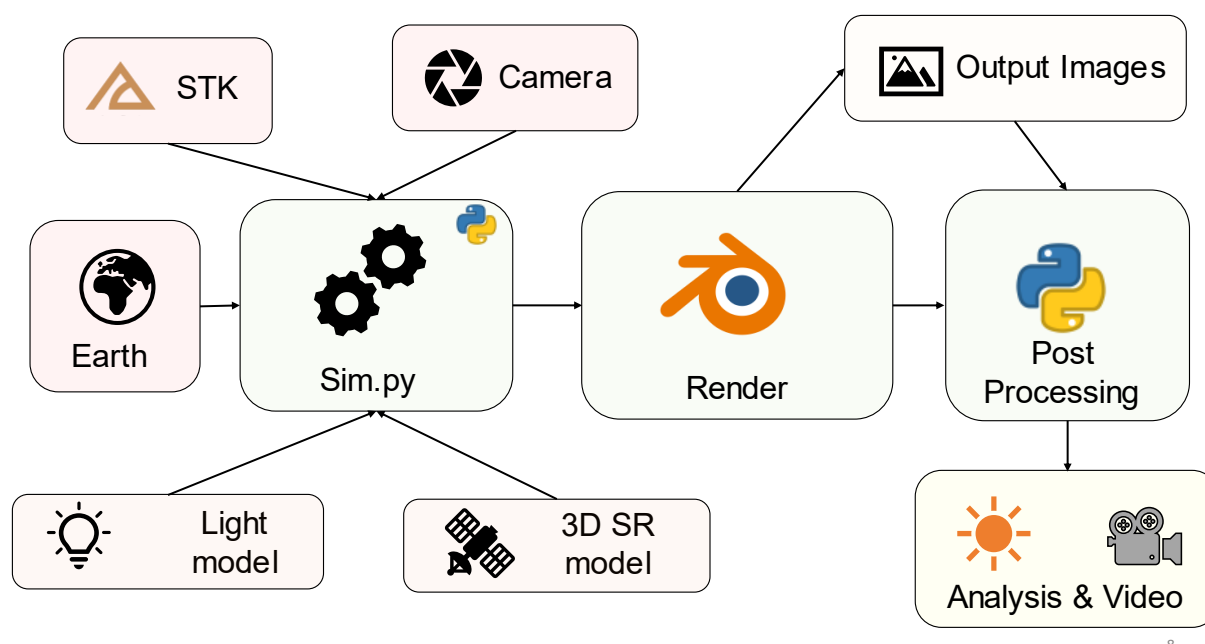


Figure 55: Simulator Workflow

The first *script Sim.py* takes all the necessary inputs and allows Blender to create the simulation environment, which is displayed in the graphical interface of the program. In addition, a function has been developed that allows you to take random coordinates from the relative file and save the corresponding images obtained from the simulation in a folder called "Results", which will later represent the input folder for the Post Processing file.

The *Sim.py* file must be opened in the Blender python interface and must be launched in order to create the simulator.

Downstream of the first script there is therefore the Blender interface that allows you to graphically manage the simulated scene. Once all the Blender processes have been completed, you can then use the Post Processing file, which will allow you to perform the analyzes and obtain the desired outputs.

### 7.1.1 PostProcessing.py

The processes necessary for the use and analysis of the images obtained through Blender is called *PostProcessing.py* and must be used after finishing all the processes described by *Sim.py*.

The file contains a loop that analyzes all the images in the Results folder individually. Images are saved in .png format, for the reasons described in Section 5, and contain pixels initially encoded in the sRGB color space.

The code transforms each image into an array that contains the RGB values for each individual pixel; exploiting the potential of the python *NumPy* library it is possible to perform mathematical operations between matrices.

The pixel values in sRGB are then transformed into a linear RGB color space, from which, through the formula described in Section 6.1.3, the relative Luminance value is obtained for each pixel, thus obtaining a two-dimensional matrix of dimensions equal to the size of the sensor in terms of pixels. Once the Relative Luminance has been obtained, always following the processes described in Section 6, we move on to the Luminous Flux value that affects each pixel and starting from that we obtain the values of photons per pixel and consequently the number of electrons.

The output of the code is represented by heatmaps that describe, using a color scale, the trend of the radiometric quantities analyzed. In a future implementation it will be possible to regain a color image from radiometric values.

The following schematically represents the workflow of the Post Processing file.

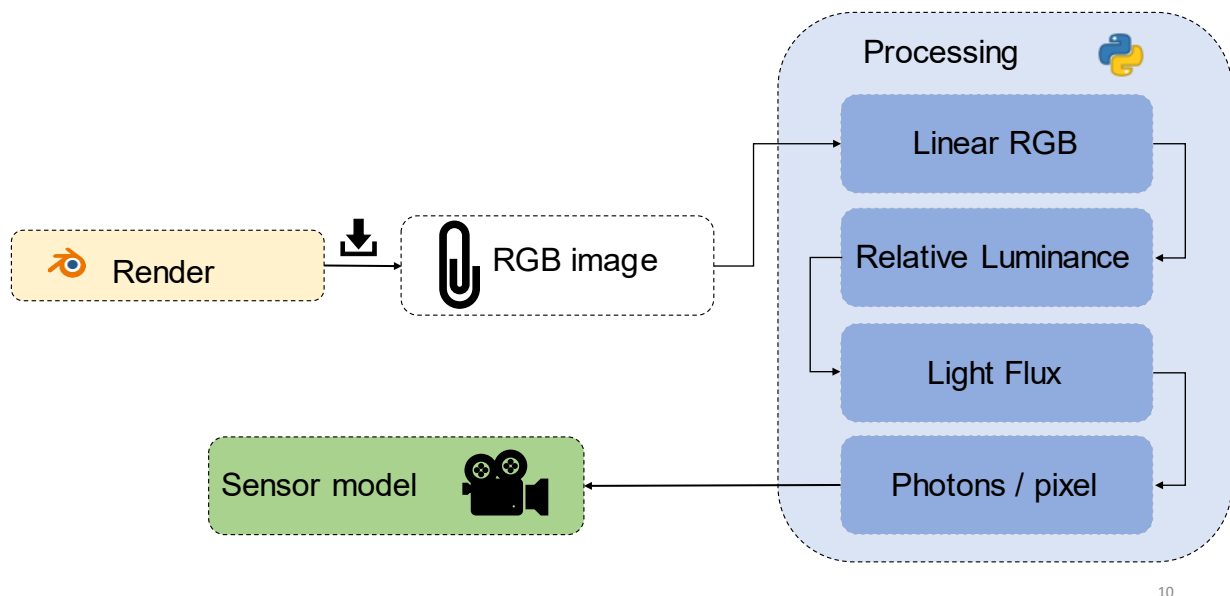


Figure 56: Post Processing Workflow

From the results obtained from The Post Processing it is therefore possible to define the parameters of the optical sensor that will be used on the satellite, in order to obtain an image with the desired characteristics and that is functional to the achievement of the mission objectives.

The process of transformation of the matrices is linear and consequently is proportional and related to the color of each pixel, for this reason it was necessary to develop a tool that worked in the correct conditions of exposure and lighting of the scene a priori.

The analysis provided by Post Processing also allows the study of the docking conditions of SROC on the main body of SpaceRider; knowing the lighting of the approach bay it is possible to understand any critical points and identify the need for use and the position of any thermal markers, to facilitate the final stages of the maneuver.

As will be described in the section on the presentation of the results, many rendered images are devoid of information because they represent SR in its eclipse phase, in which it is not directly affected by sunlight. This is because the process of generating the rendered images is completely random along SpaceRider's orbit around the Earth. Artificial intelligence tools are able to recognize the images suitable for an analysis from those with little information available and can therefore significantly streamline the workload of the radiometric analysis tool.

### 7.1.2 Absolute Physical parameters calculation

In Section 6, studying the mathematical model of the optical sensor described a double integration, spatial and temporal, to determine the absolute radiometric quantities, without being tied to the unit of Area and the unit of time.

Moving on to the present case, spatial integration is carried out by multiplying the relative value by the actual area of the single pixel, which in the case of the sensor used is defined as a perfect square whose area results:

$$Area_{pixel} = 3.75 * 3.75 [\mu m^2]$$

The time factor of the integration is instead linked to the exposure time of the camera on the scene and indicates the period of time in which the camera shutter remains open allowing the passage of light inside it, so that it reaches the film of the optical sensor, with a consequent accumulation of charge. Please note that the exposure value is calculated as:

$$EV = \log_2 \left( \frac{A^2}{T} \right)$$

Since the optical sensor chosen in this simulation is:

$$Aperture = 4 [f_{STOP}]$$

An exposure time value of:

$$T_{Exp} \approx \frac{1}{2000} [s]$$

It is then possible to obtain an absolute value of photons per single pixel.

## 7.2 Blender and Post Processing Results

In this section, both mathematically and graphically, the results obtained after the radiometric analysis carried out on the images obtained as output of the Blender simulator are reported.

As described, the images are randomly generated by the code and represent Space Rider throughout its orbit, in lighting conditions that vary from image to image.

A focus on the bay approach is then presented, which can be seen as a starting point for further ore specific analysis.

The results are reported by comparing the image that is produced by Blender, with the two-dimensional map relating to the Luminous Flux per single pixel and the consequent number of photons that hit the pixel itself.

It was decided to use images obtained randomly by means of python code, to be able to comment on the results related to good and less good lighting conditions.

The images are presented in two formats:

1. The sRGB format, resulting from blender's rendering of the scene, which is the starting point of the analysis.
2. A heatmap that represents the luminous flux value coming out of each pixel, measured in Watts per square meter. It should be borne in mind that the maximum permissible value is that provided by a perfectly reflective surface and is therefore equal to the value of the average solar flux reaching the upper layers of the atmosphere, equal to:

$$Flux_{MAX} = 1367 \left[ \frac{W}{m^2} \right]$$

This value corresponds to the pure white color on the heatmap, while the minimum value, corresponding to a completely absorbent surface, is related to the black color. In an ideal



case, with an image format that supports a very wide dynamic range, the limit colors should not appear, as you can appreciate even the smallest variations in brightness and solar flux. With the format used, due to the problems related to the calibration of the exposure value presented in Section 4.3, it is possible to obtain limit values, implying areas where it is not possible to carry out the radiometric analysis correctly. Fortunately, these situations occur only in a condition of perfect perpendicularity of the rays of solar radiation on the most reflective surfaces of SpaceRider. None of the random images simulated and analyzed presented this problem.

Below are the results obtained from a simulation in which the SROC CubeSat observed SpaceRider from a distance of about 200 [m], with the optical parameters of the camera described above with a resolution of 600\*600 pixels in order to reduce rendering time, so that a complete view of the satellite can be obtained. We analyze both situations of good lighting that offer well-defined results, and images in which the boundary conditions do not allow a good success and significance of the analysis.

### 7.2.1 Results: Bad Lighting Conditions

This section contains cases in which the lighting conditions do not allow to obtain sufficient information of the SR conditions, not putting the optical sensor in a position to analyze the scene profitably.

#### 1. Bad lighting: Too dark

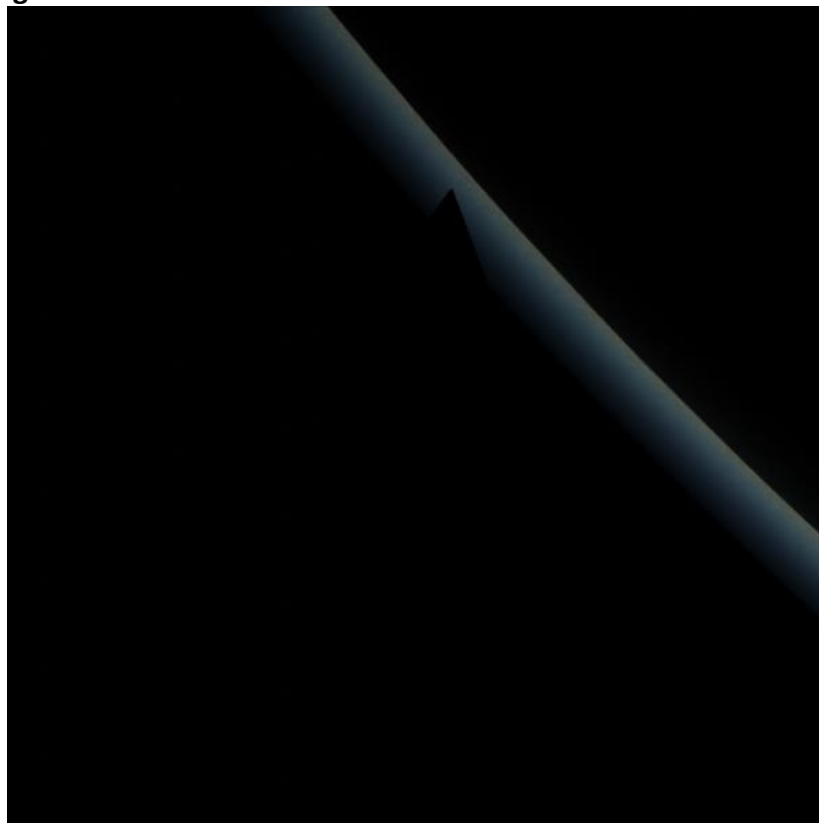
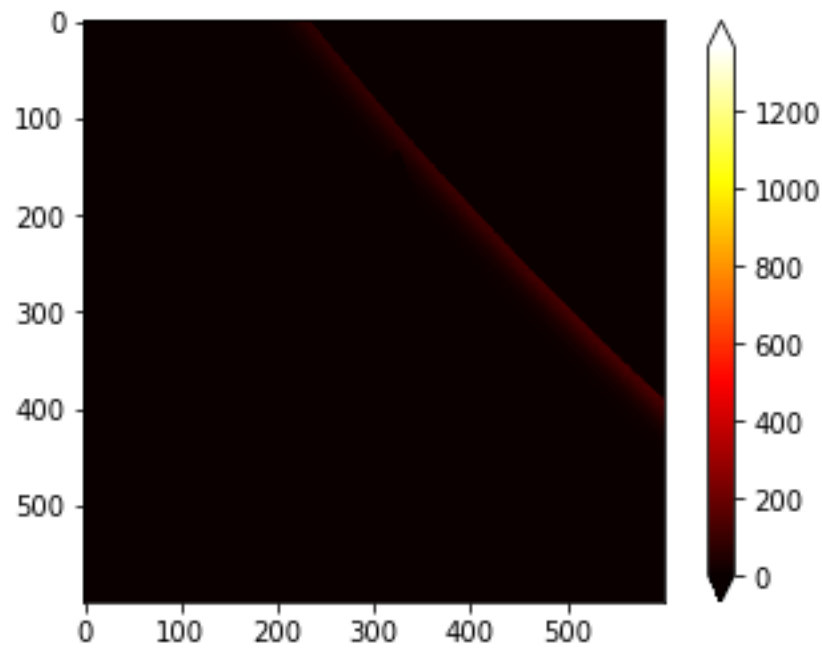


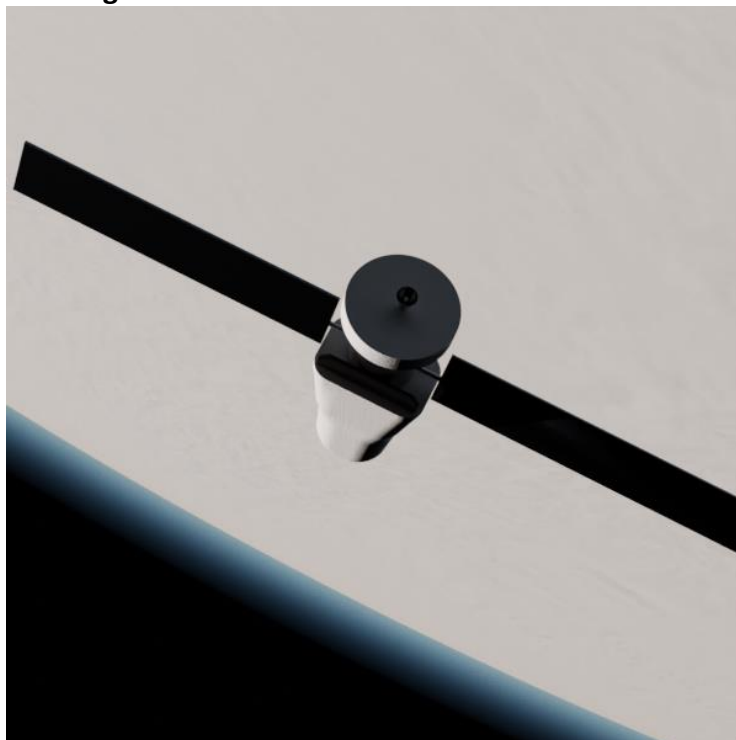
Figure 57: SR Eclipse



**Figure 58: SR Eclipse Flux heatmap**

Figures 57 and Figure 58 show the rendering result and the relative heatmap of the luminous flux of a moment in SR's orbit when the satellite is in partial eclipse behind planet Earth. The results are obviously represented by a very dark image, from which the silhouette of SpaceRider is difficult to distinguish. Analyzing the Flow map, which will be proportional to the number of photons per pixel, it is noted that relevant information is obtained only with regard to the atmospheric layer, because of light scattering.

## 2. Bad Lighting: Too Bright



**Figure 59: SR Too Bright**

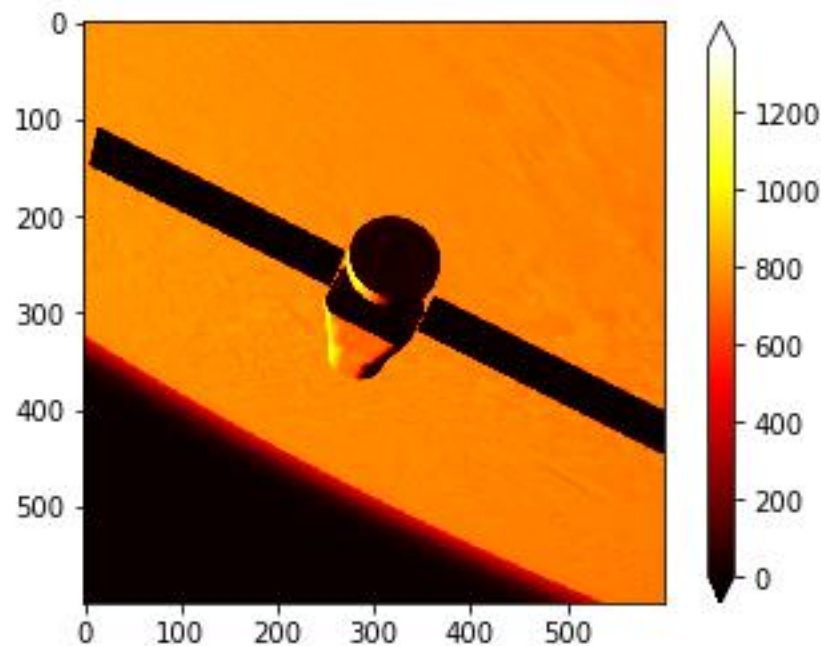


Figure 60: SR too bright heatmap

Figures 59 and Figure 60 present a situation in which SR is captured by the optical sensor at a small azimuth angle with the Earth and its atmosphere as a background. Although the scene seems at first glance well-lit and clearly visible, it represents an unfavorable condition for the purposes of observation, as the parts of the tail cone and solar panels are very dark and in shadow, while the main body of the satellite tends to merge with the atmospheric surface of the planet, showing a very similar color and consequently a comparable reflectivity. In fact, a confusion between the edges of the spacecraft and the background itself is possible, going to invalidate the radiometric analysis. Despite these problems, it can be noted that the results provide a value of about 800-1000 [W / m<sup>2</sup>] of luminous flux coming out of the main body of the satellite, a value that is likely and accurate compared to a real scene.

### 7.2.2 Results: Good Lighting Condition

In this section, on the other hand, two different situations in which radiometric analysis has been successfully applied, obtaining significant and useful values for further development. The focus was to obtain detailed and differentiated information with respect to the background of the image scene that contains the satellite, paying particular attention to the difference in behavior of the different materials that make up SpaceRider, even with a poorly detailed version of the satellite such as the one used in these simulations.

#### 1. Good Lighting: SpaceRider Only

A simulation is represented in which the only object visible in the scene is SpaceRider, with a black background due to the deep space outside the atmosphere.



Figure 61: SpaceRider

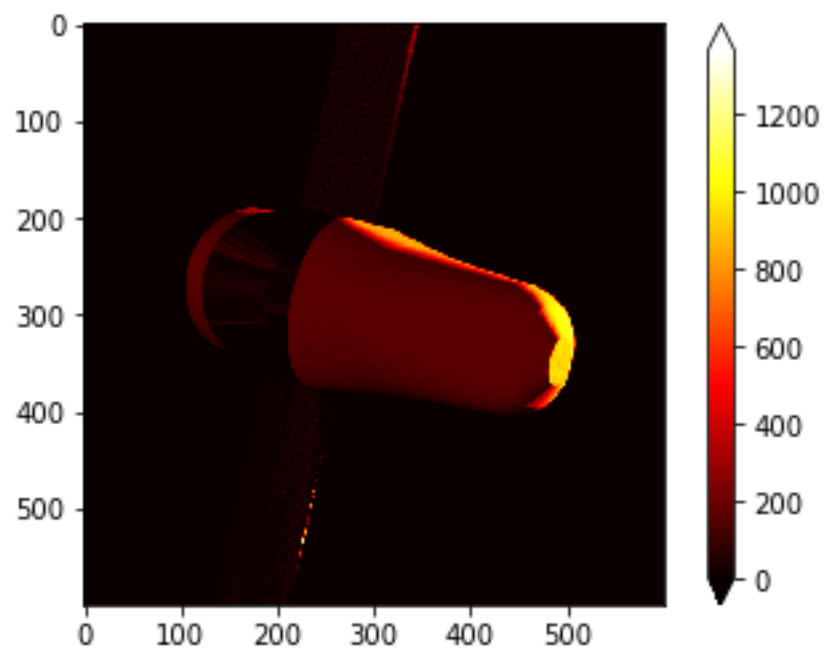


Figure 62: SpaceRider heatmap

Figures 61 and Figure 62 have achieved good results. SpaceRider is immersed in deep space and illuminated in a transversal way; there is no interference with the other bodies of the scene and the entire body of the satellite is well defined and with a good contrast to the background. It is possible to appreciate the difference in illumination between the back and belly, as well as the great differences between the components due to the materials of which they are composed. All this is reflected in the Heatmap, which clearly describes the differences in luminous flux reflected and captured by the sensor. In fact, there is a flow of about 400 [W/m<sup>2</sup>] on the front and about 1000 [W/m<sup>2</sup>] on the back; the solar panels and the details of the tail cone are much less reflective due to the materials, but there are also small differences on them. With more detailed models, the results can only be clearer and more precise.

## 2. Good Lighting: SR and Earth

In the results below, however, a scene that includes both the SpaceRider model and the model of the Earth, clearly visible with its atmosphere and its clouds, is analyzed.



Figure 63: Space Rider over Earth

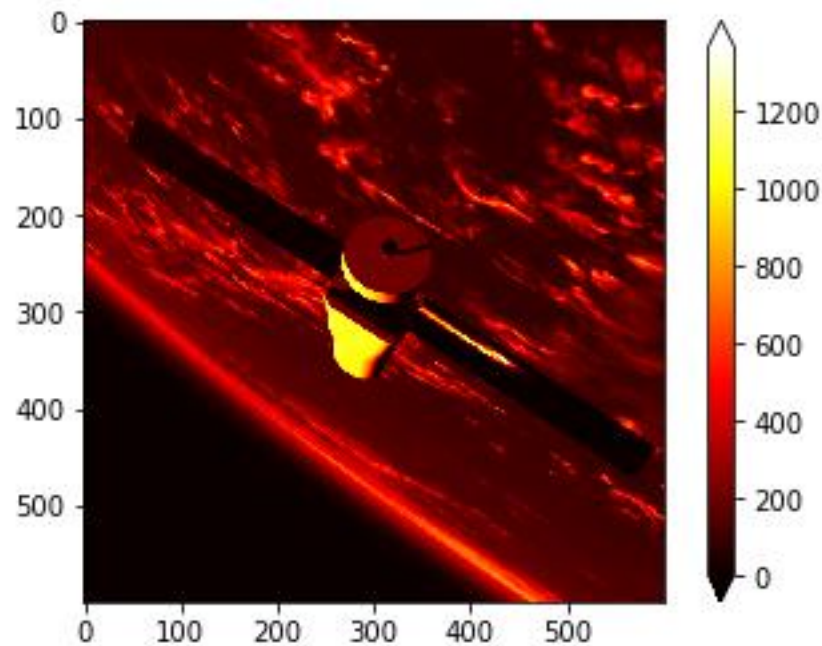


Figure 64: SpaceRider over Earth heatmap

In Figures 63 and Figure 64 a scene very similar to that seen in the previous chapter concerning the bad lighting conditions is represented, but in this case all the objects are clearly visible, illuminated and distinguishable from each other. The result is a clear and easily interpretable image, in which all objects are well defined, thus being able to evaluate the differences both visual and in radiometric terms between satellite, clouds, atmosphere and earth's surface. Also note on the solar panel of the right wing of SR the highly reflective characteristics of the material of which the panels themselves are composed when hit by a perpendicular or almost perpendicular radiation.

A tool was then created that could extract the radiometric quantities from any image provided as input. By following the processes described in Section 6, you can focus on any of the quantities considered. The results are significant, but an increase in image quality allows a greater differentiation between the brightest and darkest parts but going to dilate in a substantial way the production and analysis times of the images themselves.

### 7.3 Case Study: Cargo Bay

The cargo bay represents the focal point of the radiometric analyses to be carried out for the SROC mission, as it represents the physical place where the CubeSat will be contained and consequently the place where the docking maneuver will end at the end of the observation phase. In this regard, using a more detailed model of SR, a first preliminary analysis of the lighting conditions of the bay during the final stages of the aforementioned maneuver was carried out. An elementary model of SROC was created consisting of a parallelepiped of 12 U in size:

$$DIM_{SROC} = 29 [cm] * 20 [cm] * 30 [cm]$$

The SROC model is needed to recreate the shadow that the satellite exerts on SR during re-entry with the Sun behind it.

The results of this analysis will then be necessary to determine the need to use any additional instrumentation in order to safely and effectively terminate the maneuver.

Below is the image obtained through the rendering of Blender and the related heatmaps that represent the distribution of the luminous flux and the number of photons that hit each pixel.

The scene was created with the following settings:

- Solar radiation perpendicular to the plane containing SpaceRider.
- SROC distance from cargo bay of 0.5 [m].
- Camera settings:
  - Focal Length = 13 mm
  - Aperture = 2.8
  - Dimension = 1280\*960 pixels

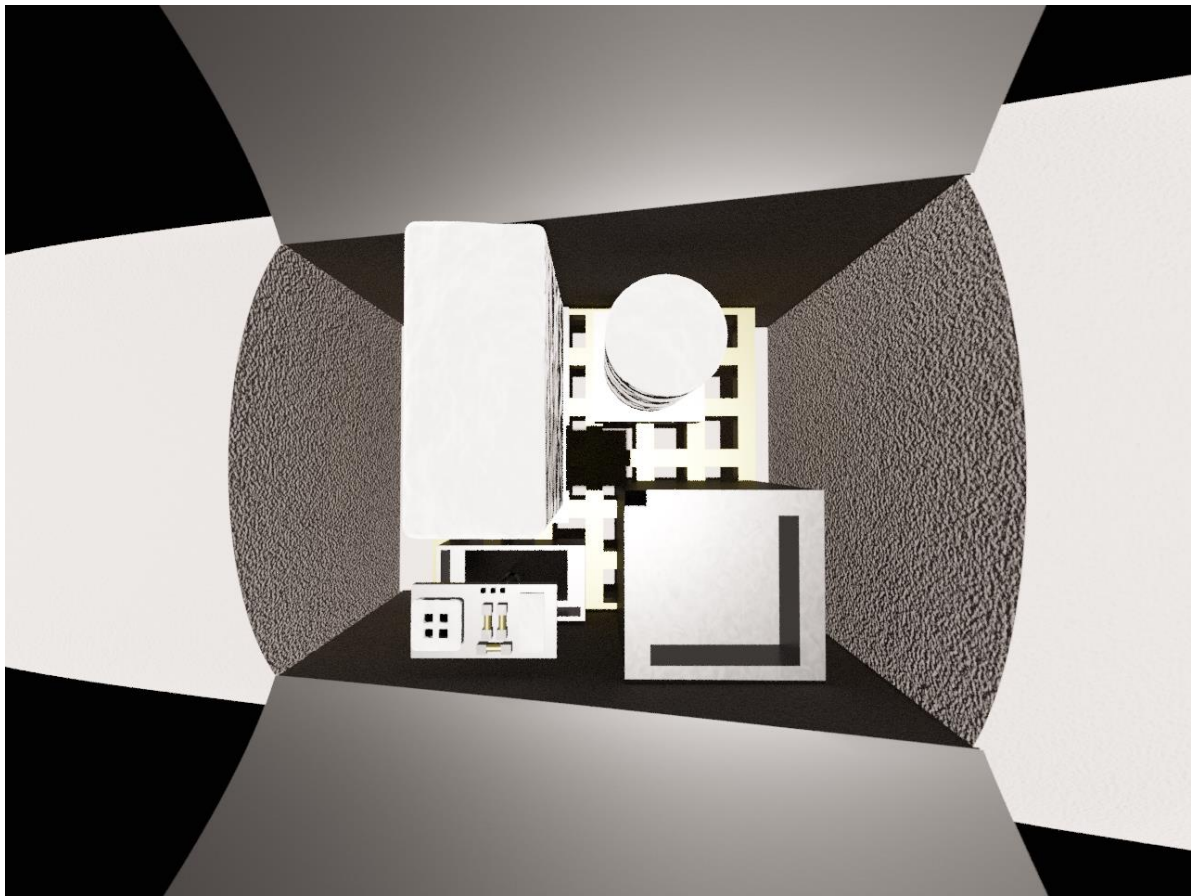


Figure 65: SR Cargo Bay

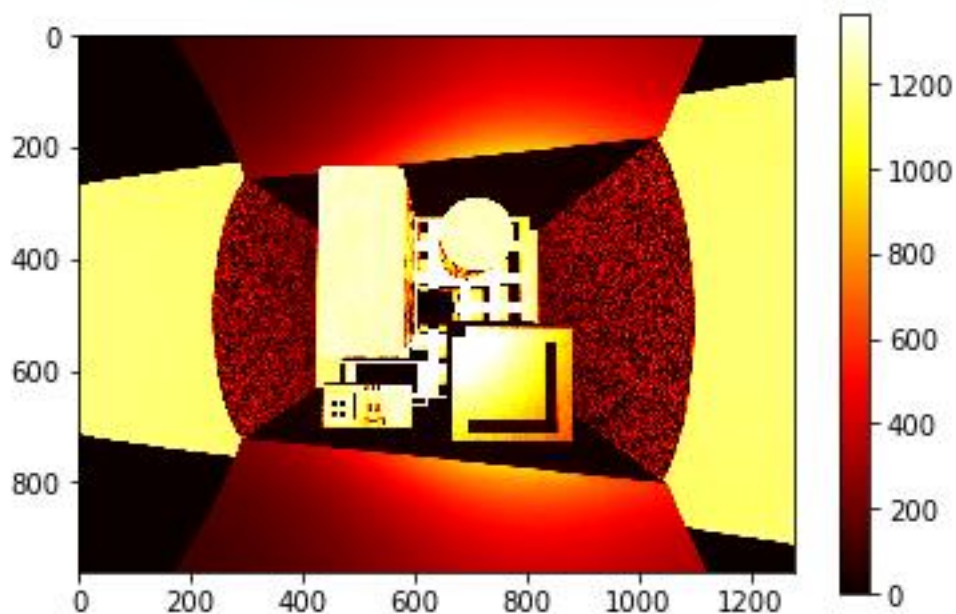


Figure 66: Cargo Bay – Flux

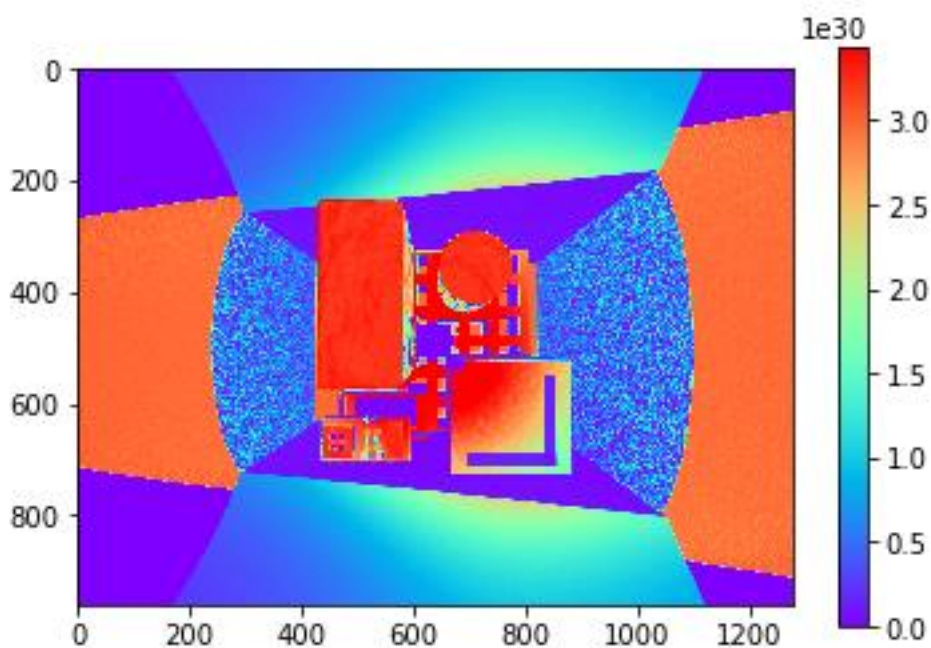


Figure 67: Cargo Bay - Number of Photons

As you can see, there is the shadow of the SROC CubeSat on the bay, but the latter is still clearly visible and well lit. In this first test it was therefore noted that even near the end of the docking maneuver you are in good lighting conditions. Further analysis will be carried out as the angle of the solar radiation changes to better understand the radiometric operation of the conditions during the maneuver itself, facilitating the resolution of any problems.



## 8 Pangu

It is therefore clear that Blender, despite being an extremely performing and ductile instrument, is not designed to carry out radiometric analysis accurately as desired by the project.

As described in section 2.5, PANGU is designed to create and analyse under a graphical aspect space scene. It is in fact very simple to use, as it allows you to use a very user-friendly interface for the definition of the parameters necessary for the creation of the desired scene, but above all, having been created so that it was physically accurate in all its parts for spatial scenes, it can be used to carry out analysis with good precision and accuracy.

Among the different tools available, PANGU allows the visualization of the scene both with a camera in the visible field, and with a simulated LIDAR, in order to graphically display reliefs and altitude of the terrains and objects represented. The combination of these two techniques can be used for example for the study of the surface of an asteroid or to evaluate the possible docking site of a satellite on a planet with its characteristic irregular surface, which can be easily implemented in the program, as will be explained in section 8.1.

Thermal analysis can also be carried out, as the created scene can be viewed using an infrared camera.

Below are the main steps that were followed for the realization of the 3D scene related to the project.

### 8.1 Scene creation

For the creation of the scene the processes described in [2] have been followed, which determine and explain how to import, modify, or create various objects and parameters within PANGU. In particular, for the scene in question, we tried to recreate a simulation environment similar to the one previously seen in Blender. In fact, a model of the globe was created, simplified in this first phase as we will see in paragraph 8.1.1, the SpaceRider model (8.1.2) was imported, an atmospheric model was created (8.1.3) and finally an optical sensor was implemented that simulated the camera present on SROC.

The luminous model has been left by default, as it represents a physically accurate Sun; in order to obtain the correct lighting conditions, it was necessary to establish only the positions of the simulated bodies within the simulation. As in Blender, also in PANGU sunlight is represented by a uniform field of light.

The big difference from Blender, due to the fact that PANGU was designed for simulating spatial scenes, is the absence of problems related to the size of objects. If in fact in the old simulator it was chosen to scale all the objects compared to the real size in such a way that they were smaller and more manageable by the program, both in terms of simulation and in terms of rendering and visualization, in the new simulator all the objects have the real dimensions and no problems of any kind have been found. It is therefore possible to simulate objects with orders of magnitude of difference in size, such as Earth and SR, in the same scene, whether they are positioned relatively close to each other, or whether they are at a huge relative distance between them.

Please note that the following procedures have been implemented in PANGU using a file called *sim\_VIS.sh* that contains code that describes the creation of the scene. This file is launched from the Terminal in the Linux version of the simulator and allows you to view, through the graphical interface of PANGU, the simulated scene.

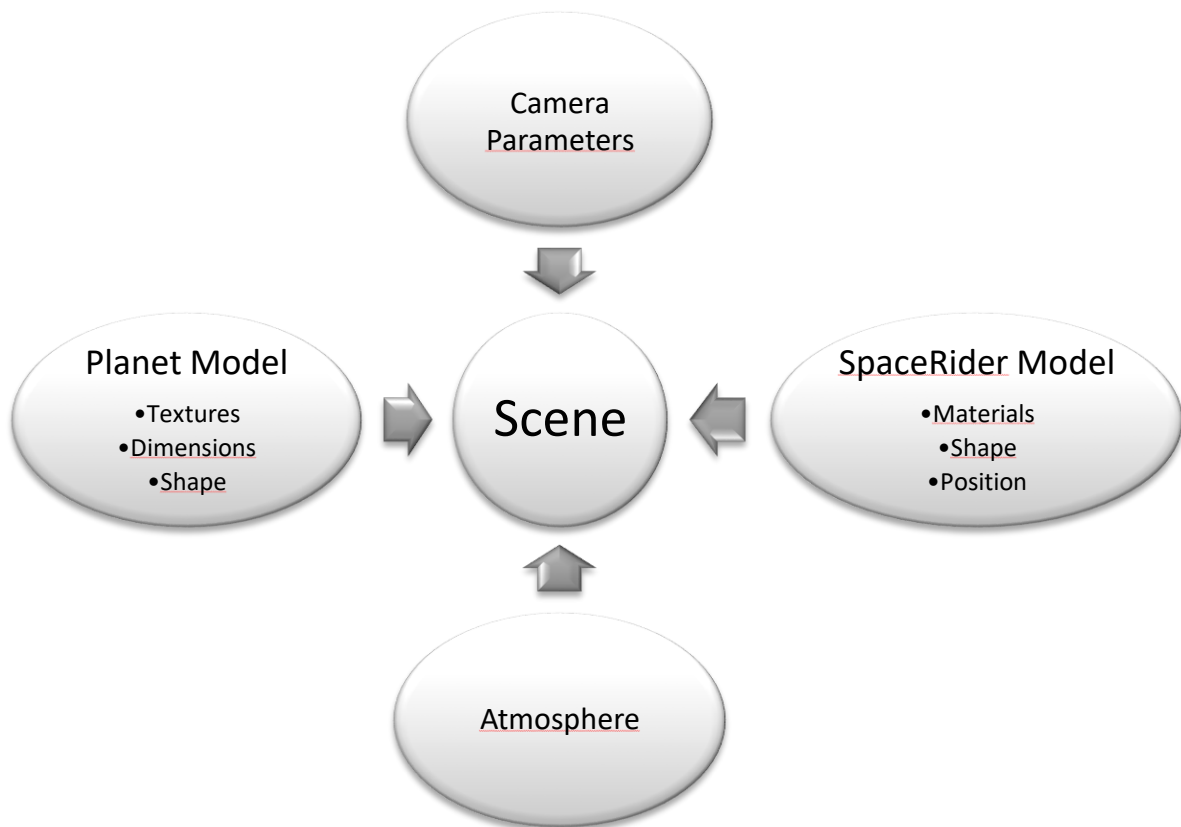


Figure 68: PANGU simulator workflow

In figure 68 the workflow of the code for the creation of the scene is schematically represented, in the following paragraphs the different parts that compose it will be analyzed in detail.

### 8.1.1 Planet Model

The procedures for the realization of models of planets are described in the fourth chapter of (26), called "Whole World Modelling".

This type of model is used in PANGU to simulate large quasi-spherical objects, such as planets, moons or asteroids. In addition to dimensions, you can create models with very accurate shapes and surface finishes. In PANGU this process can be carried out in different ways, chosen according to the needs of the user:

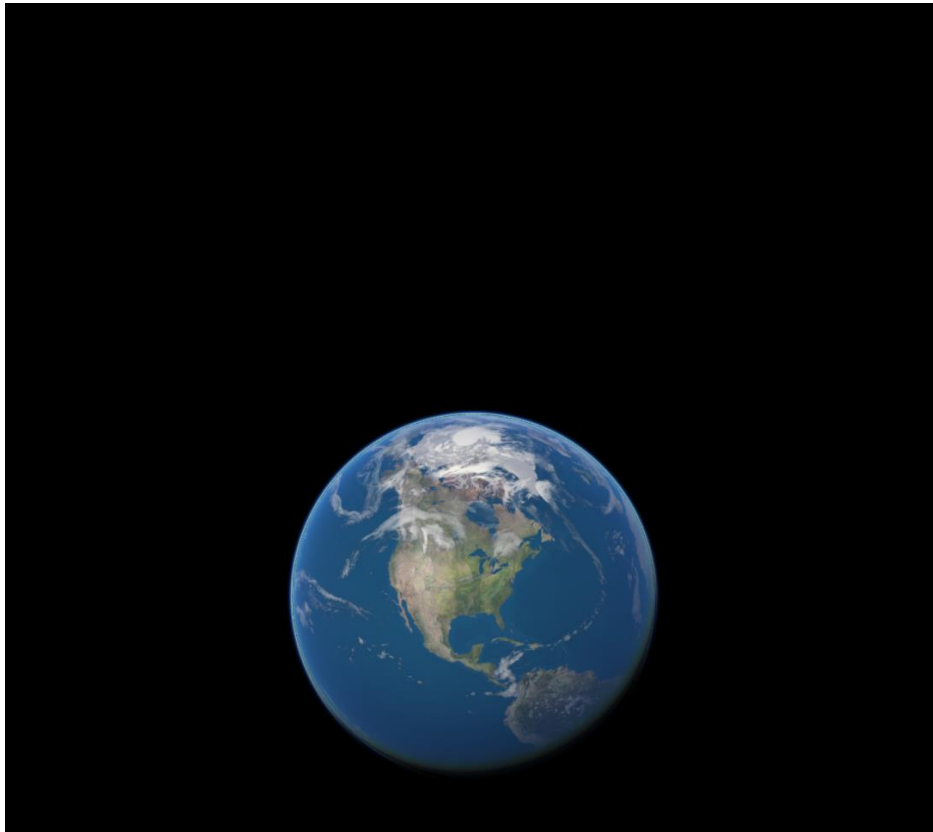
1. A structured oblate spherical model designed to simulate entire worlds by applying textures suitable for a spherical or oblate model.
2. A spherical model at radial height in which the heights of the vertices of the spherical model are modified the height values obtained from a DEM radial height. This type of model can be extended by combining with a DEM-based model to create a high-resolution patch on the worldwide model.
3. An entire model of the planetary globe created by directly projecting a DEM radial height.

Given that the Terra non represents the central focus of the project, but only a part of the surrounding environment in which SpaceRider is immersed, which represents the object of interest of the entire mission, it was decided to use the first model described for the creation of the planet.

In particular, the Earth was created from a sphere centered in the origin of the scene, with a radius equal to the average Earth radius.

$$R_{Earth} = \sqrt{\frac{A_{Earth}}{4 * \pi}} = 6371 [km]$$

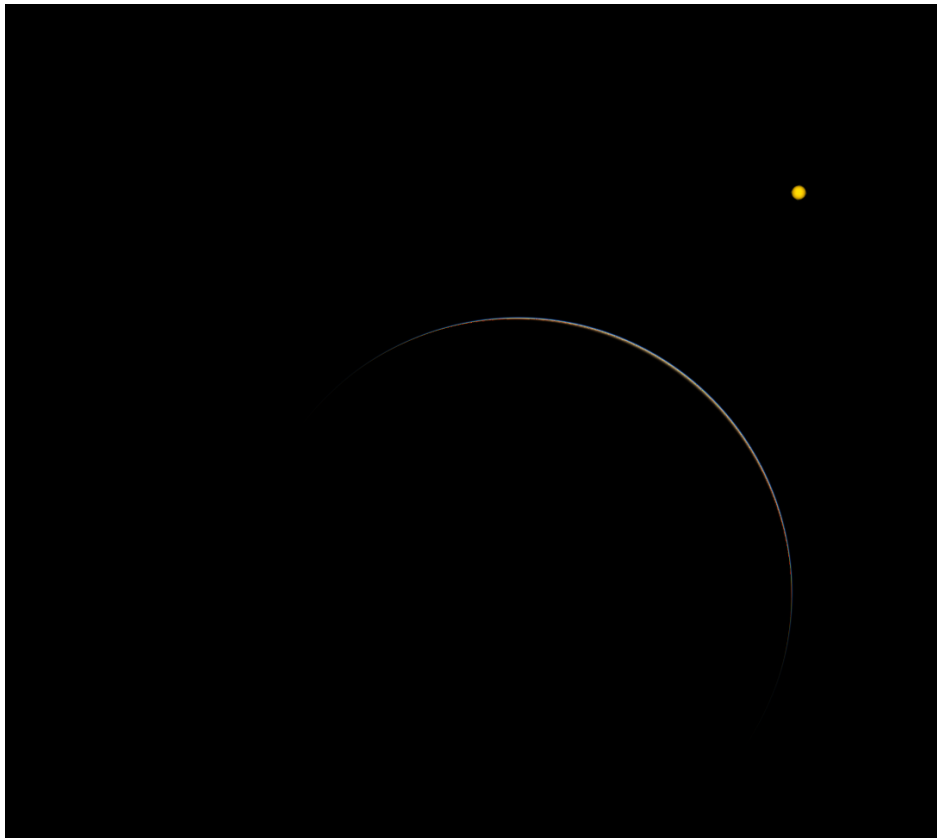
On the sphere thus created, a surface finish representing the earth's surface with clouds was applied. The result is visible in Figure 69.



**Figure 69: PANGU Earth Model**

We then obtain a plausible terrestrial model, completely comparable with the one previously used in Blender. The lighting conditions of the scene, maintaining the default settings of solar radiation, are at first glance very similar to those chosen for the simulator in Blender.

As can be seen from Figure 70, which represents an observation of the shadowed face of the Earth, the Sun is represented graphically by a yellow sphere, so as to represent the yellow dwarf nature of the star of our planetary system. PANGU allows you to choose whether to keep this view or delete it.

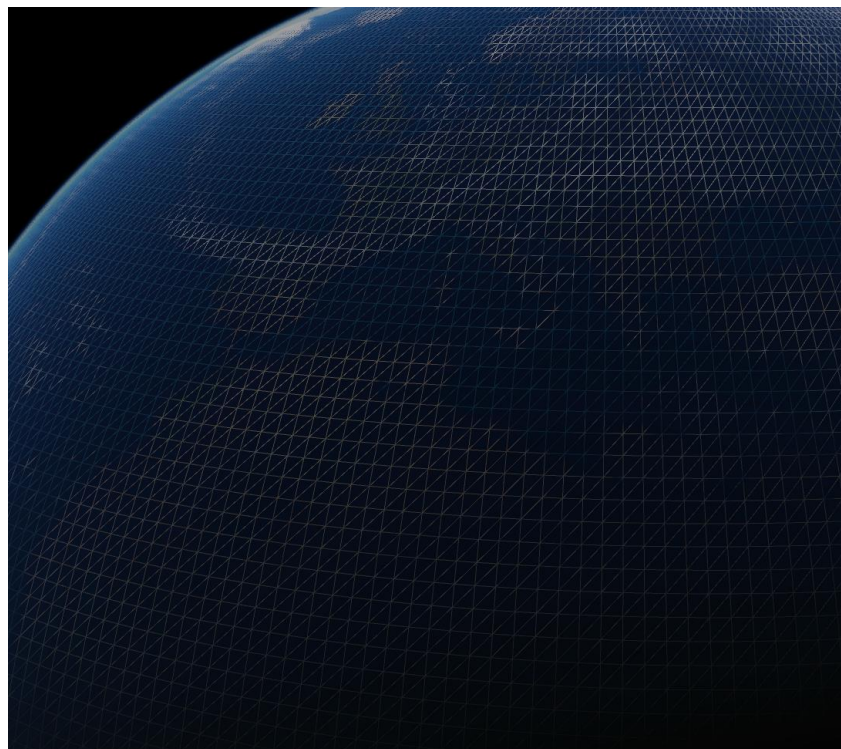


**Figure 70: Earth dark face and Sun Model**

To make clearer the process behind the creation of the planetary model, in Figure 71 and Figure 72, a detail of the Earth is represented. As you can see, the view from space on the Mediterranean Sea is reported, in the first image you have the result of rendering, from which you can appreciate the surface finish of the planet, albeit without reliefs or curvatures; in the second, the mesh at the base of the created sphere is displayed. The 3D model is therefore composed of many nodes connected to each other to simulate a sphere, on which the desired texture is applied a posteriori.



**Figure 71: Mediterranean Sea**



**Figure 72: Earth mesh under Mediterranean Sea**

In future implementations, a model based on a DEM file may be used, in order to create a more realistic earth surface, thanks to the presence of high reliefs, high reliefs and ripples, typical of the real shape of the Earth.

### 8.1.2 Space Rider Model

As for the Space Rider model in this first version of the simulator on PANGU, it was chosen for simplicity to import it as a single object, not fragmented and divided into all its parts.

The model is created as a dynamic object, capable of being moved in the scene, starting from a ".pxn" file that is imported. The ".pxn" file is created from the Space Rider Blender template.

The input parameters that must be provided to the model are:

1. Position, in reference to the center of the scene which in our case is represented by the center of the Earth. The coordinates will then be expressed in the ICRF reference system.
2. Orientation in space, uniquely defined with quaternions.
3. Scale factor
4. Material

The result is represented in Figure 73.

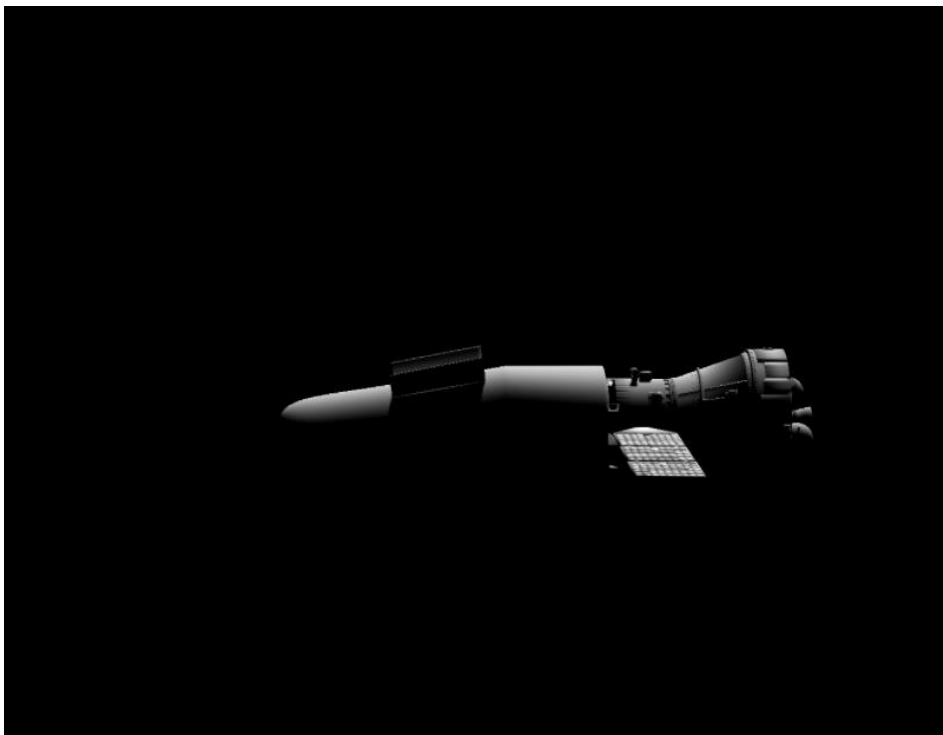


Figure 73: SpaceRider model in PANGU

Being a single dynamic object, it is not possible to differentiate surface finishes and materials on the different parts of SR, in fact in this model the entire body of the satellite, including solar panels, is made of the same material, maintaining the default one of the program. The result, as can be seen in Figure 74 is very good from the point of view of geometry, but still poor as far as the realism of the materials is concerned.

Future implementations will aim to differentiate the different components of the satellite in order to make it realistic and to give greater credibility to the analyzes that will be carried out by means of the tool.



**Figure 74: SpaceRider View over Earth**

The object is in fact all the same color and with the same optical properties, making the final result strange and unexpected. Further implementations will be related to the model of the bay containing the payloads, which in this version is represented only by the opening of the doors.

### **8.1.3 Atmosphere Model**

Of particular interest for carrying out radiometric analysis is the presence of an atmospheric model in the simulator. In fact, it allows the existence of light phenomena that substantially modify the conditions in which SpaceRider operates, making possible scattering phenomena and reflection of light radiation. The atmospheric model is defined by numerous parameters that will be explained later. It was decided to use an exponential model in which the density of particles in the atmosphere decreases with increasing altitude. The choice fell on this model because it is necessary for the creation of images from outside the atmosphere, as a model with uniform variation in density would not produce the desired effects.

Figure 74 shows the terrestrial model described in section 8.1.1 with the absence of atmosphere on the left and the atmosphere applied on the right.



**Figure 75: Earth Atmosphere effect**

The image is captured from the same identical point of view and under the same lighting conditions. Already at first glance you immediately notice a big difference in the result, making the Earth with an atmosphere very similar to a real image taken by a satellite. As expected, the illumination of the globe is also different, the model without atmosphere seems darker and with less distributed light, unlike the one with the present atmosphere that is more "wrapped" by solar radiation. This occurs due to the aforementioned optical light scattering effects due to the atmospheric model.

### **8.1.3.1 Atmosphere Parameters**

This subsection describes the parameters that define the atmospheric model used.

First of all, the size and position of the atmosphere must be defined; in PANGU they are implemented starting from the definition of minor radius, greater radius and point of origin, going to create a perfectly spherical model. The parameters are defined as follows:

$$R_{ATMmin} = R_{EARTH}$$

$$R_{MAX} = R_{EARTH} + 100 [km]$$

It then creates an atmosphere centered in the origin of the scene with a thickness of 100 [km] starting from the surface of the planet. The value of 100 [km] was chosen following similar examples of terrestrial atmospheres.

In addition to the physical properties just described, it is necessary to define the optical properties of the atmosphere itself, in such a way as to correctly simulate the behavior of the real atmosphere.

### **Optical Depth**

The optical depth of the atmosphere, normally symbolized by the Greek letter tau ( $\tau$ ), is the standard measure of how much light attenuates. It depends on the wavelength, so a separate value is



specified for each color channel. It is possible to specify the optical depths of the gas (Rayleigh) and dust (Mie) and in our case, always referring to pre-existing models we used:

$$\begin{cases} \text{Tau Reileigh} = T_m = [0.024, & 0.024, & 0.024] \\ \text{Tau Mie} = T_r = [0.0464, & 0.1080, & 0.2648] \end{cases}$$

### Single Scattering Phase Function

The Henyey-Greenstein function defines the scattering behaviour of the aerosol particles in atmosphere. It is defined as follow:

$$f_m(\theta) = \frac{1 - g^2}{4\pi * (1 + g^2 - 2g\cos(\theta))^{\frac{3}{2}}}$$

Where:

- $(\theta)$  is the scattering angle.
- $g$  is the wavelength dependent parameter of each color channel.

In our case the parameter  $g$  is defined as follow:

$$HGg = [0.76, \quad 0.76, \quad 0.76]$$

### Single-Scattering Albedo

The single-scattering-albedo is the proportion of aerosol attenuation that is due to scattering rather than absorption. This parameter is also wavelength dependent and then it is defined by three parameters, one for each color channel. In our simulator it is defined as follow:

$$w = [0.9, \quad 0.9, \quad 0.9]$$

### Scale Height

Given the exponential nature of the model, it is necessary to define a parameter that describes how the density of the particles varies as the altitude increases. It is also related to the decrease in air temperature with increasing altitude. As in the case of scattering parameters, the Scale Height is also defined for both gases and dust, being then described by two parameters (Raileigh and Mie), which in our case are:

$$\begin{cases} H0m = 1200 [m] \\ H0r = 8000 [m] \end{cases}$$

Physically the Height scale represents the increase in altitude that determines a decrease in pressure of the exponential factor "e". It is constant at a given temperature and can therefore be represented by an isothermal curve. As seen above it is indicated with "H" and can be calculated as follows:

$$H = \frac{kT}{mg}$$

Where:

- $k$  is the Boltzmann constant measured in  $\left[\frac{J}{K^{-1}}\right]$ .
- $T$  is the mean atmospheric temperature measured in  $[K]$ .
- $m$  is the mean mass of a molecule in atmosphere measured in  $[kg]$ .
- $g$  is the gravitational acceleration measured in  $\left[\frac{m}{s^2}\right]$ .

Combining this equation with the perfect gas law, the variation of pressure determined by the changing altitude is obtained as follow:

$$P = P_0 * e^{-\frac{z}{H}}$$

Where:

- $P_0$  is the pressure at sea level measured in [Pa].
- $z$  is the altitude measured in [m].

In this version of the simulator a mean temperature of 273 [K] has been chosen, obtaining a value of  $H$  equal to 8000 [m].

In Figure 75 and Figure 76 are represented two different views of the atmosphere from an inner point of view, in order to visually validate the model.



**Figure 76: Atmosphere - Sky color variation**

In the simulator the sky was created with a completely black color; placing the camera inside the newly created atmosphere and pointing outwards the result is that in Figure 76. As you can see, it is blue, which tends to vary evenly as the angle of acquisition of the image changes. A similar result was then obtained in the real world, as expected. This test confirms that the phenomena of light diffusion and scattering typical of the Earth's atmosphere have been correctly implemented.



**Figure 77: Atmosphere: Sunset over the sea**

In Figure 77, on the other hand, it was decided to simulate a sunset situation on the open ocean with the Sun outside the frame. It is easy to notice the incredible resemblance to a real sunset, with the atmosphere that when the angle of observation on the horizon lowers, tends to represent warmer colors, following the chromatic scale. This confirms the correct modeling of the phenomenon of atmospheric scattering.

#### **8.1.4 Camera Model**

In this first version of the simulator a camera was used in the visible field with the basic settings of PANGU, as no special evaluations and shots were required, but only a simulator working in the field of visible light. In PANGU you can impose the camera settings regarding the position and pointing of the optical sensor. In particular, we can define the three Angles of Euler rotation of the chamber (Roll, Pitch and Yaw) and the spatial positions on the three axes through lines of code.

Alternatively, it is possible to define the position of the camera with the parameters of range ( $r$ ), azimuth ( $az$ ) and elevation ( $el$ ) and then connect it in terms of pointing to a desired target, spatially defined by the coordinates on the 3 reference axes.

In the simulator, this second option was chosen in such a way that the camera followed and constantly pointed to Space Rider.

In a similar way to what was seen in Blender, the dimensions of the optical sensor can be defined in terms of pixels, going to define the quality of the output image.

## 8.2 Issues: Atmosphere

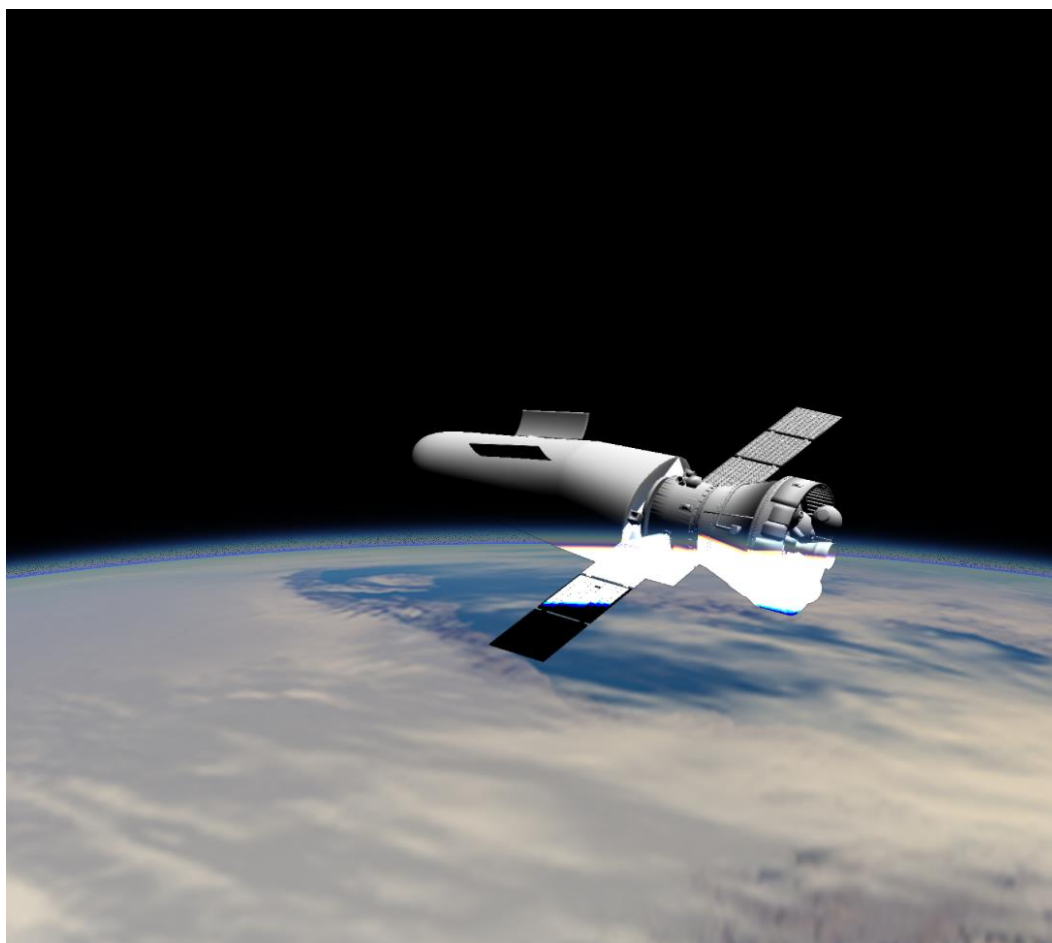
Once all the components necessary for the creation of the desired spatial scene have been described and implemented, we have moved on to the visualization of the entire simulator. The file View\_VIS.sh was then launched by Terminal, which allowed the creation of the scene in PANGU starting from the code. As expected, all the bodies had been effectively created and the lighting and atmospheric conditions were correct.

However, going to vary the angle of view of the camera on SR have obtained unexpected results. In fact, when the dynamic model of the satellite overlaps the Earth's atmosphere, very strange and unexpected graphic results are obtained, as can be seen in Figure 78 and Figure 79.



Figure 78: Atmosphere issue #1

As you can see, the satellite completely superimposed on the atmosphere is graphically completely black, even if the lighting conditions would provide for an illuminated body.



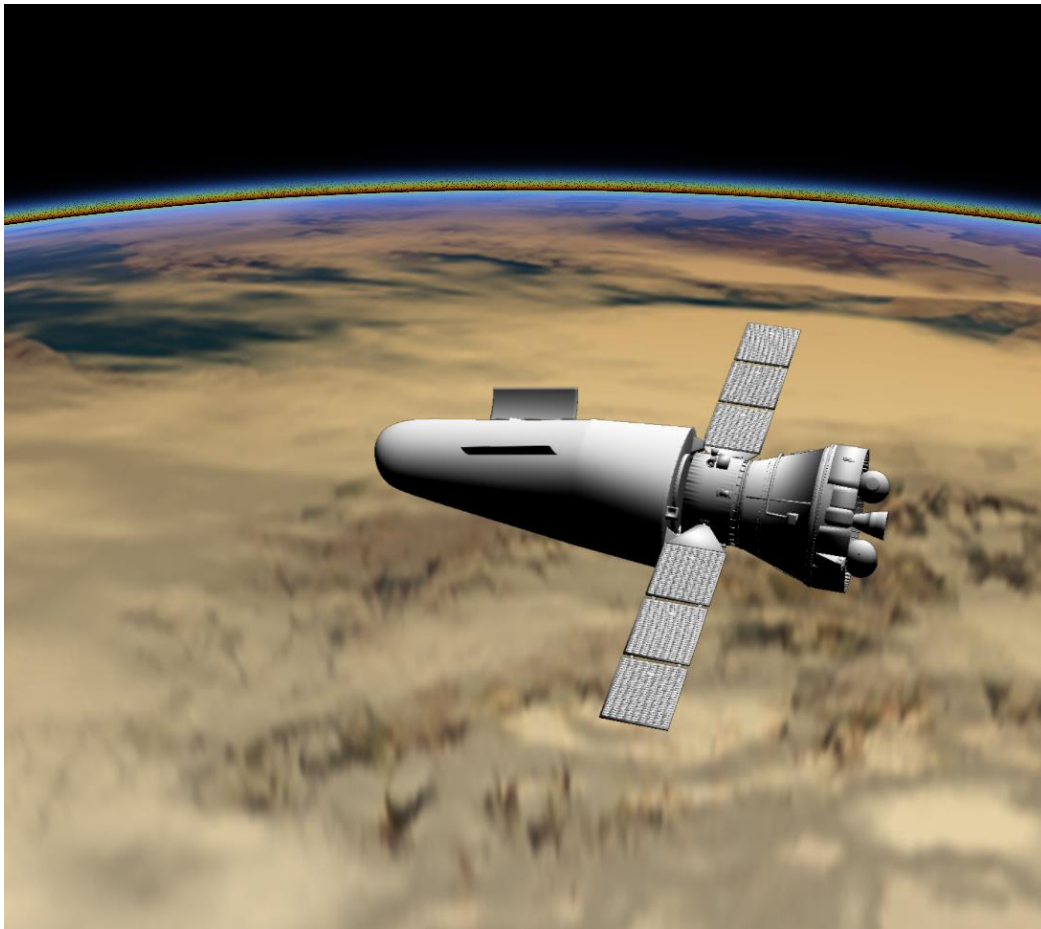
**Figure 79: Atmosphere issue #2**

Going to vary the angle of view so that only part of SpaceRider was overlapped on the atmosphere and the Earth, the result of Figure 79 was obtained. The Body is correctly rendered in the part that has the sky as its background, while in the parts where the background is represented by Earth or atmosphere strange effects are obtained, with parts either completely overexposed or completely black, thus implying an unrealistic and not physically accurate image, from which it is not possible to obtain information for radiometric analysis.

### **8.2.1 Issue Solution**

To solve the problem related to the atmosphere, several research have been carried out that have not given positive results. Also, thanks to the help of software developers the problem was identified in the basic code of the program itself. Being a high-level problem, we chose to find and implement a temporary solution, trying to get around the problem with a temporary solution.

Going to change the parameters of the atmosphere it was noticed that by increasing the thickness of the atmospheric layer the problem disappears. Increasing the thickness up to at least 400 [km], instead of the initial 100 [km], the overlap problem does not arise. As you can appreciate in Figure 80, SpaceRider despite being with the Earth as a background is well visible and well lit, without display problems.



**Figure 80: SpaceRider and Earth – Wrong Atmosphere**

With this first solution all the atmospheric parameters set previously lose their validity at the physical level, as all the boundary conditions for the simulation of a physically accurate atmosphere are no longer respected. The result is therefore a very strange atmosphere, which has unexpected colors, resulting visually "burned". This effect is due to the scattering and density variation parameters, which are directly proportional to the altitude above sea level. In fact, an extremely high-pressure reduction of the particles that make up the atmospheric gases is obtained, going to negatively affect all the other parameters, and therefore implying an image far from the desired result, albeit with the resolution of the visualization and overlapping problems seen previously. To circumvent this further problem, the parameters of the atmosphere have been modified, in order to obtain a visually acceptable result even at the cost of having an atmospheric model that is not completely accurate. In detail it has been seen that by changing the value of height scales of the atmosphere with the same proportions of the variation imposed on the thickness of the atmosphere, a good result is obtained. This is due to the fact that a higher value of height scales slows down the decrease in pressure of the atmosphere as the altitude changes, thus obtaining a plausible trend, even if scaled to a thickness 4 times greater than the starting model. The result is shown in Figure 80.



**Figure 81: Atmosphere issue solution**

The image represents the historical result obtained with this method of correction; the following features can be noted:

1. Good visualization and illumination of the SR model and the Earth even if overlapped.
2. No interference problems between the different bodies and the atmosphere.
3. Earth's surface of a correct color even with the presence of the atmosphere, unlike Figure 79.
4. Atmosphere with correct colors and with a good density distribution.

However, the following inaccuracies are still present:

1. The atmosphere on the horizon is much thicker than the real one.
2. Incorrect radiometric analyses on the horizon belt due to the conditions previously described.

### 8.3 Results

The obtained simulator is a good starting point for carrying out radiometric analysis and for the study of the lighting conditions of Space Rider. In this first version the SROC CubeSat is not implemented physically but is represented by the simulated camera in the scene, following the movements that the satellite itself should perform. Subsequent implementations will see the study of the conditions present in the docking phase, an analysis possible after having correctly imported a more precise and differentiated model of SpaceRider, which contains the right materials and the bay dedicated to payloads well detailed.

Being PANGU physically accurate in all its parts and above all being designed for the study of scenes representing space, the whole process described in Section 6 is no longer necessary, but radiometric data and the relative values of the quantities can be obtained directly from the program.

It is also necessary to emphasize the great improvement in terms of time required to render the scene.

Figures 82-83-84 represent renderings obtained from PANGU, containing the complete scene. As described in the camera model, the size of the images is:

$$DIM_{PANGU} = 1280 * 960 [pixel]$$

And the time taken for the production of each image is:

$$t_{PANGU} < 1 [s]$$

As for the corresponding Blender simulator, due to the complex light transmission algorithms and the non-peculiarity for scenes of this type, the average dimensions and times of the single frame obtained were:

$$DIM_{Blender} = 600 * 600 [pixel]$$

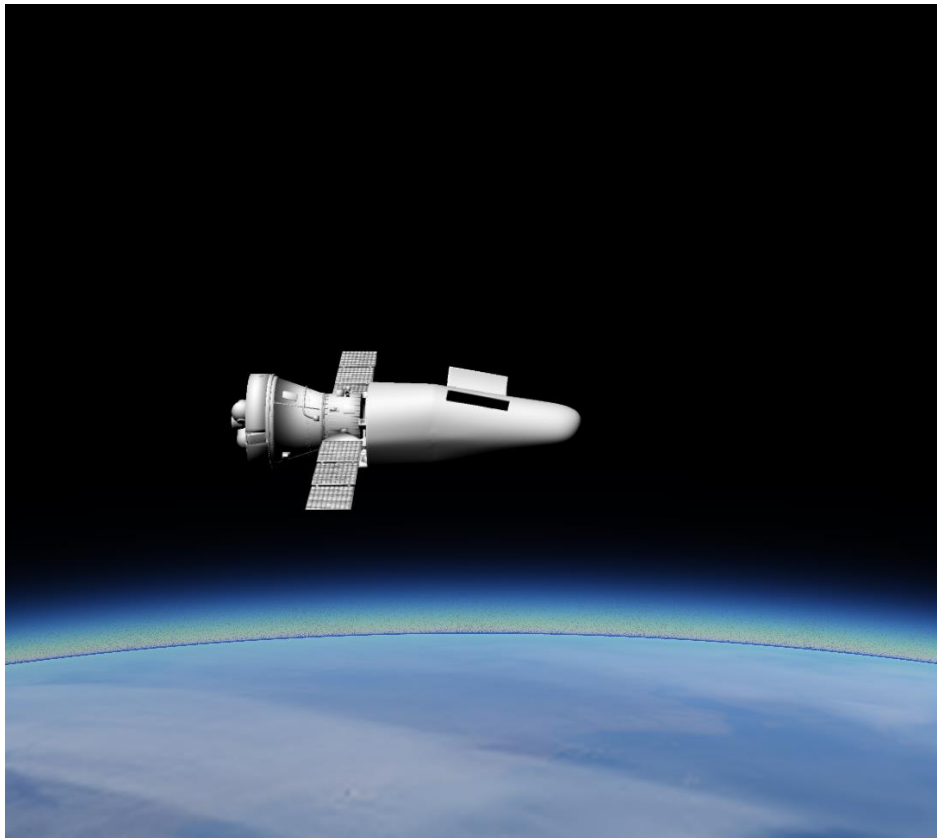
$$t_{Blender} \approx 60 [s]$$

With the same image quality, it is therefore clear that with PANGU images are obtained in a much shorter and more reasonable time.

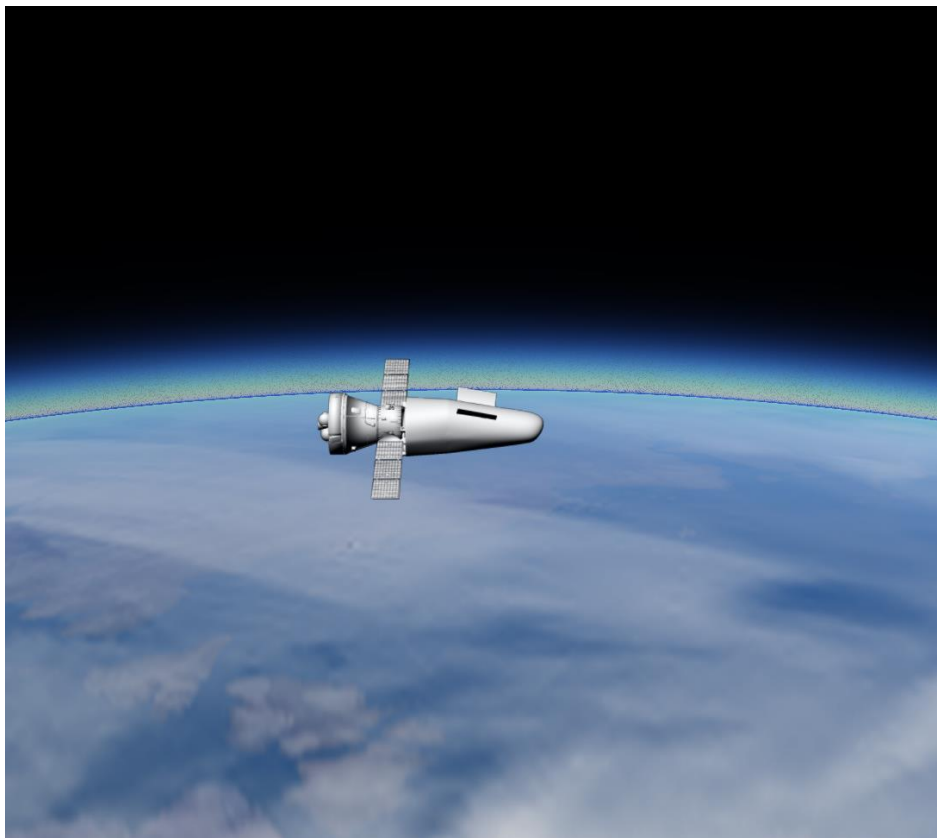
As mentioned above, 3 different renderings obtained from the PANGU simulator are reported to describe the result of the developed simulator at the visualization level. The 3 images differ in the azimuth angle imposed on the camera:

1. Figure 81: Azimuth angle of 20°, in this configuration SR is located on the black background relative to the sky, with the Earth placed at the bottom of the image.
2. Figure 82: Azimuth angle of 28°. In this configuration SR straddles the profile of the planet, immersed in the atmosphere in the two-dimensional visualization. As can be seen, there are no problems of interference and overlap described in Section 8.2 either on the part of the planet, or in the transition between the atmosphere and the black background.
3. Figure 83: Azimuth angle of 90°. In this image the optical sensor is placed vertically above the satellite, the body of SR is immersed in the terrestrial background, and it is clearly visible how the lighting conditions are respected correctly for all bodies. You can also appreciate the profile of the shadows on the body. The analysis of the shadows is very important because it will allow to carry out the analysis of the approach to the bay during the docking phase of SROC, which will be the protagonist of the future developments of this project.





**Figure 82: SpaceRider View - 20°**



**Figure 83: SpaceRider View - 28°**

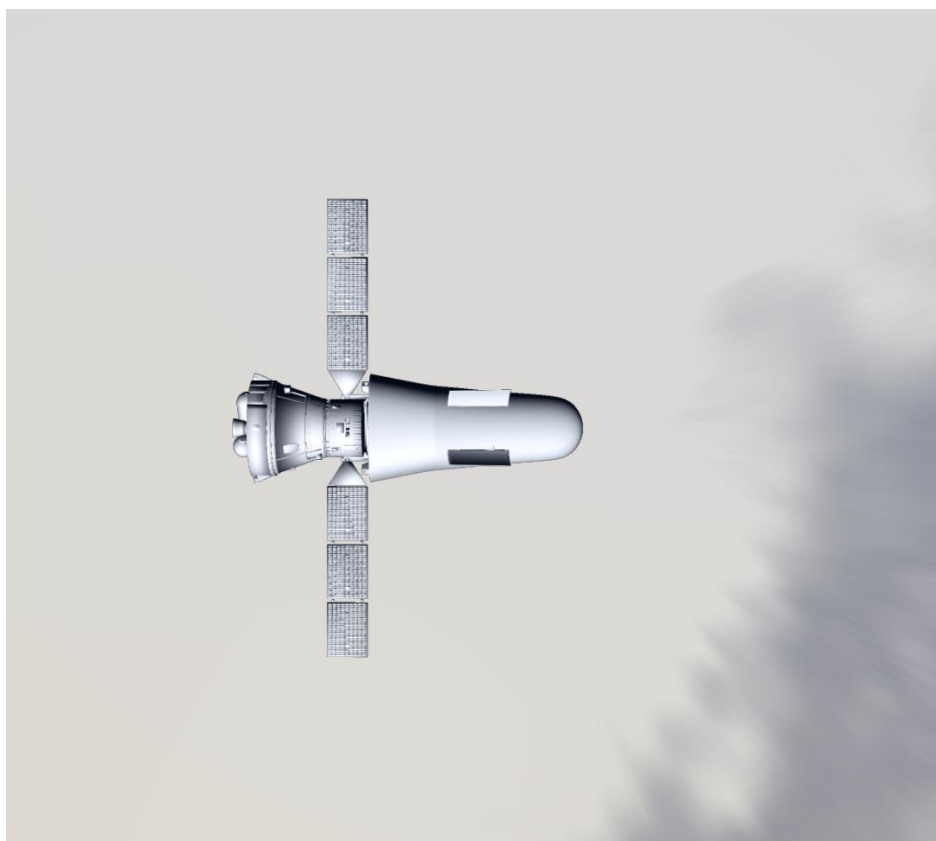


Figure 84: SpaceRider View - 90°

### 8.3.1 Pixel Flux Map: PANGU

In order to create a comparison between the PANGU simulator and the simulator created in Blender, we chose to use the python tool related to Post Processing, described in section 7 of this document, also for images obtained from PANGU.

By concentrating the comparison on the map related to the energy flow of each pixel due to lighting conditions, the following results were obtained, correlated with the images represented in the previous chapter.

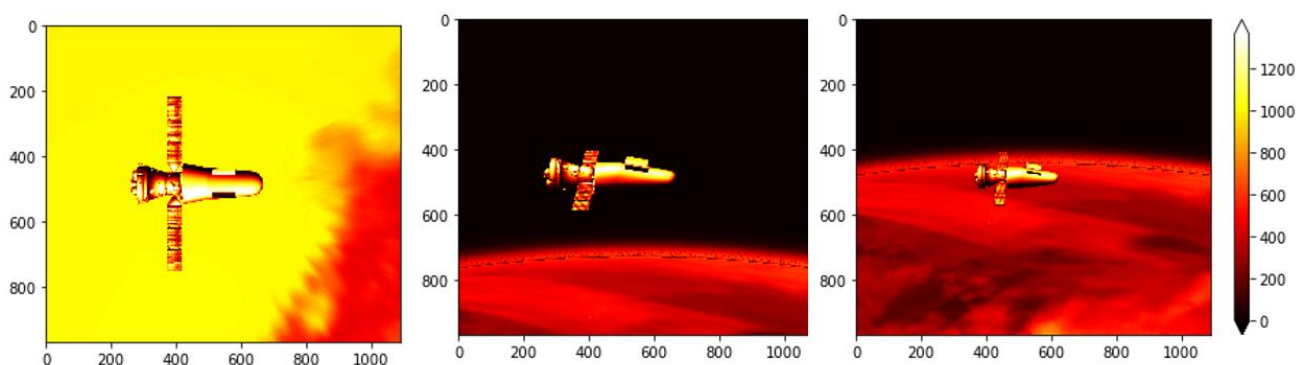


Figure 85: PANGU - Pixel Flux

The three images represent the luminous flux values for each pixel, measured in Watts per square meter. The values on the axes indicate the size of the PANGU images and the legend on the right is common for all images and represents the distribution of the color scale on the heatmap: a black

color will correspond to a value of 0 [W/m<sup>2</sup>] while a white color a value of 1367 [W/m<sup>2</sup>]. The intermediate colors represent values as the flow is gradually higher as the colors lighten.

Analyzing the results, we note that the values obtained are very similar to those relating to the Blender simulator, thus emphasizing the goodness of the latter with the chosen configuration values, even if it cannot guarantee certainties at the radiometric level due to the very nature of the program, not designed for this type of analysis.

## 9 Blender&PANGU: Comparison

Blender and PANGU are both valid tools for graphic simulations of spatial environments, each with its own strengths and weaknesses.

In Blender you can create very complex and realistic models and can be used to create videos of excellent quality of the orbital simulation. In addition, the big point in favor of Blender is its open-source nature and its ease of use, being therefore in constant development and used by a large number of people it is possible to find a solution to the problems that are encountered simply by comparing with other users of the Blender community on the dedicated forums. On the other hand, the difficulty in handling objects of different sizes in the same scene, the difficulties related to the physical accuracy of the radiometric analysis and the implementation of the sensor, and the dilated times necessary for the rendering of the images are very limiting for the purposes of the project.

PANGU, on the other hand, is a software dedicated to analysis similar to the desired one, it is therefore possible to recreate realistic situations of spatial scenes without difficulty and ensuring good physical accuracy thanks to the mathematical models at the base of the software, also combining an excellent speed of image rendering. The disadvantages of its use are mainly related to the fact that it is not an open access software, used by few people and therefore with little documentation at the base, which can be a problem when encountering obstacles during the programming and realization of the desired scenes. It is also less immediate to learn to use than Blender.

Both software therefore allows to obtain results related to the implemented scene, but PANGU offers more accurate and reliable results; on the other hand, Blender allows you to comfortably create high-quality video simulations, which can be used to define a high-level space mission during its mission phases or to make advertising videos of the mission itself.

## 10 Conclusion

The Project represents the starting point for the carrying out of a complete radiometric analysis for the evaluation of the lighting conditions of SpaceRider by the SROC CubeSat. The results obtained describe the expected conditions in reality with good physical accuracy and can help in the future development of the mission profile.

The Blender-PANGU dualism also allows you to have a multi-platform project, able to take advantage of the advantages that each program has to offer.

The simulators have been tested and analyzed and are able to provide a large number of results, moreover the Post processing scripts created and used are generic and can be applied to any project of this type, undergoing only a few changes.

Determining SpaceRider's orbital conditions will allow for even more precise results.

Future developments of the project are related to the use of a more realistic and differentiated SpaceRider model in its parts, so that it can better simulate the light phenomena on its surface.

Downstream of all future implementations, the goal will be to obtain a simulated image of the sensor from the radiometric quantities analyzed, to provide information on the characteristics of the optical sensor to be used on SROC.

Focusing on the docking phase of the satellite, a model of SROC's own approach to the SpaceRider bay will be studied to evaluate the lighting conditions in a precise way and define the need or not to insert other devices, such as thermal markers, able to guide the satellite to maneuver in bad lighting conditions, to allow the proper performance of the mission.

## 11 References

1. Colleges, David J. Eck - Hobart and William Smith. *Introduction to Computer Graphics*. 2021.
2. [https://en.wikipedia.org/wiki/Computer\\_graphics](https://en.wikipedia.org/wiki/Computer_graphics). [Online]
3. <https://www.simscale.com/docs/simwiki/cfd-computational-fluid-dynamics/what-is-cfd-computational-fluid-dynamics/>. [Online]
4. <https://www.simscale.com/blog/2019/05/fea-for-beginners/>. *SimScale*. [Online]
5. <https://www.simuleon.com/>. [Online]
6. <https://www.creativebloq.com/features/best-3d-modelling-software>. [Online]
7. <https://www.blender.org/>. *Blender*. [Online]
8. <https://pangu.software/>. *PANGU*. [Online]
9. Rowell, Nick, et al. *PANGU: VIRTUAL SPACECRAFT IMAGE GENERATION*.
10. <https://www.star-dundee.com/products/pangu-planet-and-asteroid-natural-scene-generation-utility>. [Online]
11. Londei, Marina. *Motori di Rendering a confronto in Blender: Teoria e Applicazioni*. 2015.
12. Kajiya, James. *THE RENDERING EQUATION*. Pasadena : California Institute of Technology, 1986.
13. [scratchapixel.com](http://scratchapixel.com). [Online]
14. Nimier-David, Merlin, et al. *Mitsuba 2: A Retargetable Forward and Inverse Renderer*. s.l. : École Polytechnique Fédérale de Lausanne.
15. Scalise, Antonio, Corpino, Sabrina e Daddi, Guglielmo. *Evaluation of lighting condition for SROC*. s.l. : Politecnico di Torino, 2021.
16. Cornetto, Anthony Dominic e Suway, Jeffrey. *Validation of the Cycles Engine for Creation*. s.l. : SAE International, 2019.
17. [timeanddate.com](http://timeanddate.com). [Online]
18. [physics.stackexchange.com/questions/226127/what-determines-the-sharpness-of-a-shadow](https://physics.stackexchange.com/questions/226127/what-determines-the-sharpness-of-a-shadow). *Physics Stack Exchange*. [Online]
19. Zheleznikova, O. Ye., Prytkov, S. V. e Kokinov, A. M. *Selection of an Optimal Method for Calculation of Correlated Color Temperature*. s.l. : Ogarev National Research Mordovia State University, 2016.
20. International Electrotechnical Commission. *INTERNATIONAL STANDARD - IEC 61996-2-1*.
21. Hernandez-Andrés, Javier. *Calculating correlated color temperatures across the entire gamut of daylight and skylight chromaticities*. 1999.
22. Gimena, Lazaro. *EXPOSURE VALUE IN PHOTOGRAPHY. A GRAPHICS CONCEPT MAP PROPOSAL*. s.l. : Universidad Pública de Navarra, 2004.
23. [openexr.com](http://openexr.com). *OpenEXR*. [Online]
24. Farrell, Joyce E. *Digital camera simulation*. Stanford : Stanford University, 2011.
25. Chen, Junqing. *Digital Camera Imaging System Simulation*. s.l. : IEEE.
26. University of Dundee. *PANGU v6.01 User Manual*.
27. [elprocus.wordpress.com/2013/08/02/anatomy-of-the-active-pixel-sensor-photodiode/](http://elprocus.wordpress.com/2013/08/02/anatomy-of-the-active-pixel-sensor-photodiode/). [Online]
28. [cs184.eecs.berkeley.edu/sp19/lecture/21-66/image-sensors](http://cs184.eecs.berkeley.edu/sp19/lecture/21-66/image-sensors). [Online]