POLITECNICO DI TORINO

Master of Science Course
in Aerospace Engineering

Master of Science Thesis

Design and Implementation of a VR-Based Pressurized Rover Environment for Lunar Samples Analysis

Tutors
Prof. Alfonso PAGANI
Dott. Dario ZAMANI

External Supervisor
Prof. Magesh CHANDRAMOULI

Candidate
Matteo D'AQUILA

December 2025

POLITECNICO DI TORINO

Master of Science Course
in Aerospace Engineering

Master of Science Thesis

Design and Implementation of a VR-Based Pressurized Rover Environment for Lunar Samples Analysis

Tutors
Prof. Alfonso PAGANI
Dott. Dario ZAMANI

External Supervisor
Prof. Magesh CHANDRAMOULI

Candidate
Matteo D'AQUILA

December 2025

Acknowledgements

I would like to write a few words to thank all those who have supported me throughout my journey and in the writing of this thesis.

First and foremost, my supervisors, Prof. Alfonso Pagani and Dr. Dario Zamani, who gave me this opportunity by proposing this thesis and personally guided me during my time at the Politecnico di Torino in Italy; they were always available and willing to help with anything I needed. Special thanks also to my colleague Marco Crestini, that worked alongside me on this project.

I would also like to thank my external supervisor, Prof. Mahesh Chandramouli of Purdue University Northwest, who welcomed and supported me in the United States and was always very available, involving me in various activities and helping me grow from an academic point of view. Furthermore, I would like to thank Dr. Aleeha Zafar who, while we were in Hammond at PNW university, supervised the scripting work carried out by my colleague Marco and me, and provided us with continuous help and support.

Finally, I would like to thank everyone of the Space It Up! consortium, who supported and advised me throughout this journey, and all the members of Thales Alenia Space who provided us with technical support during the pre-production phase.

Abstract

Pressurized rovers will play a crucial role in enabling extended extravehicular activities (EVAs), intra-vehicular operations, and crew habitability during the upcoming lunar missions. These vehicles will serve as mobile laboratories and living quarters, providing both protection and functionality in the harsh lunar environment. However, developing a Lunar Pressurized Rover (LPR) demands novel methodologies for design, validation, and training, as no prior operational experience, testing facilities, or physical prototypes currently exist or have been tested. An extensive literature review clearly highlighted the absence of dedicated training platforms for astronauts in such contexts, where no tools are currently available to facilitate the understanding of the tasks and operations to be performed on the lunar surface and within LPRs. To address these needs, this thesis presents the design and development of "*RoVR*", a Virtual Reality (VR) simulation of the interior and functional environment of a lunar pressurized rover for astronaut training. Specifically, this work describes the design and the implementation of tasks and activities to be performed within the Scientific Workspace section, where astronauts inside the LPR will be able to perform procedural interactions to analyze lunar samples through a simulated Virtual Reality platform. The primary objective of this work was to develop a system capable of ensuring a high level of user interaction while preserving a strong sense of realism. A key focus was to achieve a degree of accessibility so users could stay fully focused on the training tasks without being distracted by technical obstacles while within the LPR. In conclusion, this framework proposes an approach to enhance mission preparedness and to provide an interactive environment in which users can perform various operations inside the rover. The system integrates realistic physics, accurate interior modelling, and user interaction design based on scientific research, demonstrating that VR can serve as a valuable complementary tool to traditional astronaut training methods, offering cost-effective, repeatable, and safe practice.

Index

List of Figures

List of Tables

List of Abbreviations

AR – Augmented Reality
API – Application Programming Interface
APK – Android Package Kit
BSDF – Bidirectional Scattering Distribution Function
CAD – Computer-Aided Design
CSA – Canadian Space Agency
DIMEAS – Department of Mechanical and Aerospace Engineering
EVA – Extra-Vehicular Activity
ESA – European Space Agency
FBX – Filmbox
HDR – High Dynamic Range
HMD – Head-Mounted Display
HLS – Human Landing System
HUD – Heads-Up Display
IAC – International Astronautical Congress
IDE – Integrated Development Environment
ISS – International Space Station
JAXA – Japan Aerospace Exploration Agency
KARMA – Knowledge-based Augmented Reality for Maintenance Assistance
LCD – Liquid Crystal Display
LED – Light Emitting Diode
LOD – Level of Detail
LOI – Level of Interaction
LPR – Lunar Pressurized Rover
LSS – Life Support System
MEMS – Micro-Electro-Mechanical System
MFD – Multi-Functional Display
MR – Mixed Reality
NASA – National Aeronautics and Space Administration
PBR – Physically Based Rendering
PNG – Portable Network Graphics

PPL – Plane-Polarized Light
RAM – Random Access Memory
RV – Reality–Virtuality (Continuum)
SAFER – Simplified Aid for EVA Rescue
SBD – Simulation-Based Design
SEV – Space Exploration Vehicle
UHF – Ultra High Frequency
UI – User Interface
URL – Uniform Resource Locator
URP – Universal Render Pipeline
UV – Ultraviolet
VHF – Very High Frequency
VIEW – Virtual Interface Environment Workstation
VR – Virtual Reality
XPL – Cross-Polarized Light
XR – Extended Reality
XRI – XR Interaction Toolkit

Chapter 1 – Introduction and state of art

1.1 Introduction

Over the last decade, technological advancements in the aerospace industry have reignited human interest in returning to the Moon. More than 50 years after NASA's historic Apollo 11 mission, which marked a milestone in space exploration, a renewed global effort is underway to establish human presence beyond Earth. This effort is driven by both government and private space agencies, motivated by different factors such as the new space economical market that is beginning to rise, bringing the attention of the world on the space industry. Regarding this matter, there is NASA's *Commercial Crew Program* which is starting to include private companies in the space industry. For example, *Crew Dragon Resilience* reusable spacecraft operated by SpaceX has been launched on 16 November 2020 within *Crew-1* mission, carrying NASA's astronaut to the ISS. Furthermore, Elon Musk is planning fully private, all-civilian orbital flights (as in *Fram2* Mission planned in April 2025, bringing civilians to the Polar Orbit), and the goal in the near future is to establish the so-called "Space Tourism" to the Moon and beyond. In addition, Jeff Bezos's *Blue Origin* is developing its *Blue Moon lander*, contributing to NASA's Human Landing System (HLS), with the purpose to go further in lunar exploration.

All this renewed interest will result in the Moon being expected to provide new opportunities for lunar-based astronomical observations, and one of the primary objectives of lunar exploration missions is to develop livable habitats and infrastructures on the Moon, including operational bases, research laboratories, and pressurized rovers designed for extended surface exploration. Such subjects are important not only for maximizing the Moon's potential as a site for scientific experiments exploiting in-situ resources, but also for serving as a starting point for future deep-space missions. With this purpose of sending humans back to the lunar surface, NASA is planning the *Artemis III* mission, part of the Artemis Programme, a space exploration program carried out by NASA and with an international collaboration of different space agencies such as the European Space Agency (ESA), the Japan Aerospace Exploration Agency (JAXA) and the Canadian Space Agency (CSA) with a long-term objective of establishing a continuous and self-sufficient human presence. During the mission the astronauts will reach the moon surface and will do scientific work inside Starship, SpaceX's two-stage fully reusable super heavy-lift launch vehicle under development and conduct a series of moonwalks to explore the surface in the south lunar pole.

Establishing a sustainable human presence on the moon could be an important achievement as a basis for interplanetary and deep space exploration. However, to ensure the success of such long-term human missions, particular attention must be given to astronaut training and preparation for the space environment. For example, during the Space Shuttle missions, astronaut training was extremely specific and crucial, which encouraged the exploration of virtual reality as a training solution. In this context, the utilization of simulation and extended Reality (XR) technologies could provide a significant contribution. Simulations allow replication and testing of different mission scenarios and environmental conditions, while XR provides a human-centered perspective, enabling astronauts to experience operational challenges in an immersive and controlled setting, ultimately enhancing mission readiness and efficiency.

1.2 **XR History and state of art**

Extended Reality is defined as a term that comprehends real-and-virtual combined environments and human-machine interactions modelled by a computer. Within XR technologies it's possible to define Virtual Reality (VR), Augmented Reality (AR) and Mixed Reality (MR):

- VR technology creates a completely digital world wherein users can explore and interact with elements inside this simulation and are completely immersed.
- AR technology refers to both the digital world and the real (physical) world seen side-by-side (digital elements overlaid on real-world) and enables the imposition of virtual components on to the real environment.
- MR technology is similar to AR but there is a specific blending of the virtual and physical elements and a digital interaction with a real-world.

In 1994, these technologies were defined in terms of *reality-virtuality (RV) continuum* [1] by Paul Milgram, so as a line where the two extremes represent the completely real environment on the left and the completely virtual environment on the right, while everything included within this line represents the XR technologies spectrum, as illustrated in *Figure 1*, the user can interact with virtual components just as with the real world, triggering software responses in the XR system.

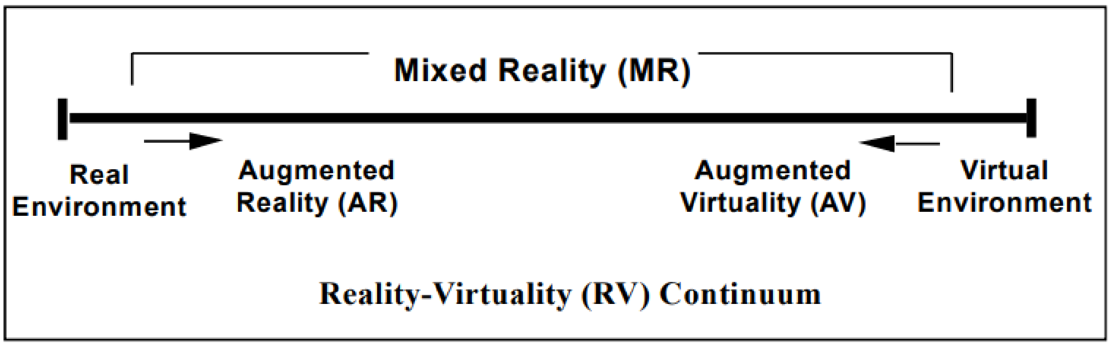


Figure 1. Concepts of RV continuum

1.2.1 **Virtual Reality**

Regarding Virtual Reality, its purpose is to immerse users in 3D digital environments that stimulate their senses. First VR device was *Sensorama* (Figure 2. Sensorama simulator), designed in 1962 by cinematographer Morton Heilig, that provided a controlled environment, where the viewer's head is immersed in the equipment. The scenario simulated was a motorcycle ride through Brooklyn where

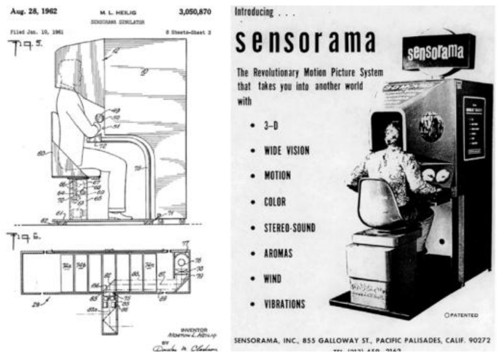

Introducing . . .
sensorama
The Revolutionary Motion Picture System that takes you into another world with
• 3-D
• WIDE VISION
• MOTION
• COLOR
• STEREO-SOUND
• AROMAS
• WIND
• VIBRATIONS
SENSORAMA, INC. 855 GALLOWAY ST., PACIFIC PALISADES, CALIF. 90272

Figure 2. Sensorama simulator

many of the senses of the user were stimulated by the Sensorama simulator.

The first attempt to immerse a user within a computer-generated environment utilizing the first so-called head mounted display (HMD) worn like some kind of helmet was Ivan Sutherland's *Sword of Damocles* created in 1968, which is considered as the real inventor of the VR As we imagine it nowadays [2].

His idea for enhancing the realism of the simulation was that users, moving their heads, expected that their view would change along with the movements, so he created a head motion tracker with a mechanical arm. In the 1980s, NASA created an HMD prototype device called *Virtual Interface Environment Workstation* (VIEW), which featured a magnetic tracking system instead of a mechanical one and also reproduced localized sounds cues in fixed positions or in motion trajectories relative to the user [3]. Since the 1990s, several advancements in VR have been made. One of the most important, designed by Carolina Cruz-Neira, was called the *CAVE*, a high-resolution visual interface in a room where three walls and the floor were used as projection screens for computer generated stereo images seen by the user with a pair of stereo glasses [4]. Based on NASA's advances, several companies began developing VR technology for home use. For example, in 1994 the Nintendo company announced the launch of its new VR console for video games, called *Virtual Boy*. To see the stereoscopic images, the user had to look at two little LED screens. In the new millennium, after 2016, the VR headset industry became popular in the video game and training industries, and several companies released VR headsets as we know them today, both wireless and wired [5]. The one with more success, and that gives a direction to the industry was Oculus Quest that in 2022 was rebranded as "Meta Quest 2" in line with Facebook's new strategy. The Quest 2 features a Snapdragon XR2 processor and is equipped with an LCD display for each eye, offering a resolution of 1832 x 1920 pixels and a refresh rate of 120 Hz. It also includes four cameras, two of which are positioned downward to track hand movements [6].

Interacting with a virtual environment is a critical component of many VR applications. Tracking systems of a variety of mediums (optical, magnetic, ultrasonic, inertial, etc.) enable the position and orientation of physical objects to be calculated within a physical space in real time. This becomes especially valuable when calculating the correct viewing perspective for the user. Coupled with gesture recognition algorithms, tracking systems allow natural body movements to be translated into functional interaction techniques. [7]

1.2.2 **Augmented Reality**

The term "Augmented Reality" was first introduced in 1992, by Thomas Caudell and David Mizell [8]. They stated as follows: "*This technology is used to 'augment' the visual field of the user with information necessary in the performance of the current task, and therefore we refer to the technology as 'augmented reality*". The first AR device consisted of a helmet with a mechanical counterweight arm and used ultrasonic transducers to track head movement, connected to a TX-2 computer. To show the images to the user, the device utilized two little cathode ray tubes. One of the first practical applications of this technology was to help in the maintenance tasks of a laser printer, in the in 1993 for the Knowledge-based Augmented Reality for Maintenance Assistance (KARMA) project (*Figure 3*).

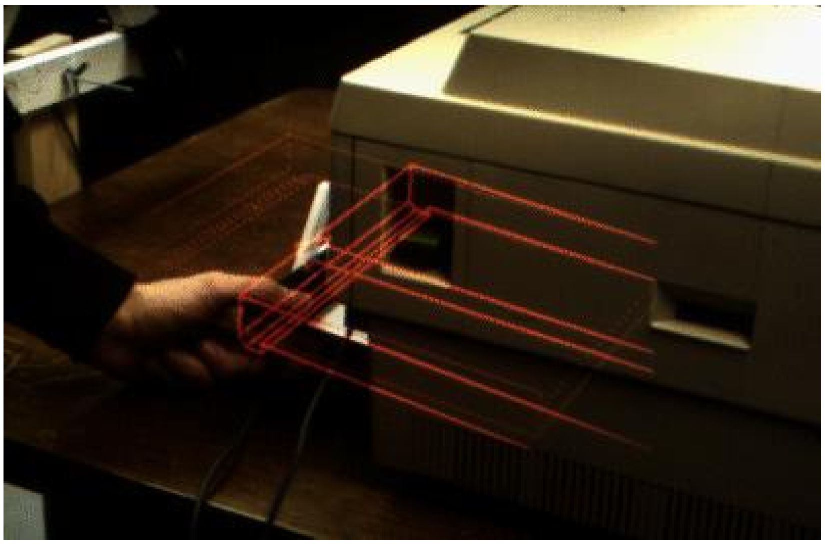


Figure 3. 1993 KARMA AR system [8]

In 1998, AR techs were integrated in sporting events, such as football, as a helping tool for the referees, adding a virtual line, imposed on the images, to the position of the players to evaluate if they were in offside irregular position during the games. In 2012 Google announced *Project Glass*, a device featuring a lens frame equipped with a camera, a small screen positioned above the right eye, and a touch-sensitive bar on the right temple of the frame. These lenses enabled users to quickly search for information and overlay it onto their field of view. In 2017 the product was relaunched with some improvements and modifications addressed to the industrial market, where augmented reality glasses could be very helpful in the production process.

1.2.3 Mixed Reality

Figure 4. Microsoft HoloLens first design

Mixed Reality blends virtual and physical and digital elements from VR and AR and lets the user interact with virtual objects in a real environment. Unlike the other two, MR technology was developed much later over time. In fact, the first application was created in 2010 by Microsoft, the *Baraboo project*, the first name of what would become Microsoft's *HoloLens* (*Figure 4*), which was only released to the market in 2016. This device had transparent lenses that allowed users to view the real world while overlaying digital images onto it. In addition, being wireless and highly convenient to use and transport, highlighted the great potential of MR in education, design, and manufacturing. Few years after the first one, Microsoft produced also *HoloLens 2*, with a field of view that's twice as big as before. To generate the images three lasers in the HoloLens 2 shine into a set of mirrors that oscillate as quickly as 54,000 cycles per second so the reflected light can paint a display. Those two pieces together form the basis of a *microelectromechanical system* (MEMS) display. The HoloLens is produced for corporations, not consumers. It's designed for what Alex Kipman, Microsoft's software engineer and main developer of HoloLens, calls "*first-line workers*", people in shops, factories and industries [9]. In addition, even Apple, in 2023, entered the market of XR, and developed *Vision Pro*, a more advanced, in terms of resolution and interaction, pair of glasses that allowed user navigation and interaction with digital content. This rapid progress shows how XR technologies are becoming part of everyday life, not just for entertainment, but as practical tools for professionals across various industries.

1.3 Digital Twin

As described by S. Boschert and R. Rosen in [10] Digital Twin refers to a comprehensive physical and functional description of a component, product or system, which includes more or less all information which could be useful in all lifecycle phases. A digital twin for a space application has been developed in the Apollo Program and described by NASA as "an integrated multi-physics, multiscale, probabilistic simulation of an as-built vehicle or system that uses the best available physical models, sensor updates, fleet history, etc., to mirror the life of its corresponding flying twin" [11]. In [12] it's said that the twin was also used extensively for training during flight preparations. During the flight mission it was used to simulate alternatives on the Earth-based model, where available flight data were used to mirror the flight conditions as well as possible, and thus assist the astronauts in orbit in critical situations. The main purpose of digital twin technology is to serve as an all-encompassing digital counterpart of a real-world object. This innovation highlights the two-way relationship between its digital and physical forms, striving to align and replicate the real object's actions, efficiency, behavior, performance and overall lifespan within the virtual space. Digital twin technology is currently one of the most closely watched innovations in the engineering world due to the vast opportunities it offers. In fact, in 2018, Lockheed Martin recognized it as one of the most revolutionary technologies in the aerospace and defense sectors.

1.4 **Developing platforms**

In recent years, game engines have quickly become popular for designing virtual environments. Their fast and flexible tools make it easy to create rich and immersive visual experiences in a very short time. These frameworks allow creators from various disciplines (specifically engineering, architecture, education, and entertainment) to design immersive and interactive visual experiences efficiently. Among all the available platforms, the two most popular used today are Unreal Engine and Unity.

Unreal Engine was created by Epic Games and launched in 1998 originally designed for the gaming industry. The Unreal Engine platform was developed using C++ and offers a high degree of portability, supporting various platforms including desktop, mobile, console, and virtual reality and incorporates dedicated libraries for game creation [12]. Developers can easily access and use all the key code components to create impressive games in a familiar way by applying object-oriented programming.

In [13], Unity is described as a versatile, cross-platform game engine that comes with its own integrated development environment (IDE). Developed by Unity Technologies, it stands out as one of the most comprehensive and powerful tools for creating both 2D and 3D games. Unity supports multiple deployment targets, including web, desktop, console, and mobile platforms, offering developers an all-in-one solution for interactive content creation. It has different advantages:

- It offers exceptional community support and tutorials for learning.
- Unity's Asset Store is extensive and well-curated.
- It supports scripting in C# for flexible development.
- The engine enables the creation of multiplayer games.
- Unity is straightforward to learn and apply.

For this project, we decided to use Unity as development platform because of its intuitive interface and the extensive support it provides to users who are just starting out in game development (*see Chapter* 4 – Unity implementation and game development*)*.

1.5 Related work

In this subchapter, different studies, considerations and applications of VR technology in the space sector for training and design purposes will be shown. Astronauts training approaches based on realistic simulations are highly versatile, customizable, and efficient in terms of safety, costs, and time savings. VR simulations allow the trainees to experience physical feedback while executing tasks as illustrated in [14], and different mission scenarios and environment parameters can be replicated and tested. For example, in [15], Hochberg et al. analyzed Simulation-Based Design (SBD) approaches to assess their contribution to the design and planning phases of the engineering process. With the aid of VR prototypes and 3D models, different environments were created to support some of the simulation-based design activities. The adoption of this process based on VR, enables a faster and more reliable design of components and systems. In this study a foundational basis has been provided to address identified engineering elements for a Moon mission. These analyses and designs were produced by student teams using the VR based environments. Students also interacted with NASA engineers, who provided feedback to them. The environments analyzed were three:

a) An automated Lunar landing system from Gateway to the Moon to transport supplies equipment and other components.
b) A Cable robot-based system for transporting supplies on the moon surface.
c) Study of path planning approaches to transport materials on the Lunar surface.

The whole environment of the three analyses has been programmed with C#, JavaScript and the Unity 3D game engine, while the CAD and 3D models has been modelled with SolidWorks.

In the first case of study, as stated in the paper, the focus was to explore and propose design alternatives for the transportation of payloads from the Gateway to the Lunar surface. In this environment an autonomous transport system capable of docking with a storage station functioning in lunar orbit was studied. It would undock after collecting the payload from the station and then move to the lunar surface and land with the thrusters. The transportation system was designed to carry the estimated payload mass of the NASA Multi-Mission Space Exploration Vehicle. The VR simulation of the landing phase is showed in *Figure 5*.

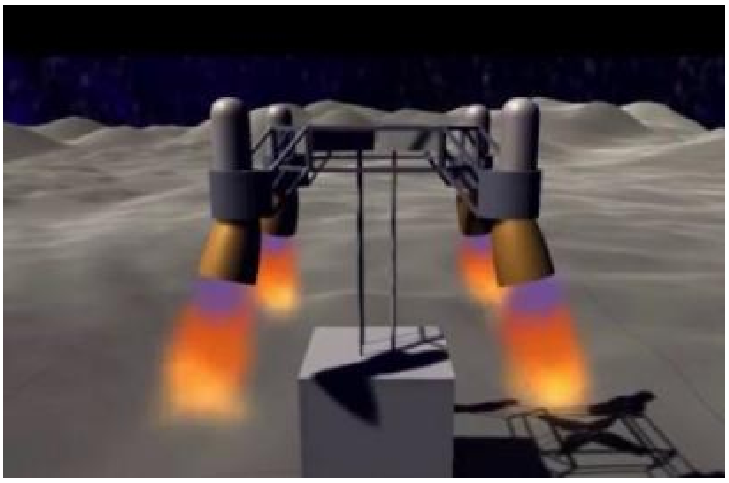


Figure 5. A view of the Simulation Environment showing the transportation moving closer to the Lunar surface to drop the payload [14]

In the second case a cable robotic system was proposed to pick up and target payloads (equipment and supplies) and transport them to various locations of the lunar surface (*Figure 6.* top view of the Lunar surface, the obstacles and the transportation vehicles [15]). Various crane concepts were developed through the SBD method. A VR environment was built enabling users to visualize concepts, evaluate different options, and suggest improvements. Additionally, simulations allowed NASA engineers to engage directly with the VR models and offer feedback on the preliminary designs, following system engineering practices.

The last environment objective was to determine collision-free paths for the transportation of the payloads while moving on the lunar surface. The virtual simulation gave a visual aid of the correct path calculated by the algorithms and showed in a more realistic way the results of the calculations.

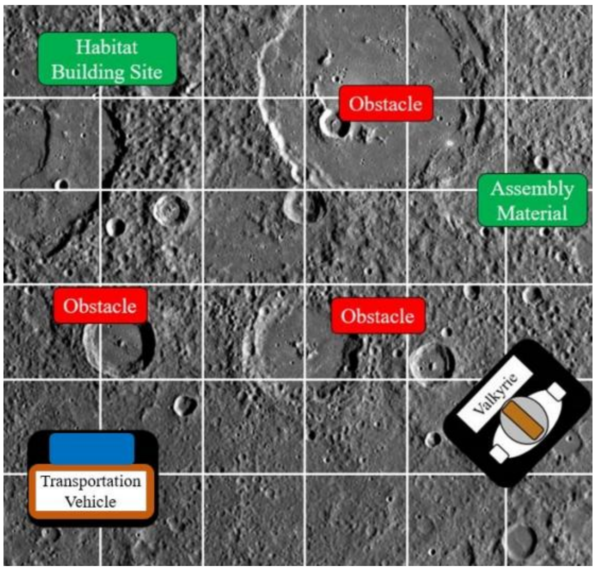


Figure 6. top view of the Lunar surface, the obstacles and the transportation vehicles [15]

Angelica Garcia, from the NASA's Virtual Reality laboratory of the Johnson Space Center in Houston, highlighted that VR systems are critical for astronaut training and that the market in this sector is starting to be very important for the future missions [16]. NASA's VR lab (*Figure 7*) in Houston uses *HTC Vive Pros* and the Lighthouse tracking system for training, with two immersive stations that allow two crew members to be immersed in the same virtual environment simultaneously. The paper described by Garcia shows three major *Hardware-in-the-Loop* VR simulation systems, connecting actual hardware components to a simulated environment that mimics the real-world. Two of these simulations analyzed are:

a) The Simplified Aid for EVA Rescue (*SAFER*) system.
b) The Mass Handling System *Charlotte.*

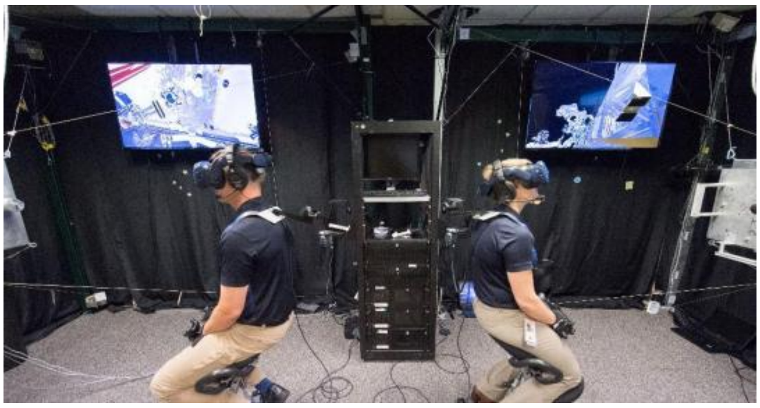


Figure 7. NASA's VR lab training Setup [16]

SAFER system is a backpack attached to the LSS on the space suit and has been part of astronaut training since before 1994. Its purpose is to be a redundant system to perform spacewalks, and it's required to be used during every EVA. SAFER simulation handles and controls all aspects of physics, dynamics, sensor inputs, avionics, power systems, and thruster logic models. It has been simulated in the VR lab with the aid of *Trick*, a NASA framework for simulation, and its purpose was to perform a checkout of the system looking for system failures in all the subsystems. A second part of the training included a flying simulation of the SAFER system through the VR. Furthermore, thanks to the adaptability of the Virtual Reality technology, many configurations and scenarios with different rotation rates, lighting or fuel quantity were customizable by instructors, showing the usefulness of this type of simulation. Regarding the Mass Handling System, Charlotte robot uses eight motors that are driven by the simulation to move according to the configured payload mass properties. It's easily reconfigurable and does not require physical payloads that have to mimic real mass properties, giving sensations close to what astronauts experience in space.

During the IAC 2024 event in Milan, Franchini et al. from Politecnico di Torino presented a study regarding lunar exploration application of Virtual Reality for astronauts training [17]. In this study three hypothetical mission scenarios have been produced.

In the first environment the user could immerse himself into a lunar base concept. All the moon terrain has been virtually reproduced based on the data from Apollo and other missions. Textures of materials were produced to give a more realistic feeling to the user. In addition, thanks to the Unity physics engine, the astronaut could understand how to safely interact with objects in the simulation with the moon gravity level (*Figure 8*). The avatar of the player was also rigged to let the astronaut to associate his movements in the real environment with the animation of the character.

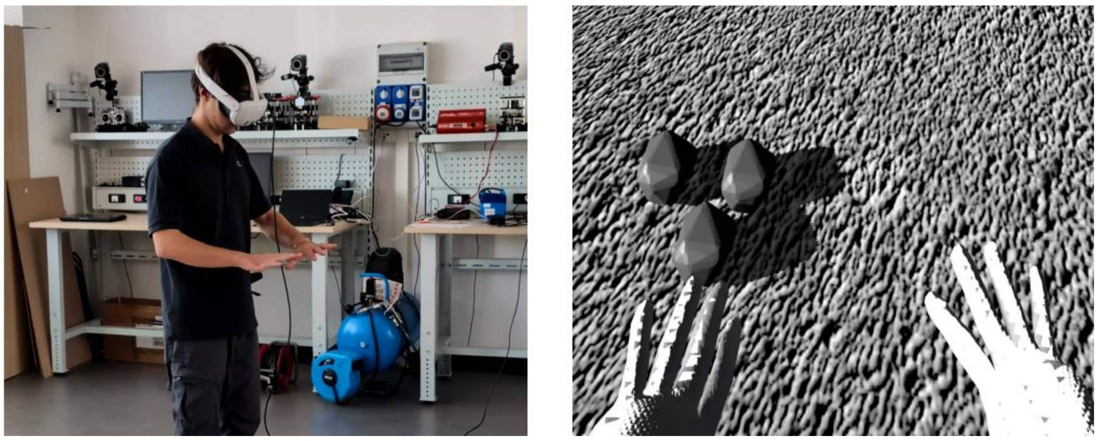


Figure 8. The real-world movements into simulation: user and rigged avatar correspondence [17]

The second simulation was composed of a scene to perform rock sample collection with the aid of a series of tools for digging. The user could interact with the tools and use them to collect samples.

The last environment simulated described a Moon landing site near the South Pole with a descend lander and a semi-autonomous exploration rover. The rover was configured to autonomously navigate through the environment by generating a 2D grid map from laser scan sensor readings, allowing a path to be planned by the user, that could also operate it with a joystick. It was also possible for the user to watch the cameras of the surrounding during the navigation and to monitor the rover progress and status through a set of tools designed to visualize real-time data.

In conclusion, from the "Acta Astronautica Journal", Andrea E.M. Casini et al, from the department of Mechanical and Aerospace Engineering (DIMEAS) of Politecnico di Torino, have conducted research on the analysis of a Moon outpost dedicated to the validation of enabling technologies for human space exploration [18]. In this case study was developed a VR facility with a first-person interactive perspective, allowing for specific in-depth analyses of ergonomics and operations, with a focus on illumination rates led to identify favorable locations for the outpost and to study the light conditions. A first draft example of lunar base elements was created into an interactive virtual scene. Also, a Martian scene has been reproduced with the VR tool to show its great reconfigurability and its potential use for future Martian mission design. This kind of virtual simulation technology showed a high potential for training purposes. As a matter of fact, gradually integrating such tool into the standard processes will lead to positive training association and familiarization.

These studies showed that Virtual Reality has the potential to serve as a versatile, cross-disciplinary platform, enabling innovative and more efficient testing of modular design, optimization of operational procedures, familiarization with low-gravity conditions, and training for activities on the moon surface.

1.6 Thesis outline

Chapter 1 introduces the context of this work. It begins by highlighting the rising importance of human space exploration and the need for realistic simulations in preparation for lunar missions. The chapter continues with an overview of Extended Reality (XR) technologies: Virtual Reality (VR), Augmented Reality (AR), and Mixed Reality (MR). A review of related work follows, identifying existing tools and research projects that inspired this thesis. The chapter concludes with the present outline to guide the reader through the structure of the thesis.

Chapter 2 describes the methodology adopted to develop the VR simulation. It defines the initial deductive approach used to formulate a storyboard describing all possible events that may occur within the simulation. Subsequently, it specifies the software tools and equipment employed for programming and game development.

Chapter 3 focuses on the 3D modelling process. It details the creation of the virtual 3D assets, including the pressurized lunar rover's interior and exterior systems and elements along with the development of a top-down functional and hierarchical block diagram of each component to be modelled. The chapter discusses how to ensure visual realism and physical accuracy, and how to obtain a good performance in a VR environment.

Chapter 4 describes the implementation of the simulation within the Unity game engine. It covers the integration of 3D models into Unity, the development of the tasks, their interaction mechanics, and the programming of user interfaces and training logic.

Chapter 5 shows the results achieved in this project, and how the simulation has been developed.

Chapter 6 presents the conclusions drawn from the project, summarizing the results achieved. It includes an assessment of the simulation's performance and the possible future implementations.

Chapter 2 – Methodology

2.1 VR game developing and pre-production

Considering the renovated interest of the scientific community in lunar exploration and the upcoming missions planned by multiple space agencies and private companies, it was decided to develop a simulation environment of potential relevance for the near-term market. In particular, the focus has been placed on the simulation of the internal configuration of a pressurized rover operating on the lunar surface, which in the future could be adaptable to hypothetical Martian missions as well. With this approach, it becomes possible to build a practical and adaptable training environment aimed at supporting astronauts and giving them the knowledge to handle these vehicles in future missions.

The development of a simulation of a LPR (*Table 1*) is a process comparable to that of immersive video game design. During the pre-production phase, the main goals of the simulation are defined through an extensive literature review, balancing scientific accuracy with user experience. This stage includes the identification of software tools (game engines, frameworks) and hardware requirements (headsets, controllers). Therefore, this phase represents a critical moment where technical constraints and research objectives must be aligned to build a coherent foundation for the simulation.

This process has begun with the identification of the most suitable technology to be used. Virtual reality was identified as the preferred solution, since for applications on the lunar surface it was considered an innovative method to immerse astronauts in an environment that closely replicates real conditions, allowing them to fully engage with the scenario and carry out various onboard operations. Another advantage of VR technology is that within a completely digital environment, any type of scenario can be created and filled with whatever objects are required for the experience. In addition, being immersed in this virtual space captures users' attention Furthermore, it offers several advantages, such as:

- Safety in training, user-friendliness and accessibility.
- Interactivity with the environment and objects within the simulation.
- Modularity and upgradability of scenes and scenarios.
- Significant time and cost savings during development and testing.
- Ergonomic studies enhanced.

For this simulation, the Unity 3D engine was selected due to its easy and clean interface. The engine's scripting API uses C#. Multiple scenes can be created and loaded at the same time, each containing a hierarchical tree of "*GameObjects*" functioning as a scene graph. The position, rotation, and scale of each *GameObject* are defined relative to its parent. In addition, the hardware selected for the consumption and development of the VR lunar habitat is the Meta Quest 2 headset, and its two controllers. The choice of this device is motivated by the fact that it was made available for this research by my thesis external supervisor Prof. Magesh Chandramouli, from the Department of Computer Graphics at Purdue Northwest University.

The Meta Quest 2 is a standalone VR headset, which means it does not require a dedicated external PC or tethered setup to operate even if in this configuration all the system performance depends solely on the device. Nevertheless, this feature greatly simplifies deployment, reduces infrastructural requirements, and ensures higher availability in training contexts where fast installation and flexibility are essential. Secondly, the Quest 2 represents an optimal cost–performance trade-off. Equipped with a Qualcomm Snapdragon XR2 chipset, 6 GB of RAM, and support for refresh rates up to 120 Hz, it provides sufficient computational and graphical performance. With the tracking feature, users can move naturally and interact with their hands, while their avatar mirrors these motions in the virtual environment. At its most basic configuration, this avatar is represented by a digital version of the headset and controllers. The user, before starting the simulation, must define the boundaries of the virtual play area. These can either be done manually or automatically set by the headset based on the real-world dimensions of the room where the device is being used.

2.2 Storyboard iterations

To ensure a structured and consistent user experience, the design process began with an initial storyboard generation to highlight the training purpose of the project. In the first iteration of this process, the simulation was divided into two main missions, each organized into sequential tasks. These preliminary scenarios, although not utilized in the final storyboard, provided an insight of possible future integrations and development of the "*RoVR*" application, thanks to the high modularity and upgradability of the Virtual Reality technology.

The first proposed mission focused on the transition from the lunar lander to the pressurized rover. The sequence began inside the lander, immediately following touchdown on the lunar surface. At this stage, the operator was required to perform several critical checks: verifying the lander's operational status, monitoring onboard systems such as power and life support, and ensuring their own health parameters were within safe limits. Communication with Mission Control on Earth was integrated as part of these initial steps, reinforcing the importance of procedural validation and real-time reporting. Once these checks were completed, the operator moved on to planning to move to the LPR. This stage introduced environmental monitoring and preparation of the necessary equipment and then the actual operation to reach the rover.

The second mission was designed to focus on external operations, starting inside the pressurized rover. Here, the operator's tasks consisted of system verification and familiarization with the rover's modules, tools, and inventory. This mission also included the preparation for the EVA, structured around a virtual checklist that guided the user through equipment selection, suit wearing, and life support system connection. Unlike the first mission, this one emphasized on scientific and exploration activities on the lunar surface. Once outside the pressurized rover, the user was expected to engage with multiple tasks, such as collecting samples with specialized tools, deploying instruments for data collection, and installing cameras for documentation purposes.

In the later stages of the design process, after the literature review and the identification of the gaps in this research topic, it was decided that rather than emphasizing extended EVA procedures, it was better to simulate the internal environment of the rover. These considerations shifted the focus on intra-vehicular activities, such as cockpit operations and navigation planning, as well as scientific tasks at the onboard workbench, that will be discussed in detail in Chapter 4.

2.3 **3D modelling procedure in VR simulations**

Following this stage, the focus shifted to 3D modelling, where the main assets were created: the pressurized rover, the lunar terrain and all the objects within the rover. The modelling work relied on technical references from literature even though a lunar pressurized rover has not yet been designed or operated on the Moon, this makes the process of sourcing references more complex while also leaving room for engineering design choices based on the specific tasks the rover will be required to perform.

The 3D modelling process begins with the identification of what will be included in the simulation itself. Since the design choice consist in the modelling of the interior configuration of the LPR, the first step is to divide the space into functional blocks corresponding to each specific area (*Figure 9*). These blocks are then connected through a network of functional relationships, which together form the hierarchical diagram. It has been decided to focus on five main macro sections:

- The Cockpit: an area dedicated to controlling the vehicle systems and navigation.
- The Scientific Workspace: dedicated to experiments and the analysis of lunar samples.
- The Crew and Utility: dedicated to the needs of the crew and the basic utilities.
- The Airlock and EVA: an area to access/exit the rover via an airlock and perform Extra-vehicular activities.
- The Emergency: a set of components for emergencies.

For each of these sections a hierarchy diagram has been developed (*see Chapter 3*), but for the purposes of this thesis project, most of the operations and tasks within the simulation will be carried out in the Cockpit and in the Scientific Workspace sections.

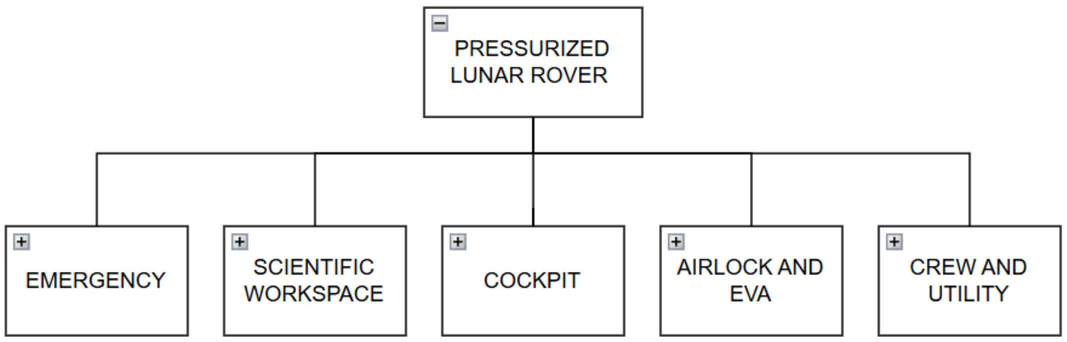


Figure 9. Main sections development

When developing a virtual reality simulation, it is really important to pay attention to optimization, since the higher is the fluidity of the final product, the stronger will be the sense of simulated reality, a fundamental aspect in this type of application. Therefore, compromises must be made to maintain a stable frame rate, adopting specific strategies, particularly in the field of 3D modelling. In fact, it is essential to keep the geometry of the models as simple as possible and to reduce the number of polygons (for example, by using textures), as these elements can heavily impact system performance. This aspect is especially relevant in our case, since a standalone VR headset (the Meta Quest 2) was chosen to increase accessibility as previously discussed. A standalone VR headset has its limitations in computational and processing power because it's not supported by any external hardware or dedicated computer system.

The first step to model an element is to use a reference to create a mesh that contains all the information of the geometry considered. Then to give a realistic look to the model we recreate, through geometry node function, the textures and the materials of each component (*Figure 10*).

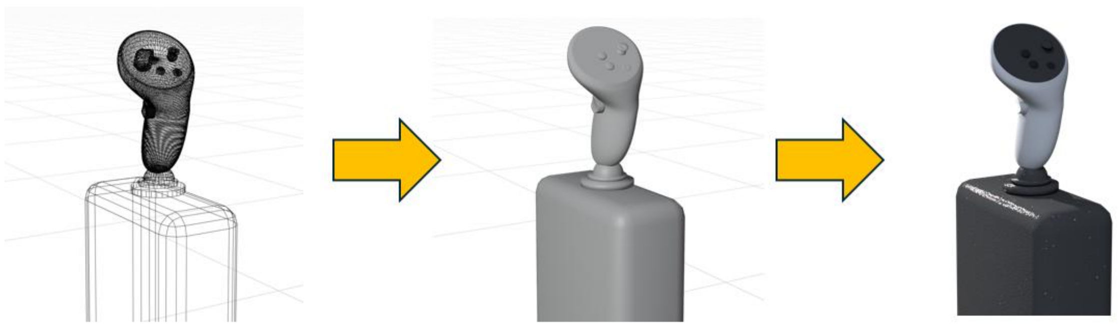
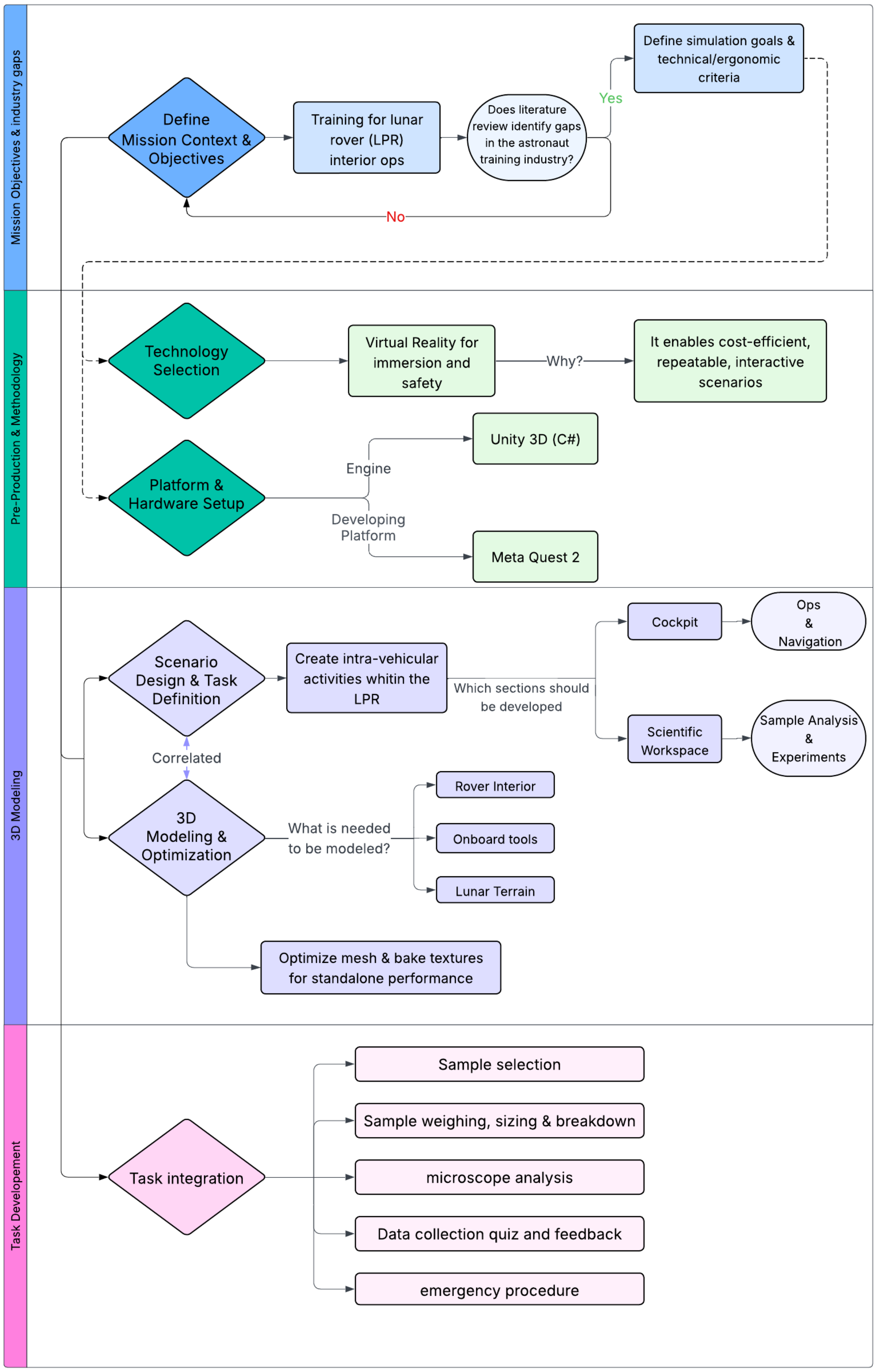


Figure 10. 3D Modelling Procedure of the driving controller

From an interaction standpoint, an effort was made to maintain a user-centered interface. Therefore, ergonomics studies have been carried out to give astronauts the possibility to focus only on the training and be less concerned with the managing of the VR systems. These studies focused on optimizing the positioning of the 3D models according to their real positioning, creating a sense of coherence between real-world gestures and virtual interactions. In this way, the interface was not only designed to be functional, but also to contribute positively to the overall sense of immersion.

Table 1. Methodology process for a LPR simulation design


Mission Objectives & industry gaps
Define Mission Context & Objectives
Training for lunar rover (LPR) interior ops
Does literature review identify gaps in the astronaut training industry?
Yes
Define simulation goals & technical/ergonomic criteria
No
Pre-Production & Methodology
Technology Selection
Virtual Reality for immersion and safety
Why?
It enables cost-efficient, repeatable, interactive scenarios
Platform & Hardware Setup
Engine
Unity 3D (C#)
Developing Platform
Meta Quest 2
3D Modeling
Scenario Design & Task Definition
Create intra-vehicular activities whitin the LPR
Which sections should be developed
Cockpit
Ops & Navigation
Scientific Workspace
Sample Analysis & Experiments
Correlated
3D Modeling & Optimization
What is needed to be modeled?
Rover Interior
Onboard tools
Lunar Terrain
Optimize mesh & bake textures for standalone performance
Task Developement
Task integration
Sample selection
Sample weighing, sizing & breakdown
microscope analysis
Data collection quiz and feedback
emergency procedure

Chapter 3 – 3D modelling

3.1 Introduction to 3D Modelling and Blender

For the generation of all the models and geometries the Blender software was utilized (*Figure 11*). Blender is an open-source 3D software used for modelling, animation, rendering, motion tracking, video editing, and game development. It was chosen because its interface is intuitive and allows a smooth workflow, making it easier to focus on the creative and technical aspects of the modelling process. Another practical advantage is its large and active community: countless tutorials, forums, and online resources are available, which helped in exploring different techniques to improve the models.

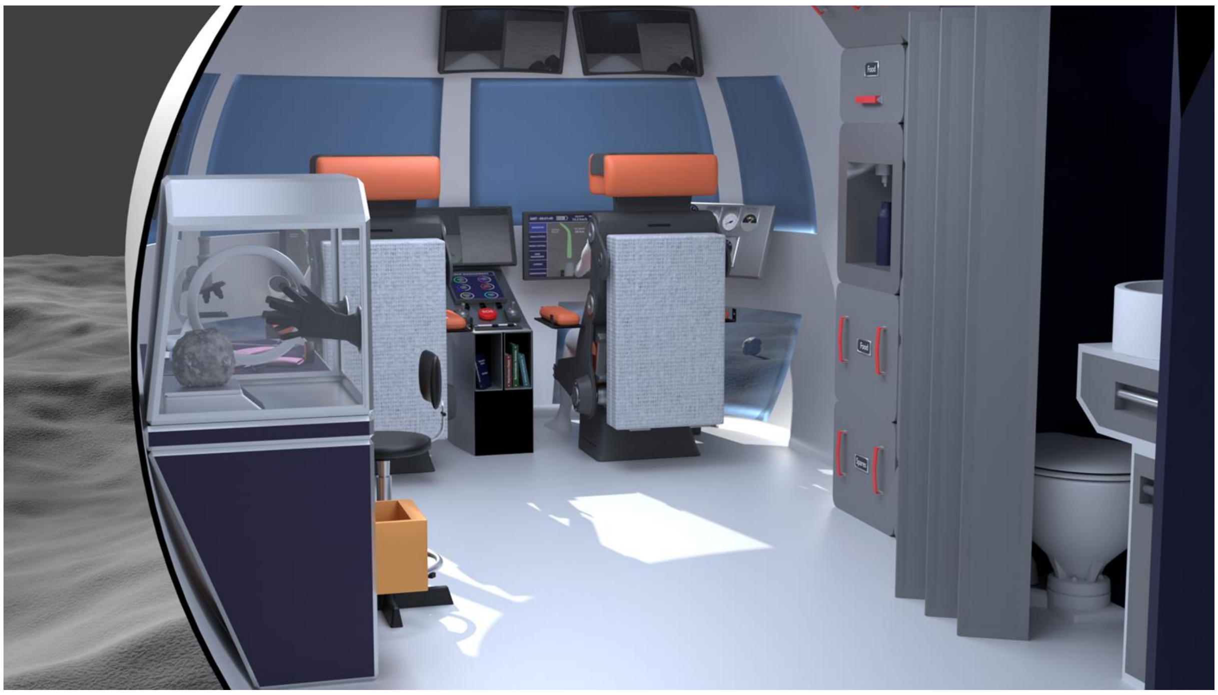


Figure 11. Blender render of LPR interior

3.2 Hierarchical diagram

The first design step consisted in conducting deep research on the configuration and internal layout of pressurized rovers, to better understand possible design solutions to be integrated into the simulation. This phase was fundamental for defining the spatial organization and functional logic of the environment.

As previously described in chapter *2.3*, the LPR was conceptually divided into five main sections: the Cockpit; the Scientific Workspace; the Crew and Utility area; the Airlock and EVA Section, and the Emergency Compartment. For each of these areas, a hierarchical diagram was developed to visualize the internal relationships between components, functions, and sub-elements within the 3D modelling process.

A hierarchical diagram (or hierarchical chart) is a schematic representation that organizes elements according to levels of dependency or function, starting from a main element and subdividing into smaller, more specific parts. In 3D modelling, it is used to define the parent-child relationships among different objects, allowing designers to establish logical connections. This structure not only helps maintain order within scenes but also supports the whole modelling process, because the developer can see all the functional connections of all the objects that will be in the simulation and can also define their level of importance in interaction terms. This approach allows the decomposition of a system into manageable subparts, ensuring design consistency and integration at multiple levels.

In the generation of the hierarchy diagrams a first differentiation had been made to subdivide even more the elements and the models. In fact, two different scales, each defining a specific characteristic that the element had to be modelled with, have been created.

- *Level Of Detail (LOD)*: This scale indicates how detailed the mesh of a 3D model should be in terms of mesh density and polygon count. Elements that are more frequently used or crucial for the simulation will have a higher level of detail compared to others. There are three levels, ranging from LOW to HIGH.

- *Level Of Interaction (LOI)*: indicates how much the user must interact with a certain model and manually execute tasks and commands with it. To be specific every model will have a LOI, depending on how much that element have to be interactable. For example, an element like the joystick controller in the cockpit section, which is used to drive the LPR, must have a really high LOI because it will be used for a lot of tasks and there will be different interactions that include that specific element. The LOI scale that was produced has 5 levels:

    - *Static (S)*: Describes elements that are not interactable by the user, so all the elements that only have a visual purpose (e.g. indicators, windshields, utilities).

    - *Dynamic (D)*: Describes elements that move within the simulation and that may be grabbed but still have almost no interest in the simulation in terms of interaction (e.g. monitors and cameras).

    - *Dynamic with low interaction (DLI)*: Describes elements that have a limited intractability, like buttons that only have to be pressed or selected to activate animations.

    - *Dynamic with interaction (DI)*: These are the main models that will be used to perform tasks. For this reason, they have to be more detailed than the other categories as they're crucial to enhance the realism of the simulation.

Obviously LOI and LOD are corelated since often an high level of interaction highlights the need for more details.

3.2.1 **Cockpit**

The cockpit section is one of the main sections of the LPR, since it is where all the multi-functional displays are located, and so where all the system are checked and controlled other than the place where navigation is carried out (*Figure 12*).

In the first iteration of the modelling process, it was decided to follow a more aeronautical set up and to imagine the cockpit of the LPR similar to an airplane one. For this reason, the idea was to have two pilot seats where the main pilot and a co-pilot could both drive the rover and also check that all the systems are working correctly. Subsequently, it was decided to adopt a design configuration featuring a single driving position on the right side and a secondary support and monitoring position for the second astronaut. This solution not only optimizes the available space by avoiding the duplication of navigation instruments but also aligns with a design philosophy inspired by terrestrial vehicles, which appears to be more suitable for an LPR.

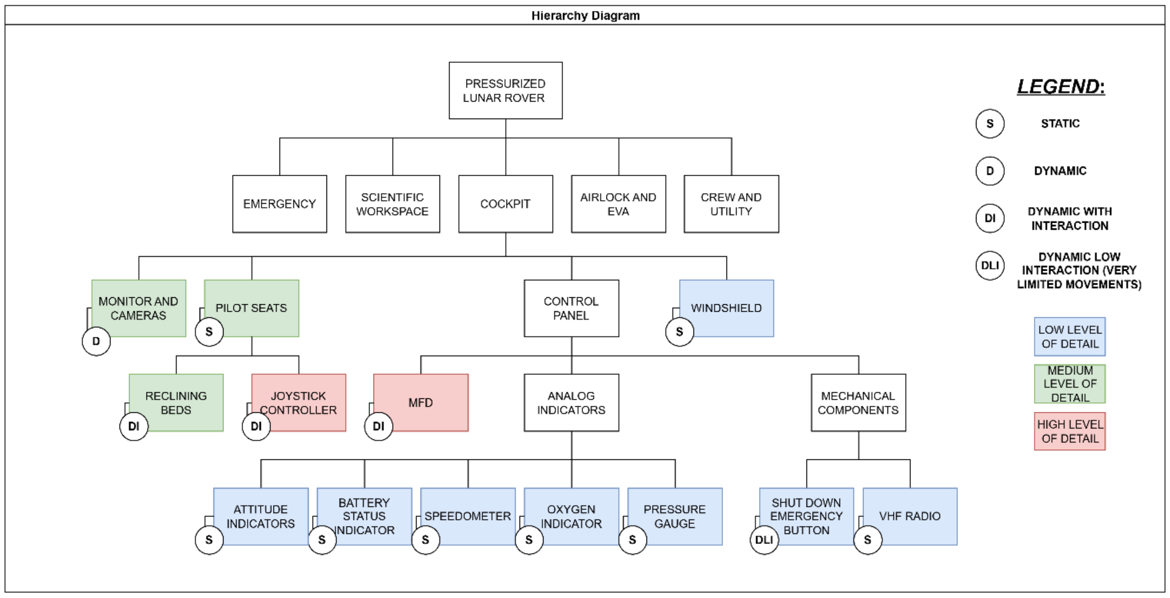


Figure 12. Cockpit hierarchy diagram

The cockpit section is equipped with two reclining seats designed to ensure comfort during interaction with the equipment. Both seats can be adjusted to a “night mode” configuration, allowing the two pilots to rest or sleep within the cabin during extended missions. The right pilot station hosts the controller for vehicle operation and navigation. Positioned directly in front of the seat is a main console with multiple display monitors showing live video feedback from the rover’s external cameras, which are essential for driving and assessing the surrounding terrain. These displays are fully customizable through the onboard computer system known as the Multi-Functional Display (MFD). Through the MFD, also available at the co-pilot station, operators can configure and monitor all internal digital systems of the Lunar Pressurized Rover (*Table 2*). The interface allows users to manage navigation and plan the LPR route, display real-time operational data, and monitor both environmental and mechanical conditions of the vehicle (*Figure 13*).

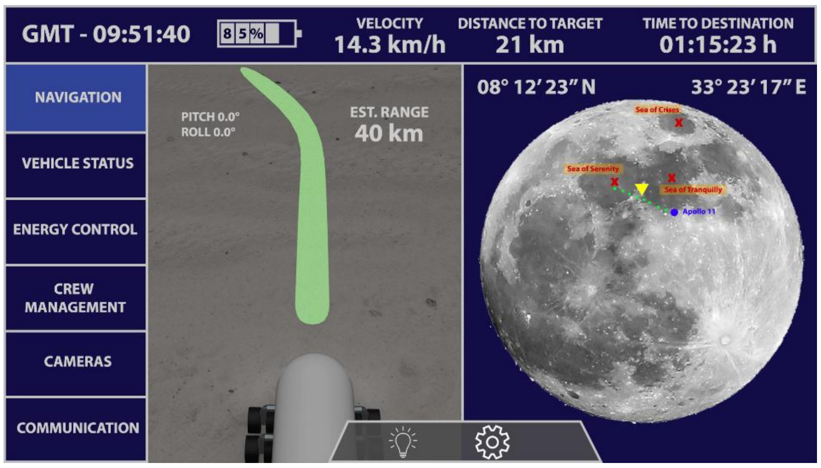


Figure 13. Example of MDF interface

For redundancy and safety, a lateral control panel located to the pilot's right includes a set of physical analogic indicators designed to provide critical data in case of system failure. These indicators display key parameters such as energy and battery management, pressure and oxygen levels, speed values and the status of the movement system. This set of instruments serves as an essential backup system, ensuring continuous monitoring and secure operation of the rover under extreme environmental conditions.

Table 2. Elements of Cockpit subsystems

System	**Digital elements**	**Analogic elements (redundant for safety)**
Navigation	Interactive 3D map, waypoints, telemetry.	Speedometer.
Driving Control	Joystick with haptic feedback (reacts to external stimuli, e.g., vibration), software assisted.	
Energy Management	Monitoring via touchscreen (batteries, solar cells, consumption).	Physical indicators for battery status.
Internal Environment	Touchscreen for temperature, pressure, and oxygen regulation.	Physical gauges for pressure and O_2 level.
Communication	Digital system with voice interface.	Manual UHF/VHF radio with selectors.

Safety & Emergency	On-screen alerts.	Physical buttons for manual shutdown.
External Monitoring	Monitors displaying different views from cameras for external monitoring.	
Movement System (Suspension, Steering, Terrain)	Digital sensors (displaying wheel steering angles and status).	Attitude angle indicators.

In terms of ergonomics, it has been studied how to collocate all the buttons and monitors in a way that astronauts can be comfortable in reaching everything during his tasks. In VR simulations it's important that the UI is simple to let the user focus on the training activities. To achieve this, in this modelling part, a rigged 3D body of a human has been created and utilized as a first element to adapt the position of all the models (*Figure 14, Figure 15*). Through this rigged body it was possible to see if the screens and the controllers were not too far from the user and also it permitted us to slender the design of the control panel. Further ergonomic studies will be conducted after the VR testing in Unity, to receive direct feedback on how the simulation responds to the user's inputs and on the fluidity of the interactions.

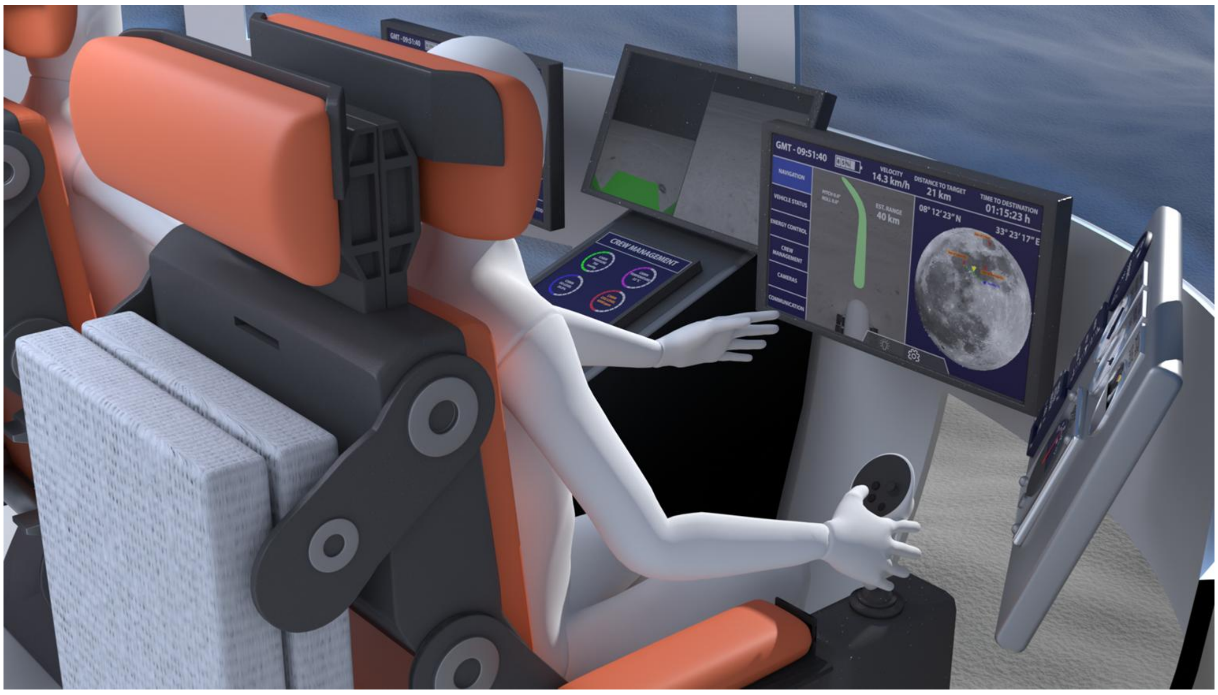


Figure 14. Cockpit pilot seat Blender render

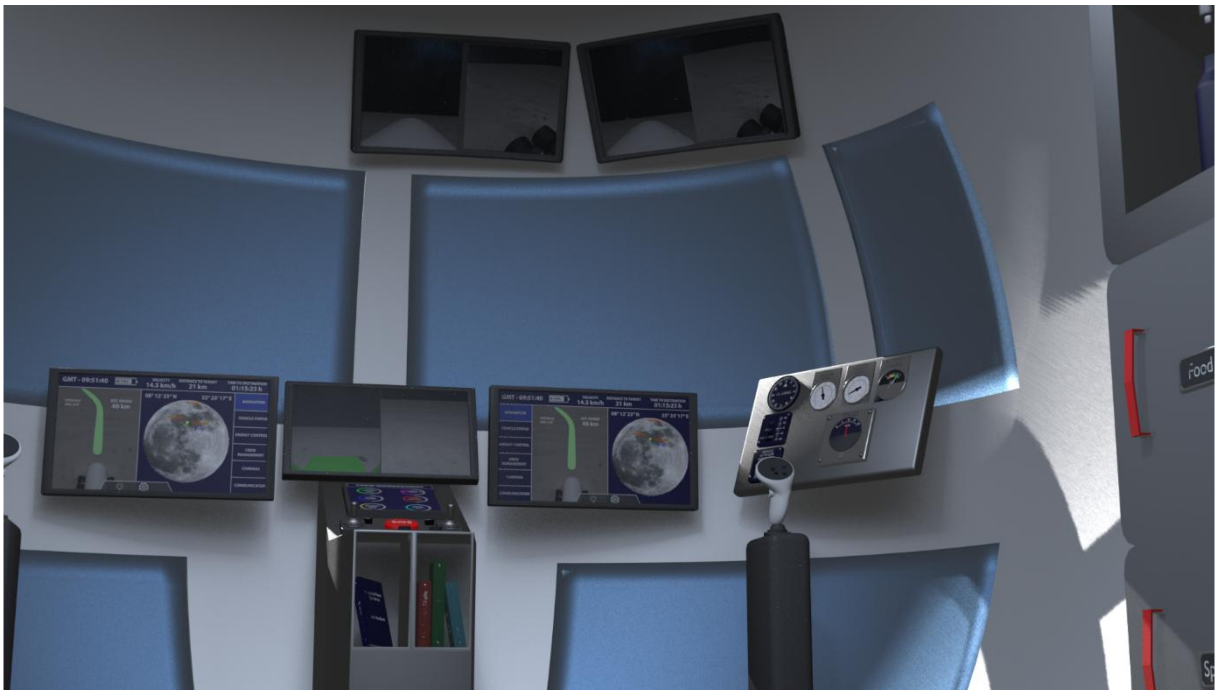


Figure 15. Cockpit configuration Blender Models render

3.2.2 **Scientific Workspace**

The scientific workspace section is the second main section of the LPR. Here the user will be able to perform various tasks onboard, working on and analyzing the lunar regolith samples with all the instruments present on the workbench (*Figure 16*).

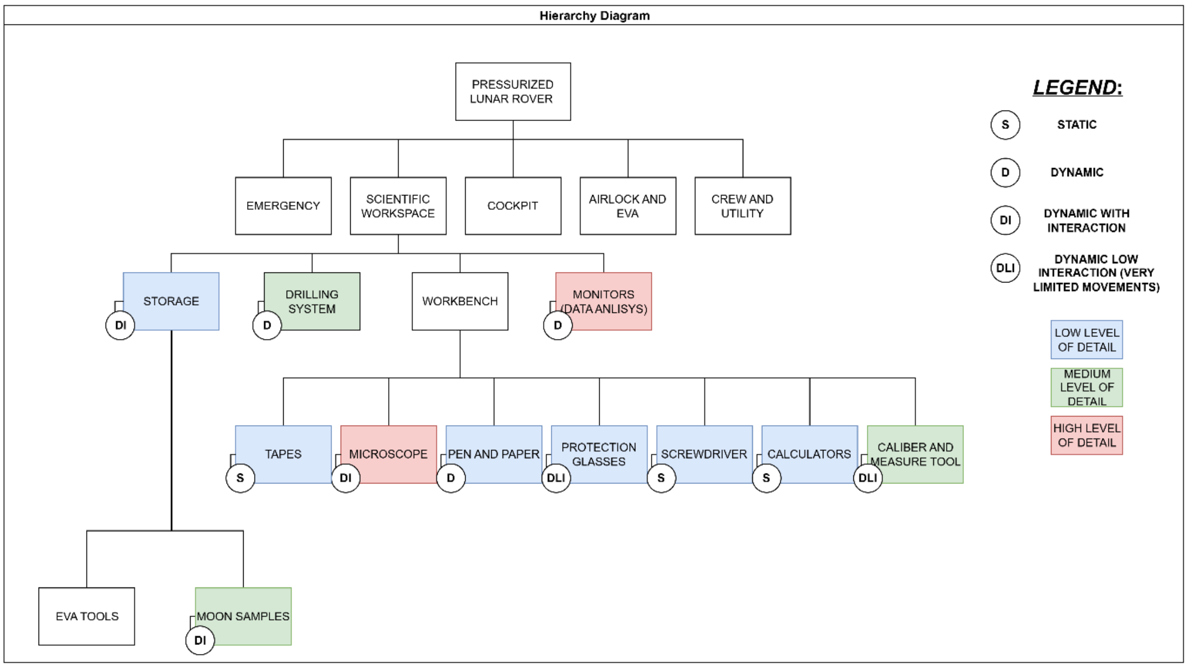


Figure 16. Scientific Workspace hierarchy diagram

On the workbench, various items which are useful for conducting experiments (*Figure 17*) such as screwdrivers, measure tools and a microscope, have been modelled alongside with a monitor that will show information and data during the task execution. The microscope is a petrographic polarizing one for thin-section analysis in transmitted light to observe the optical properties of the lunar samples. The virtual instrument reproduces the key optics of a geological microscope. Images are shown in plane-polarized light (PPL), used to observe relief, grain boundaries and fractures, and in XPL, which reveals interference colors, twinning and crystal orientations.

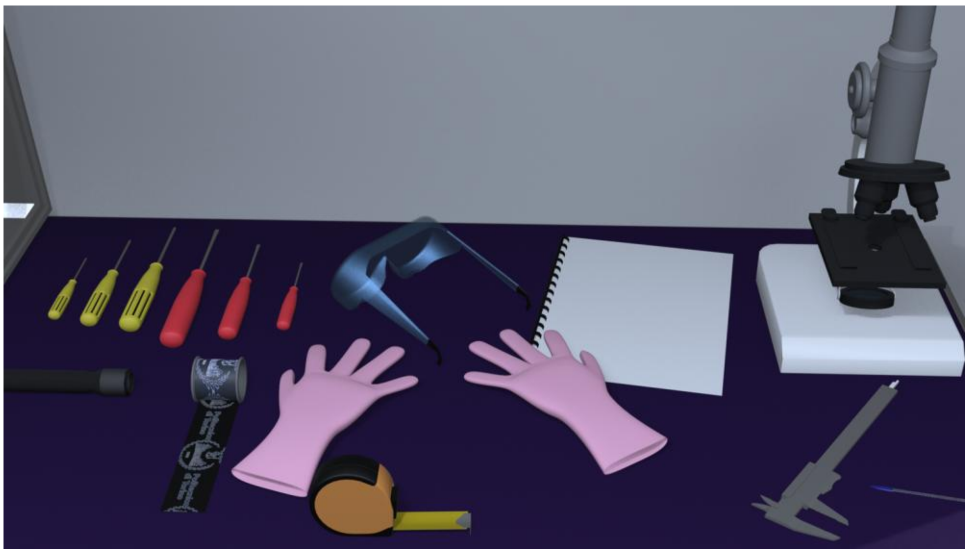


Figure 17. Workbench instruments

In addition, a virtual storage that contains different lunar samples has been designed. These samples will then be analyzed within a sealed chamber (or sealed box), a small enclosed environment equipped with gloves that allow operators to handle materials and conduct experiments without risking contamination of the rover's internal surroundings (*Figure 18*). This setup was selected because the lunar surface contains dust particles, making it essential to minimize their dispersion and prevent any interference with rover's instrumentation as well as to mitigate the risk that astronauts inside the LPR might inhale this dust.

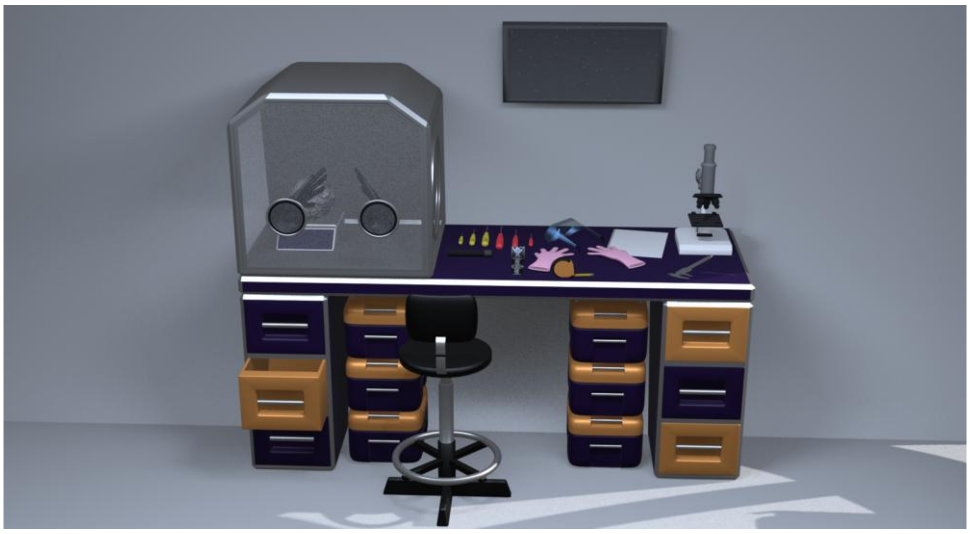


Figure 18. Sealed box and Workbench

Even if we decided to focus on the internal configuration of the LPR and not to simulate an eventual Extra Vehicular activity (EVA), in the designing process we made some low polygons models of some EVA tools such as tongs, rakes, scoops and even an electric drill for the collecting of sample on the lunar surface. These models could be integrated in a possible future development that integrates this kind of operation to expand and make the simulation more complete.

In the modelling of this section, particular attention was given to the lunar samples, as they represent the principal element of all experiments conducted within the scientific workspace. To make these experiments more engaging and realistic, a highly detailed texture of lunar regolith was created from scratch through Blender's texture mapping technique (*Figure 19*). In Blender, texture mapping involves projecting a two-dimensional image (the texture) onto the surface of a three-dimensional model to simulate the material's appearance, including its color, roughness, and fine surface irregularities. This process allows the digital surface to reproduce the visual complexity of real lunar soil, enhancing realism of the virtual environment.

Accurate and well-crafted textures are particularly important for highly interactive elements, as they influence not only the visual perception of realism but also how light, shadows, and depth cues are perceived by the user. These factors can significantly affect immersion in virtual simulations. However, adding too detailed textures could make the simulation much heavier, which may cause lag or slowdowns during tasks, especially on standalone VR headsets that have to process everything independently. So, in the modelling process it's always important to make a compromise depending on the number of models and the overall size of the virtual environment, priority should be given to the most dynamics elements and the ones that will be frequently interacted within the simulation.

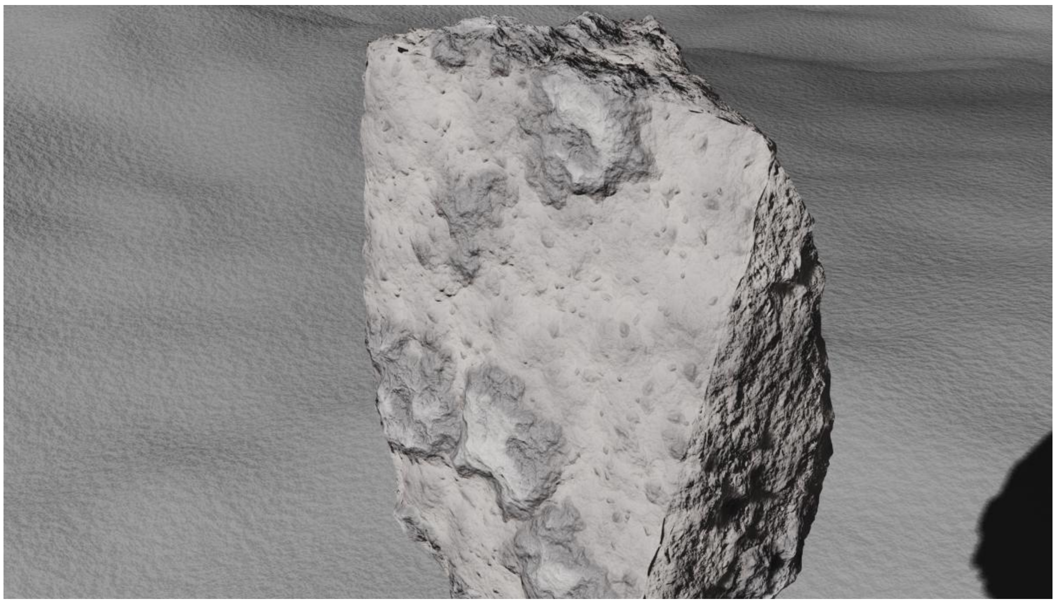


Figure 19. Texture of a moon sample

3.2.3 **Airlock and EVA**

The airlock and EVA section is divided into three main zones (*Figure 21*). The first consists of the inner hatch, which separates the rover's habitable area from the depressurization chamber. Inside this chamber is the outer port, equipped with a handle that allows the astronaut to exit the rover once the depressurization procedure has been completed. To carry out this procedure, the astronaut must enter the chamber wearing the space suit (*Figure 20*) with the Life Support System (LSS) already activated. After ensuring that both hatches are securely closed, the depressurization process can be initiated. Once the chamber reaches the external pressure level, the astronaut can open the outer port and exit the rover to begin the Extravehicular Activity. The re-entry process follows the same logic but in reverse: after completing the EVA, the astronaut returns to the chamber, closes the outer port, and initiates the pressurization procedure before opening the inner hatch and re-entering the rover's habitable area.

The second is referred to the space suit locker, where the astronaut will have the possibility to wear his suit during emergency procedures or to carry out the Extra Vehicular Activities.

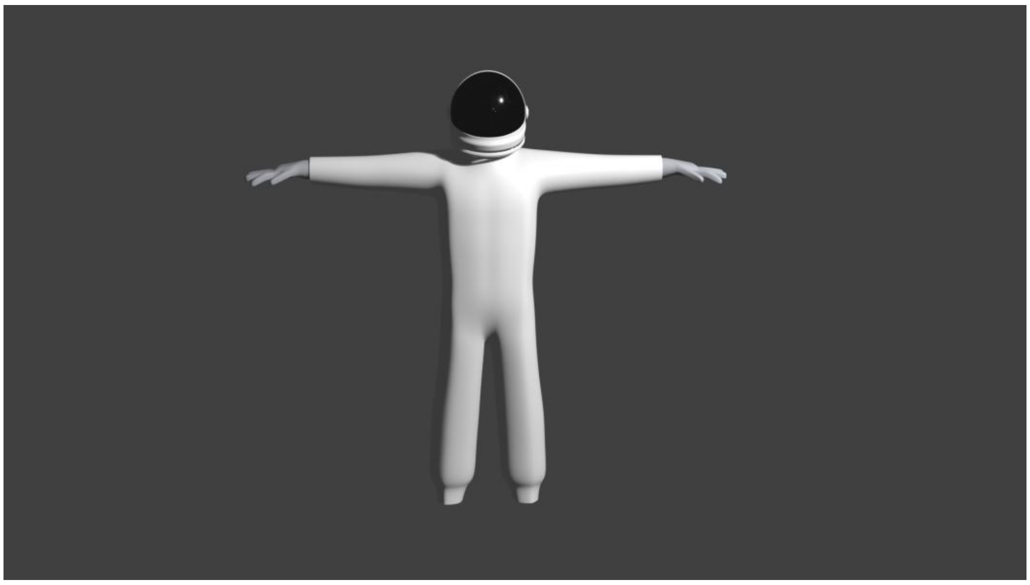


Figure 20. Space suit basic model

The third zone regards a dust removal zone, equipped with a brush and a vacuum cleaner, to wipe out all the lunar dust that could remain in the suit after an external operation. In fact, lunar soil contains a significant amount of fine dust that can easily adhere to the equipment and that could be a potential health hazard if inhaled or brought into the pressurized cabin. The fine lunar regolith particles are sharp, and their inhalation could cause respiratory irritation or long-term damage to the lungs.

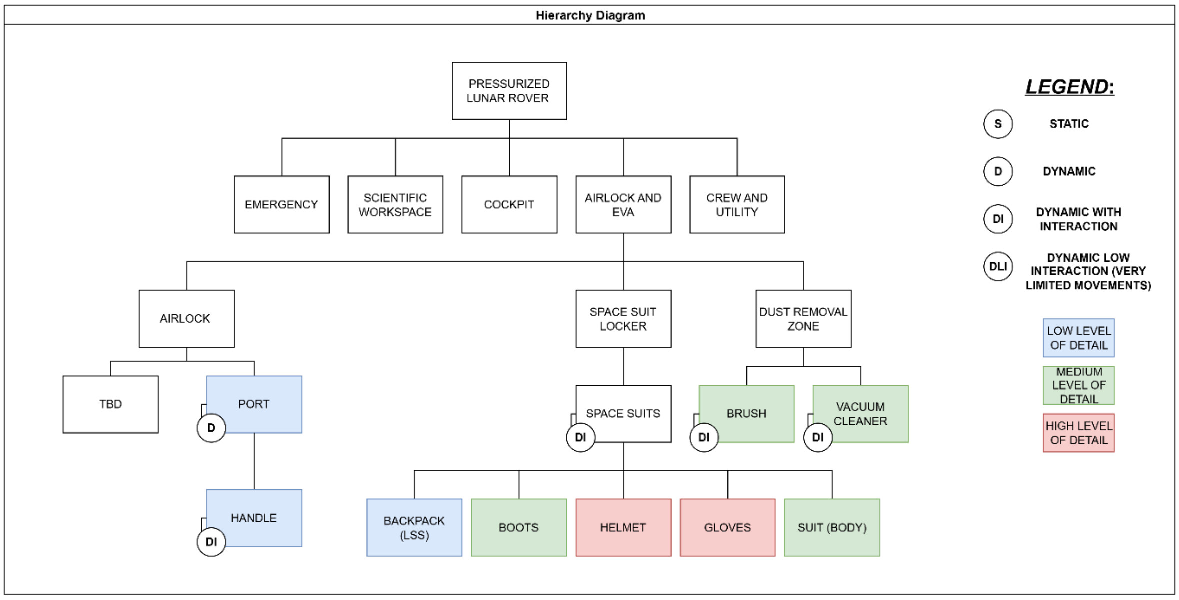


Figure 21. Airlock and EVA hierarchy diagram

3.2.4 **Crew and Utility**

The crew and utility area encompasses all the elements that enable astronauts to live and operate comfortably within the rover. This section includes a zone for personal hygiene, a

galley area dedicated to food storage and preparation, and provisions necessary for survival during mission operations (*Figure 22, Figure 23*).
It also features a small exercise zone, typically equipped with a “cyclette” or a cycle ergometer. Regular physical exercise is essential for astronauts in space because the absence of gravity leads to muscle atrophy and bone density loss over time. Engaging in daily workouts helps maintain health, muscular strength, and a good overall physical condition, ensuring the astronauts remain fit and capable of performing demanding tasks during the mission. Finally, the area includes also general storage compartments for tools, scientific instruments, and collected samples used for experiments and analyses.

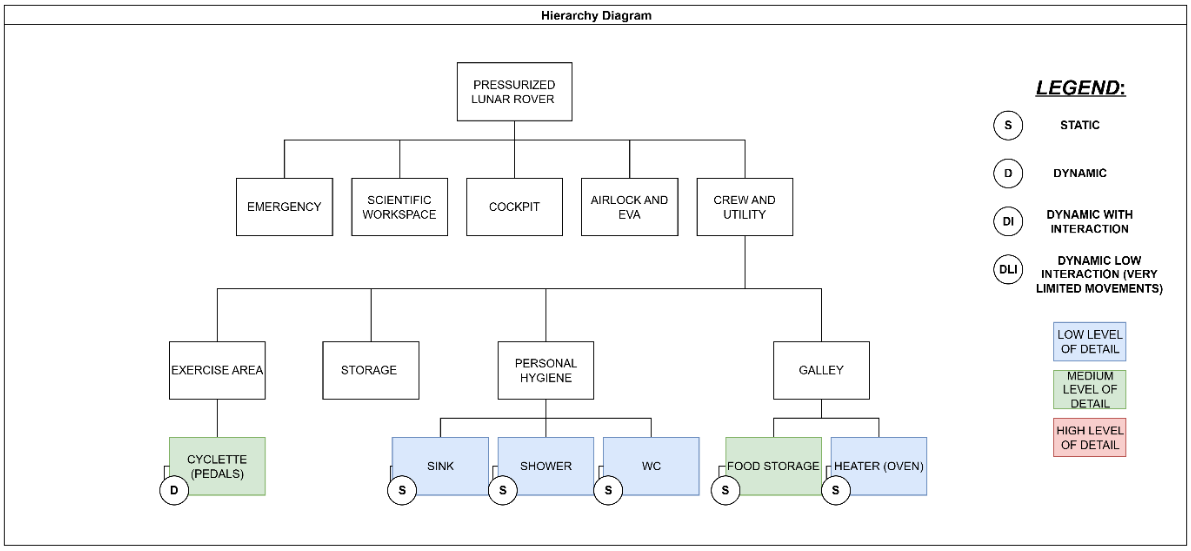


Figure 22. Crew and utility hierarchy diagram

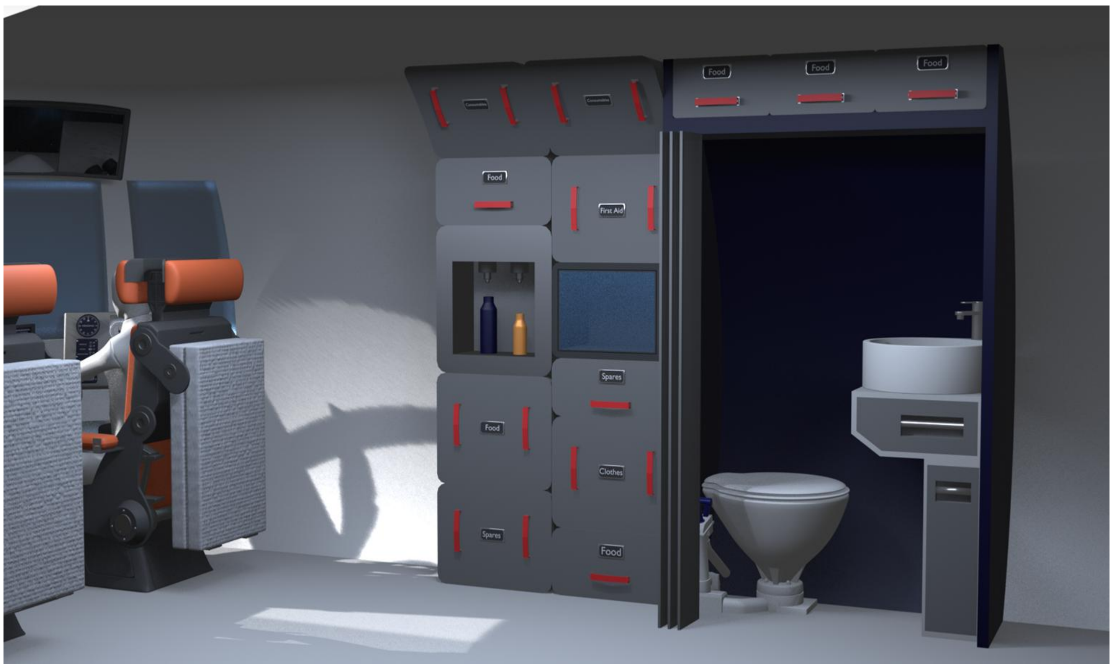


Figure 23. Utilities models

3.2.5 **Emergency**

Regarding the Emergency section, it has not been treated and developed through this 3D modelling section. The decision comes from the fact that 3D models, in this section, are not functional to the simulation process. In fact, this section has been conceived more as something related to the emergency procedures to be carried out within the tasks.

For this reason, a more detailed and practical discussion of these aspects will be provided in

Chapter 4 – Unity implementation and game development, where the Emergency section will be analyzed more concretely in connection with the operational tasks within the various sections.

3.3 Texture Baking on Blender

In this project I decided to use the baking technique to exploit the moon samples texture that I created through the UV mapping of Blender. When you are working with 3D models like the lunar rocks that must have high details, it’s often necessary to reproduce complex surface textures that convey roughness, shadows and bumps. In Blender, these effects are typically generated through procedural shaders, which use mathematical noise and displacement functions to simulate irregularities rather than storing them in image files and then applying them on an object. Procedural materials are extremely flexible and visually impressive, but they have one major limitation: they exist only inside Blender’s shader system and cannot be directly exported to real-time engines such as Unity. This is where texture baking becomes essential, indeed, baking converts the information produced by a procedural shader (color, normal direction, roughness, local shadows) into static 2D texture maps that can be read by any engine. In practice, baking records the appearance of the high-detail material onto the UV map of the 3D mesh. The result is a set of image textures (for example albedo, normal, roughness, and ambient occlusion maps) that visually preserve the richness of the procedural material, but without the heavy computation required at runtime.

For models such as lunar soil or moon rock samples, this process is important since these materials relies on fine-grained displacement and noise patterns difficult to reproduce and by baking the procedural shader to textures, the visual fidelity of the high-resolution Blender material is maintained without weighing down the simulation on Unity.

Once the textures are exported (in PNG format), they can be easily connected to a Unity material using the Standard (PBR) shader:

- ❖ Import the baked texture files into Unity’s Assets folder.
- ❖ Create a new Material and assign the baked albedo map to the Base Color slot, the normal map to Normal, the roughness (or smoothness) map to the Metallic/Roughness channel, and optionally the ambient occlusion map to its corresponding slot.
- ❖ Apply this material to the imported mesh.

The result is a lightweight 3D object that visually matches the procedural version rendered in Blender.

3.4 Concept of models and references

The reference collection phase began with an in-depth analysis of existing designs and mission concepts related to human-rated lunar vehicles. Designs such as NASA's Space Exploration Vehicle (SEV) for pressurized module dimensions and cockpit arrangement, ESA's Lunar Habitat studies, and the Lunar Gateway interior mock-ups were examined to understand how space agencies approach spatial layout, functionality, and ergonomics in confined environments. In particular, we explored mock-ups such as the ISS and Orion capsule. Observing how these environments are organized offered hints on how to arrange controls, storage, and utility areas efficiently within limited volumes. This process helped ensure that the simulated environment followed credible human factors principles.

A key principle during concept definition was to balance technical realism with computational feasibility. While the models were inspired by real aerospace equipment, their complexity had to remain manageable for real-time rendering in a VR environment. To achieve this, visual details that did not affect the user's understanding or interaction were simplified, while those enhancing immersion such as indicators and textures have been preserved.

For the cockpit section the main reference utilized for making the models are NASA's SEV, which offered insights on seat configuration and windshield placement. The idea of combining a main pilot station with a secondary support and monitoring position originated from the terrestrial vehicles design, since an LPR is totally different from an aircraft, where one operator handles navigation and the other oversees subsystems and mission parameters. The analogic indicators in the control panel were designed as a redundancy system and have been inspired by the typical aeronautical ones, in a modified way to make them suitable for a space structure like the LPR.

The scientific workspace references focused on modular laboratory environments aboard the International Space Station. The sealed chamber for sample analysis was inspired by gloveboxes used on the ISS and in terrestrial clean-room laboratories, allowing astronauts to manipulate lunar material safely without spreading dust or contaminants within the rover. Some instruments on the workbench, even if not needed for performing tasks, are useful to give visual feedback of what can possibly be present in a LPR scientific workbench in the upcoming lunar missions.

In the airlock area, the inclusion of a dust removal area, equipped with a brush and vacuum system, was conceptually inspired by studies conducted during the Apollo missions, where astronauts reported significant issues with lunar dust contamination. Meanwhile the utility section configuration was inspired by space habitat design studies focusing on long-duration missions, particularly the Lunar Gateway and the storage of the ISS.

3.5 Pressurized Rover Design Principles and Data

3.5.1 Internal design principles

Pressurized rovers must sustain crewed operations over extended traverses while preserving safety, situational awareness, and operational efficiency. Typical constraints include limited mass and volume budgets, tight energy envelopes, robust life-support integration. We adopt a functional zoning strategy that groups tasks by areas:

- *Forward zone (Cockpit):* here you can perform navigation and systems supervision near to each other to minimize reach and head-turn effort during driving. Our cockpit consolidates primary controls at the pilot's right hand and relocates secondary supervision to a support position to avoid duplicating instruments.
- *Mid-cabin (Scientific Workspace and Utility):* a dedicated bench concentrates sample handling, measurement, and data display, isolating dust-sensitive tasks from driving operations. On the other side all the utilities are positioned, with everything that's needed for everyday life.
- *Back zone (Airlock & EVA):* the egress path is isolated by a depressurization chamber and a dedicated dust-removal area to prevent contamination of the habitable zone during re-entry. Also, this is where the space suits are located.

This configuration reflects the hierarchical decomposition and supports a clean separation of workflows within the VR scene. Summing up, the areas are organized like this:

- *Cockpit*: One primary driving seat on the right with joystick controller; a secondary station for systems supervision; multiple cameras for external; MFD with all rover subsystems configuration; analogical backups co-located on the pilot's right.
- *Scientific Workspace*: Tool bench with measurement instruments, microscope, other instruments for system maintenance and a local monitor, plus a sealed glovebox for regolith samples analysis.
- *Airlock & EVA*: Inner hatch, depressurization chamber, outer port sequence, with the dust-removal area positioned on the EVA return path; The space suit area.
- *Crew & Utility*: Galley, hygiene, storage, and compact exercise equipment.

This organization has benefits in VR training: users spend less time moving around to perform the tasks in different areas and it permits to divide the task design work into modules, allowing us to act on the application in the future and add tasks or delve into different scenarios and improve the simulation.

3.5.2 LPR vessel design and dimensions

Pressurized rovers benefit from curved geometries (cylinders with rounded/hemispherical endcaps) because they minimize stress concentrations. The structural aspect is crucial, as different geometries lead to varying stress distributions within the structure when pressurized. In addition, the geometry must ensure efficient use of the internal volume, correct placement of the center of mass, and the ability to accommodate the module within the launch vehicle. Being designed as pressurized and habitable, the module must maintain an internal atmosphere of approximately *1 [atm]*. Due to the vacuum conditions outside, its structure is subjected to a pressure difference of about 1 atmosphere. The maximum pressure differential is therefore: $\Delta p = 0.1048\ [MPa] = 1.0343 [atm]$, consequently, these structures must be designed to withstand this pressure difference. We adopted a single-cylinder monocoque pressure vessel (*Figure 24*), with a rounded nose to reduce local peak stresses around the windshield. The dimensions of the LPR have been decided and discussed after a literature review. In the end the geometry consisted in a cylinder with a diameter of *3.5 [m]* and a total length of *7.1 [m]*. To keep the Quest headset at stable framerates, the shell is modelled as a single continuous pressure skin, but more details can be added in the future. Since the *RoVR* simulation is focused on virtual reality rather than on the physical design of the rover itself, the various subsystems were not dimensioned, and no stress-field calculations were performed on the cylindrical shell. However, it is important to keep in mind that a design as realistic as possible can make the simulation more credible and, consequently, improve the overall efficiency of the training.

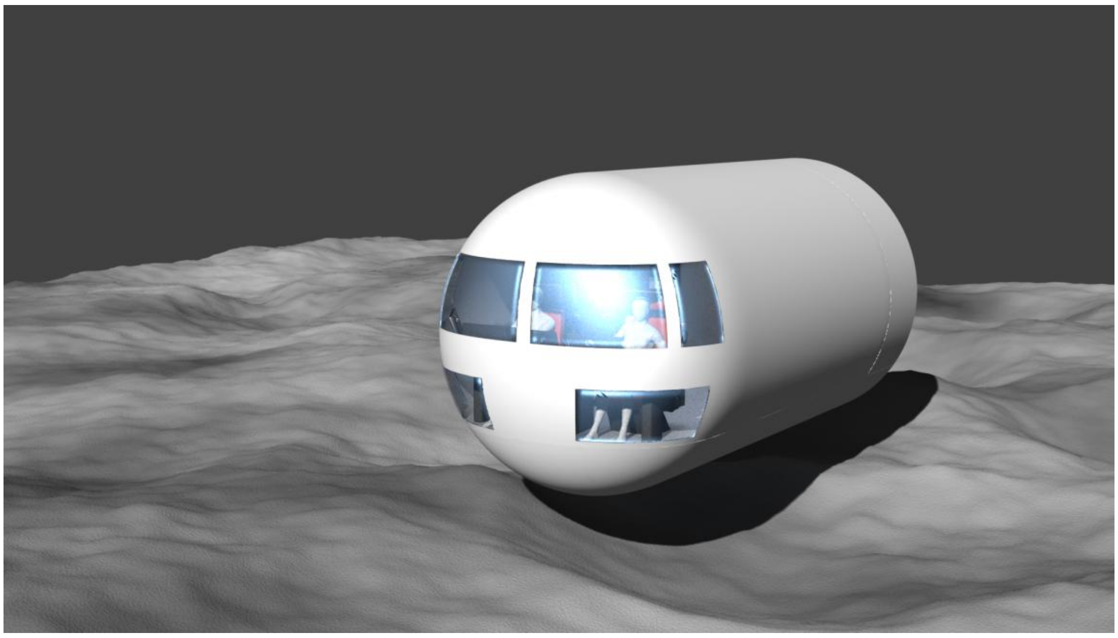


Figure 24. Simplified pressure vessel 3D model

Chapter 4 – Unity implementation and game development

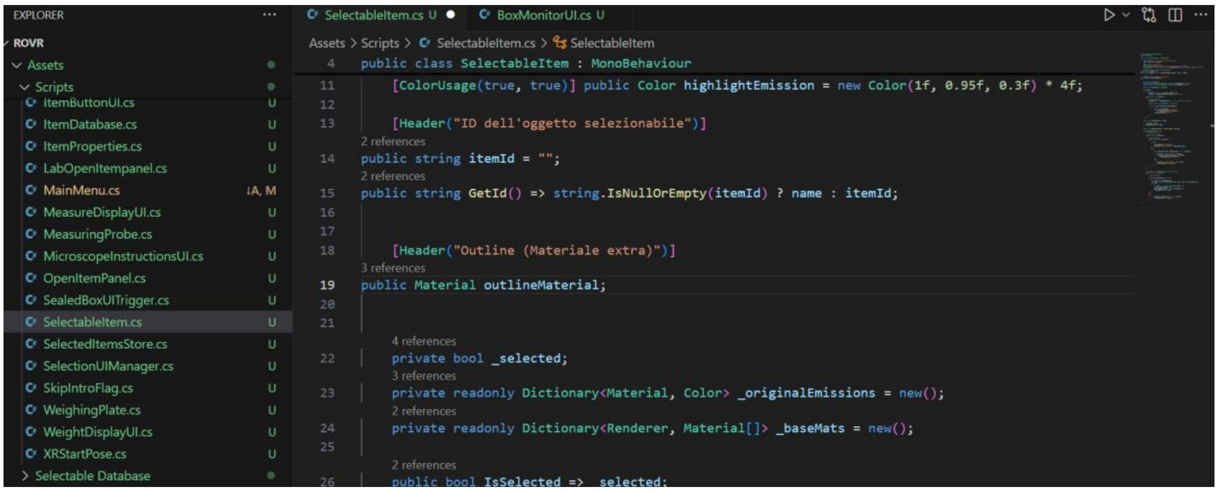
4.1 Introduction

When developing a VR based simulation, one of the first things a developer must decide is the game engine to utilize. In most cases, as explained in the previous chapters, the main development platforms utilized to produce a VR game or project are mainly two: Unity 3D Engine and Unreal Engine.

At the beginning of this project, we selected Unity as the development engine because of its versatility, stability, and extensive support for virtual reality development. It provides an environment where assets, scripts, physics and user interaction coexist are integrated to obtain an easy and well-organized workflow.

4.2 Unity Overview

One of Unity's major strengths is its component-based architecture. Each element in a scene is represented as a GameObject that can hold multiple components, each defining a specific property. A light source, a collider, or a script can all be attached independently, enabling the creation of dynamic systems. It also provides a powerful rendering pipeline with customizable options to balance realism and performance. For this project Universal Render Pipeline (URP) has been utilized, allowing advanced lighting, shadow casting and good material rendering. For the coding part we used C# scripting language, connecting scripts by functions (such as input handling, interactivity, and scene management), and attaching them to the various GameObjects as needed. To write in C#, it's possible to use Visual Studio Code, an efficient code editor integrated with Unity (*Figure 25*). It allows real-time error detection, syntax highlighting, and automatic compilation of scripts as soon as they are saved simplifying the development workflow, as any modification to a script is immediately reflected in the Unity Editor, making debugging and iteration much faster. The majority of scripts created are structured to handle a specific task. For example, controlling the movement of objects and their reaction to the rest of the environment, managing user inputs from the VR controllers, or activating events when an object is selected. Any GameObject in the scene can easily be modified or expanded simply by attaching new scripts or changing component parameters, without altering the rest of the system. This makes Unity a powerful tool for creating scientific and educational simulations.



```
public class SelectableItem : MonoBehaviour
    [ColorUsage(true, true)] public Color highlightEmission = new Color(1f, 0.95f, 0.3f) * 4f;

    [Header("ID dell'oggetto selezionabile")]
2 references
public string itemId = "";
2 references
public string GetId() => string.IsNullOrEmpty(itemId) ? name : itemId;


    [Header("Outline (Materiale extra)")]
3 references
public Material outlineMaterial;


    4 references
    private bool _selected;
    3 references
    private readonly Dictionary<Material, Color> _originalEmissions = new();
    2 references
    private readonly Dictionary<Renderer, Material[]> _baseMats = new();

    2 references
    public bool IsSelected => _selected;
```

Figure 25. Scripts produced through C# in Visual Studio Code

XR Interaction Toolkit is a framework developed by Unity to simplify the creation of interactive experiences for XR applications. We utilized this toolkit because it defines standardized components for common actions such as grabbing and selecting elements. It also manages by itself how users interact with objects and interfaces inside immersive environments. The main features that are in the toolkit are:

- ❖ *XR Rig*: represents the player's presence in the virtual world, including the camera (headset position) and controller anchors (hand positions).
- ❖ *Interactors*: the components that perform actions (Ray/Poke/Near-Far Interactors and Grab Interactable Components).
- ❖ *Interactables*: the objects that can be manipulated or triggered by the player (tools, buttons, canvas).
- ❖ *Interaction Manager*: a central system that coordinates which interactor is targeting which interactable at any given time.

Through XR Interaction Toolkit, Unity allows direct deployment to major VR platforms, including Meta and HTC devices. The toolkit provides preconfigured systems for locomotion, grabbing, and UI interaction, reducing the time needed for prototyping.

4.3 Unity Learn Pathways

Before starting the actual development, I've tried to build a solid understanding of Unity's features and development workflow through the Unity Learn Pathways platform. Unity offers these free structured tutorials called "Learn Pathways" (*Figure 26*) that let developers to fully immerse in the game and simulation development. In particular, three official Pathways has been really important to understand better the simulation design: “Essential”, “Junior Programmer” and “VR Development”. Taken together, these three pathways are a comprehensive learning foundation that combines design, programming, and Virtual Reality implementation.

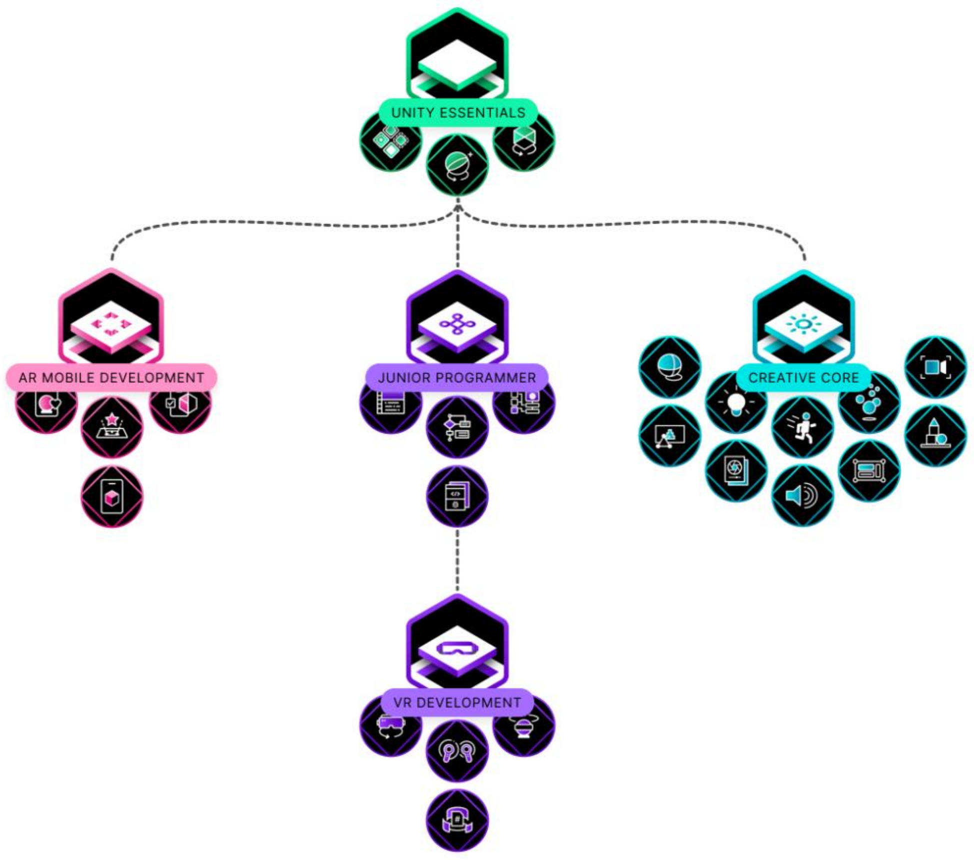


Figure 26. Unity Learn Pathways

4.3.1 **Essential Pathway**

The Essential Pathway served as an introduction to Unity’s core concepts and workflow. It covered the fundamental elements of scene composition, asset management, lighting, and physics, providing an overview of how to navigate the Unity Editor. It introduced key concepts such as the GameObject-Component architecture, the Transform hierarchy, and the Prefab system, and through practical exercises it also explored basic C# scripting. In addition, the pathway gave us helpful tips on good development practices, such as naming conventions, folder organization, and iterative testing. All these elements were really important when structuring the LPR project, ensuring that scenes, assets, and scripts were logically separated and easy to navigate and organize.

4.3.2 **Junior Programmer Pathway**

The Junior Programmer Pathway focused on developing strong C# programming and problem-solving skills within the Unity environment. It provided a systematic progression from basic syntax and logic to object-oriented programming concepts applied directly to game mechanics. It has also covered the use of Unity's API documentation and Visual Studio integration, teaching how to navigate between the editor and the codebase effectively. The main topics of the course are data types, loops, event-driven programming, arrays, class structures, inheritance and debugging tools.

4.3.3 **VR Development Pathway**

The VR Development Pathway was the most relevant for the goals of this thesis. It provided a structured approach to designing and implementing immersive experiences using Unity's XR Interaction Toolkit and OpenXR framework. Furthermore, it shows fundamentals of setting up XR projects, including the configuration of the camera rig, headset tracking and input systems for controllers and explains key principles of spatial interaction design, such as the proper use of scale, ergonomics, and motion comfort. Some of the important features that are shown here, and that I found really helpful for the project are object manipulation (grabbing, rotating, scaling in 3D space), teleportation and locomotion systems, user interface interaction through ray casting and hand-tracking and haptic feedback and sound design for immersion. Another discussed topic was performance optimization for standalone VR devices. The tutorials explained how to reduce draw calls, use baked lighting, and optimize shaders.

4.4 Version Control with Git and GitHub

To organize the work well during the project and be able to integrate the scientific workspace simulation experience into the final application we used a version control. Version control is a system for tracking and managing changes to code and digital assets over time, enabling us to record a complete history, compare revisions, branch experimental ideas and revert when needed. So, when I'm working on the scripts of the scientific workspace, version control permits me to not interfere with the work done for the cockpit section by other developers and at the end I will be also able to put my work together with the rest to complete the *RoVR* application.

For this project we utilize Git as version control system. Git is a distributed version control system that stores work as a series of commits on branches and supports merging those lines of development. We also used GitHub, a cloud platform that hosts Git repositories and adds collaboration tools and let us perform pull requests, code review and give permissions for the folders. So basically, to use this version control the first thing that we had to do was set up a link between a local folder, where the project will be saved and updated every time, and the Github repository. To do this I first created an empty local folder and an empty repository on GitHub to serve as the remote source of the link. In the local folder I opened a terminal (cmd) and initialized an empty Git history with "*git init*" by creating a hidden "*.git"* directory. Then I use the command "*git clone <URL>*" that creates a local copy of a remote repository from GitHub and automatically sets the remote origin to that URL (Uniform Resource Locator). After moving into the cloned repository directory, I added all current files with "*git add .*", that inserts all new and modified files in the current directory tree for the next commit (the dot means "everything from here downward."). Then I checked whether a main branch existed and was active using "*git branch*" and then I created one with "*git checkout -b main*". I recorded and commented the initial project state with "*git commit -m <initial commit>*" and shared it to GitHub using "*git push -u origin main*", where -u sets main as the default upstream for future pushes and pulls. From that point on, adding content follows a simple loop: copy new or updated files into the repository folder, stage them with *git add .*, capture a meaningful snapshot with *git commit -m "<descriptive message>"*, and synchronize the history with git push (or pull if I wanted to obtain new files).

4.5 Device Simulator

In this project we utilized Unity's XR Device Simulator. This tool allows the experience to be exercised directly in the Editor Play Mode without connecting a headset, which is especially valuable on computers that do not meet the Meta Quest Link requirements. The Device Simulator maps headset pose and controller actions to mouse and keyboard inputs, so it is naturally limited: certain interactions, haptics, and depth cues cannot be replicated, and the sense of presence is not comparable to a real device. Nevertheless, it provides a fast, reliable way to check logic, UI flow, and scene integration and permitted me to do a fast debugging without the need of connecting the VR headset with a cable Quest Link for every little change of the scripts during the development process.

4.6 Lunar Environnement

I built a simple but realistic lunar environment:

(1) A "*stardome*", a huge, inverted sphere (normals flipped) centered on the scene, with an Unlit material and a starfield texture on the Base Map so the sky and the stars stay still and non-reflective.

(2) A lunar terrain imported from Blender: a low-poly mesh with a normal (bump) map), PBR material with grey albedo, moderate normal strength, and tiling scaled to scene units; for physics I used a *MeshCollider*.

(3) An "Earth" 3D sphere GameObject with an Earth albedo texture and a touch of emission.

4.6.1 Gravity Setting

By default, Unity projects use Earth gravity. To simulate the Moon, I opened Project Settings and Physics and adjusted the Gravity vector so that the Y-axis (Unity's vertical) is roughly one sixth of Earth's value, about 1.63 [m/s^2] if Earth is −9.81 [m/s^2]. This gives interactions a weight and response that feel consistent with a lunar environment, without having to alter individual object settings.

4.7 User Interface (UI) setting

In the Scientific Workspace, the interface is designed first and foremost to help the user complete laboratory procedures without breaking immersion. Rather than starting from generic UI patterns, the planning begins with the tasks and then identifies the minimum set of visual aids needed at each step. Messages, icons and canvas are treated as operational aids: they explain what event is occurring, what the next action is, and how to execute it. I used World-space canvas anchored in the scene for instruction messages to help the user in completing the task and execute them without losing focus on the objective of the mission. The result of this choice is a quiet but informative UI, guiding the user through scientific procedures.

4.8 Project Structure and Workflow

I organized the project in Unity using a modular folder hierarchy, designed to keep assets and scripts separated. This clear structure allows easier debugging and scalability as the project grows in complexity. *Scenes* contained the different environments, *Scripts* included all behavioral code written in C#, and *Prefabs* served as reusable templates for interactive objects. The 3D assets developed in Blender were exported in *FBX* format and imported into Unity as new assets. I started populating the scene with 3D models exporting everything from Blender as FBX, with modifiers applied and rotation/scale baked. This simple step avoids a lot of confusion when the files arrive in Unity. For decorative props made of many small parts, I grouped them under a single parent (or joined them when reasonable) to keep the Hierarchy compact and easy to read. Materials needed a quick cleanup before export. The FBX pipeline only preserves basic setups, so anything more complex than a Principled BSDF (Bidirectional Scattering Distribution Function) → Material Output connection is effectively lost. I simplified materials accordingly. Finally, I double-checked face orientation: incorrect normals can lead to odd transparency on thin geometry, so I flipped normals wherever necessary. The majority of texture maps were resized and compressed into *.png* format and assigned to the prefabs in the scene. In Unity, I dragged the rover shell into the Hierarchy to create a GameObject. The design goal was straightforward: the walls should block the player and the equipment, but the interior must remain usable. Adding a single Mesh Collider to the closed shell sounds convenient, but in practice Unity treats it like a solid lump. That keeps the outside world out, yet it also pushes everywhere anything inside. To fix this there are two ways: going back to blender and split the shell in different panels with separate meshes or inside Unity use different box colliders to simulate the shell walls.

I developed each functional area of the rover as a separate scene, to allow independent production and testing.

The VR simulation is divided into three main modes which can be selected from the Main Menu scene (*Figure 27*):

❖ *Free Roaming Mode*: In this mode the user can move around the LPR, interact with the objects within the simulation and observe the environment. No specific tasks are planned in this mode, as it is designed to let the user freely explore and interact with the environment. From an interaction standpoint, the main actions regard the direct manipulation of objects: users can grab items, hold them, move them, and release them into the environment. Here, direct manipulation is intentionally emphasized to convey the sensation of handling components under reduced gravity. The hardware cannot reproduce physical weight, but the release behavior (falling and bouncing according to lunar gravity) communicates the correct dynamics intuitively. Interactable items are configured with Unity's XR Grab Interactable component. This setup exposes useful parameters to tune the "feel" of interaction:

- Physics properties (mass, drag, angular drag) govern how objects accelerate, rotate, and settle.
- Attach/track settings define how controller motion influences the object's kinematics (e.g., velocity and position tracking, smoothing).
- Colliders ensure robust contact handling, preventing interpenetration and enabling realistic collisions and rebounds against walls and other props.

It's particularly useful for becoming familiar with the controls and surroundings, as well as for public outreach and educational purposes.

❖ *Story Mode*: The story mode is the one where the user can select what type of experience he wants to do between Cockpit and Scientific Workspace (*Figure 27*). Here the player will be guided by the system to perform a sequence of tasks in a specific order to accomplish the whole story mission. Each experience has been produced with five main tasks to complete, each mutually connected. The workflow of the story mode's tasks has been created in a way that each task is important to complete the next one, keeping the player focused and making the experience more stimulating. Like in the Free mode the player can interact with the environment and with all the items which have the XR Grab Interactable script. In addition, the player is guided by world space canvas that gives instructions on what to do to progress within the scene. Besides that, the monitors on the workbench and inside the glove box will guide the player on completing the tasks.

❖ *Task Mode*: Here the player can decide and select which task he wants to practice. This enables a focused and repeatable workflow and it's more suited for specific training in one or more tasks. It is intended for targeted skill development.

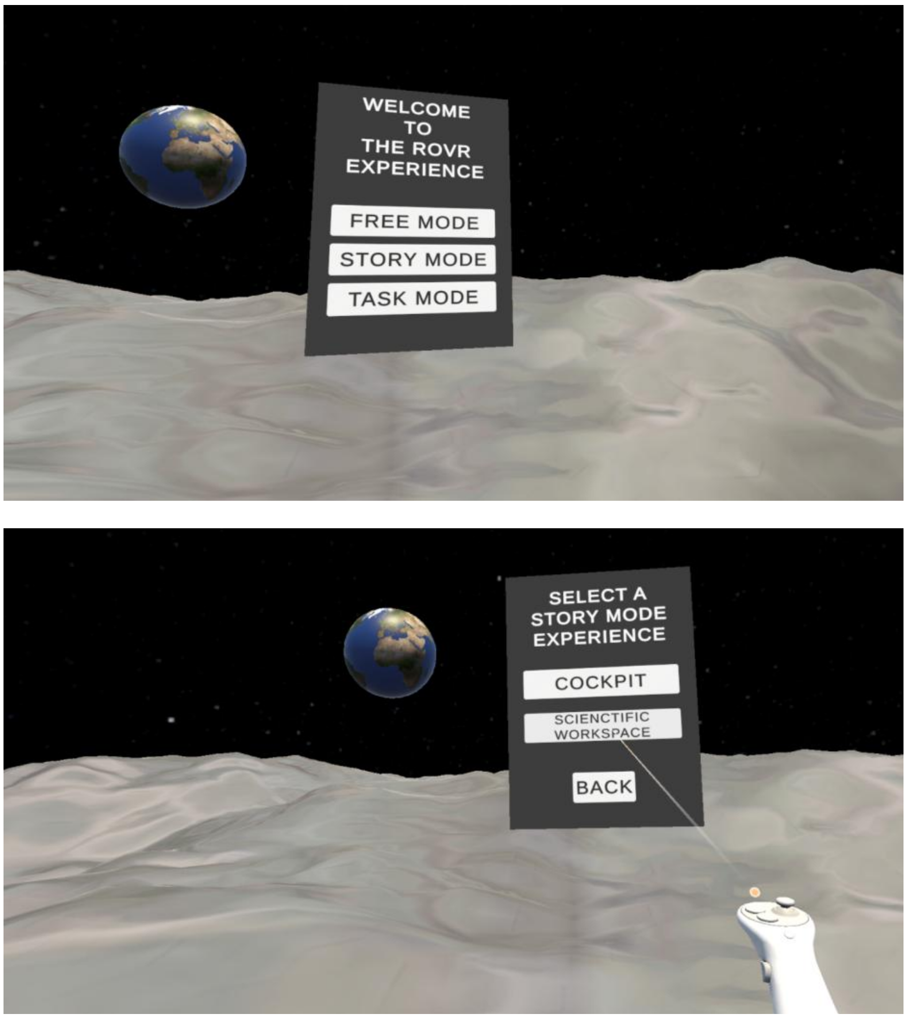


Figure 27. a) Main Menu selection; b) Story mode experience selection

4.9 Scientific Workspace task development

In this thesis I specifically worked on the scientific workspace story mode, creating a guided, task-driven experience that simulates a realistic research workflow inside the rover. I designed a coherent sequence of activities. Interaction was implemented with Unity's XR Interaction Toolkit and custom C# scripts, focusing on natural gestures (grab, press, select), immediate visual feedback and world-anchored UI panels for procedures and menu selection.

The scientific workspace experience starts from the main menu, where at the opening of the RoVR application, the player will be in an open environment with a world space selection menu (*Figure 27.* a) Main Menu selection; b) Story mode experience selection). From there if story mode and then scientific workspace are selected, the player will be transported to the scientific workspace scene (*Table 3.* RoVR app opening and main menu selection).

Table 3. RoVR app opening and main menu selection

#0	INPUT	OUTPUT
a.	User selects the Story Mode button	Main menu disappears and Story Mode menu shows up
b.	User selects the Scientific Workspace Experience button	The Scientific Workspace Experience scene loads

4.10 **Task 1: Sample and Tools selection for the experiment**

In this task the user stands in front of a lab bench with a sealed box. Here, a world space canvas welcomes and guides the player through the scene. Touching the box hides the introduction canvas and opens a selection panel (a scrollable list) with all available items and objects to bring inside the simulation. The user marks the items and presses "*Confirm*" to reach the next scene (*Table 4*).

Table 4. Task #1 description

#1	INPUT	OUTPUT
a.	User selects the sealed box	A UI list of objects and moon samples to drag into the box will appear
b.	User clicks on objects in the list	The selected objects get highlighted
c.	User presses the "Confirm Selection" button	The scene is set up with the view inside sealed box (locked with the gloves) and the object selected inside it.

4.10.1 **Implementation**

I created a world-space Selection Canvas attached to the sealed box area and managed it with a controller script, "*SelectionUIManager*". The manager receives a curated list of selectable scene objects and dynamically builds a scrollable list of buttons (one per item) using an "*ItemButtonUI*" prefab. Each button updates the corresponding object's selection state. For visual clarity, each selectable object carries a "*SelectableItem*" component. This script:

- ❖ Automatically discovers child renderers (for meshes composed of multiple sub-parts).
- ❖ Applies an HDR emission color (for a bloom-assisted glow) when the item is selected.
- ❖ Adds/removes an outline material as an extra renderer material

To make the simulation more stimulating every time the user plays it again, I created a pool of five types of moon samples with different geometry, weight and dimensions. Then, from the list of buttons, I created one called "Random Sample" that every time that is selected, it stores randomly one of the five samples that will be transferred to the sealed box scene, making every time the simulation different and more entertaining.

A simple "*SealedBoxUITrigger*" is placed on the box entrance. It exposes "*OpenSelectionUI"*, which hides the intro canvas, shows the selection canvas, and rebuilds the list. I bound this function to an interaction event (select/activate) so the canvas opens with the controller's trigger. On "Confirm", the *SelectionUIManager* script collects the IDs for all selected items and writes them to a "singleton", a pattern that ensures a class has only one instance and provides a global point of access to it ("*SelectedItemsStore"*). This bus object survives scene loads and carries just the list of string IDs.

All item IDs and their metadata (what to spawn in the box, and their reference measurements) live in a "*ScriptableObject"* database, the "*ItemDatabase"*. In the database there are all the information needed:

- ❖ *ID* (string that contains the name of the object).
- ❖ *insidePrefab* (GameObject prefab to spawn in Scene 2).
- ❖ *massGrams* (float that stores item's weight).
- ❖ *sizeMM* (Vector3; length × width × height that stores the item's dimensions in millimeters).

The spawned objects carry the *ItemProperties* component with all the information described above (ID, mass in grams, and size in mm). It allows instruments and UI to read the ground truth without coupling to the database.

Then, I created a new scene called *InsideBoxScene* where the objects selected in the scrollable list from main scientific workspace scene will be spawned. This happens thanks to a small "*InsideBoxSpawner"* reads the *SelectedItemsStore*, looks up each ID in the *ItemDatabase*, and instantiates the corresponding Prefab at the anchors within the sealed box.

4.11 **Task 2: Sample weighing and sizing and breakdown in smaller pieces**

In this task the user is already inside the sealed box, with the selected items placed inside and the instruments within reach. A monitor inside the glove box gives instructions on what to do to analyze the sample. The player drags a sample onto the balance plate and a small world-space display on the scale immediately shows the weight in grams; removing the sample resets the readout to 0.00 g. Next, the user brings the measuring instrument to the sample and the length–width–height values (in millimeters) appear on the monitor inside the sealed box. When ready, the player strikes the sample with the hammer: It breaks into distinct fragments of different weight and size. The user can then pick a fragment and repeat weighing and sizing, with the UI updating after each interaction exactly as in the previous steps (*Table 5*).

Table 5. Task #2 description

#2	INPUT	OUTPUT
a.	User drags the sample on the balance plate	The weight value appears on UI
b.	User removes the sample from the balance plate	The UI resets to '0.00 g’
c.	User measures the sample with the instruments inside the box	The dimension values appear on UI
d.	User takes the hammer and hits the sample to break it	Original object disappears; fragments appear
e.	User selects a fragment and repeats the measurements	As in the previous phases values will appear on UI after every measurement

4.11.1 **Implementation**

First of all, I had to resolve the same problem that I had with the colliders of the shell. In fact, if I used a mesh collider for the closed glove box, Unity treats it like a solid lump so the item inside will bug and get scattered outside the box. To solve this, I created a “BoxWalls” container with four thin wall colliders and a ceiling collider set to be static, non-moving colliders. In this way the items, which have colliders and rigidbodies, did not interfere with the walls of the glove box. With this solution their layer collision matrix allows normal interaction, but blocks exit through the box walls.

After the selection phase, the *InsideBoxScene* is loaded and populated by the *InsideBoxSpawner*, which reads the user’s choices from *SelectedItemsStore* and spawns the corresponding prefabs at predefined anchors inside the sealed box. A physical scale plate prefab with thin box collider slightly above the visual plate, detects objects resting on it and updates a small world-space canvas positioned over its black display. The plate logic is implemented by a “*WeighingPlate”* script that:

- ❖ Tracks entering/exiting rigidbodies and maintains a set of valid contacts.
- ❖ Compute the displayed weight.
- ❖ Supports continuous update with optional temporal smoothing for a realistic readout.
- ❖ Formats the value as “0.00 g” and resets to 0.00 when the plate is empty.

A simple measuring tool contains a narrow detection collider and a dedicated world-space canvas for the readout. Through scripting, when a sample overlaps the tool, it reads the dimensions in millimeters (L x W x H) from the target’s *ItemProperties* and prints them, updating the values in real time as the user moves the tool across different objects.

To generate the input where the sample gets broken into pieces, I decided to use a strategy which consist in creating a “pre-fractured sample” in Blender using the Cell Fracture add-on. After applying transforms and ensuring clean topology, I ran the *Cell Fracture add on* with Source Limit = 5 (to obtain five fragments for each sample), Margin = 0.002 (to avoid initial compenetrating), and a small noise value (to introduce natural irregularity). The resulting shards were grouped, their origins reset to geometry and exported as FBX with "Apply Transform" enabled. In Unity, I assembled all the fragments under their respective fractured sample prefab, each with a Convex Mesh Collider, a Rigidbody (Continuous Dynamic), and XR Grab Interactable script (Velocity Tracking). Then, I assigned a "*BreakableSample"* script to the samples and the tag "hammer" to the equivalent prefab and implemented a speed threshold to capture the force of impact (via impact velocity). When a collision satisfies both conditions (correct tag + minimum speed), the sample object is immediately replaced by a corresponding set of fragments taken from a "*FragmentPool"*, which separate and scatter physically with an explosion force centered at the hit point for natural separation. The fractured prefab is instantiated at the same world pose as the original sample, and each fragment has assigned its own *ItemProperties*, enabling the instruments (digital scale and measuring tool) to report piece-specific mass and dimension values.

To guide the workflow inside the glove box I added a monitor UI managed by a script called "*BoxMonitorUI*". The monitor is a world-space canvas with multiple pages that show instructions and hints on what to do alongside the dimension's readout page from the measuring tool.

4.12 **Task 3: Exit from the sealed box and microscope detailed analysis**

In this task the user leaves the glove box carrying over one fragment out of it. Then, when the fragment gets dragged into the microscope a small confirmation panel appears; accepting the prompt transfers the user to a dedicated "*Microscope Scene*" where the view simulates a fixed top-down microscope analysis. Here the fragment is presented as a high-resolution slide that fills the field; the left stick steps through magnifications (5× → 20× → 50× → 400×) and with an on-screen overlay shows the current zoom level. On the other hand, the right stick pans within the image and let the user navigate and analyze the image (*Table 6*). The petrographic polarizing microscope focuses on transmitted-light PPL/XPL modes, using high-resolution textures that approximate the visual appearance of lunar basalt and regolith thin sections.

Table 6. Task #3 description

#3	INPUT	OUTPUT
a.	The user chooses and grabs the fragment that he wants to examinate outside the sealed box	The UI monitor inside the box shows the instructions to the user to carry over the fragment
b.	User selects the button to exit with the fragment in his hands	The view changes and now the user is back in the free view of the workbench, and the examined object is outside of the sealed box

c.	User takes the object he wants to examine and drags it into the microscope	When the object is dragged a confirmation of the action will appear on UI: "Do you want to examine this object?"
d.	The user confirms the selection in the microscope	The view changes and now the user is within Microscope Interface, a list of controls appears on UI
e.	User moves left stick	The microscope camera zooms in/out to various resolutions
f.	User moves right stick	The camera pans around the sample analyzed and shows the details

4.12.1 **Implementation**

I introduced a "carry-over" flow for fragments: on clicking "Confirm" inside the box, the *BoxMonitorUI* detects any grabbed object via the interactor's selection list, freezes its physics, and hands it to a CarryOver. The destination scene uses "*BoxExitDrop*" script to retrieve that object, move it into the active scene, restore colliders, rigidbody settings, and re-enable *XRGrabInteractable*, spawning it at a defined drop point outside the glove box.

The microscope hand-off is gated by a dedicated "*MicroscopeDock"* trigger that filters only valid fragments (by the presence of the ItemProperties). On contact it opens a small world-space confirm panel; confirming the selection records the return scene name and canvas toggling instructions, then loads the microscope scene.

Inside the microscope scene, a *MicroscopeViewer* script drives a 2D, head-fixed UI: a world-space canvas with a masked viewport contains a single Image ("slide") that swaps sprite resolution tiers (5×, 20×, 50×, 400×) on left-stick movement (↑ to zoom in, ↓ to zoom out). The slide is intentionally oversized relative to the viewport so right-stick input pans it (↕ up and down, ↔ left and right) and give a real sensation of using a microscope. Clamping prevents panning beyond the image edges. A small script manages an HUD which shows instructions for controls and on each zoom step, the current resolution. The same component binds an exit action (with the XR trigger button) to leave the microscope and the scene. When the scientific workspace scene is reloaded, the script that permits the change of scenario automatically deactivates all previous instructions canvas and enables the new one for task four.

4.13 **Task 4: Data collection, multiple choice quiz and answer feedback**

Here, pressing the exit button returns the user to the Scientific Workspace, where the task's instruction canvas is shown. The user completes the workflow by documenting and verifying the analysis results. After returning to the Scientific Workspace with one fragment, the player grabs a pen Gameobject and taps the notebook lying on the bench. A small world-space confirmation panel ("Are you ready to start the Quiz?") appears with *Yes / Go back*. Accepting opens a three-page multiple-choice quiz laid out as notebook pages:

1. A question about the fragment's mass (in grams).

2. A question about the fragment’s dimensions (L×W×H in millimeters).
3. A question about the microscope view, where the player must pick the correct image among three thumbnails.

Only one answer per page can be selected. When the user presses *Check answers*, the quiz closes and a result message is shown on the monitor above the bench; if all answers are correct a short completion message appears, if not the message will suggest the user to try again (*Table 7*).

Table 7. Task #4 description

#4	INPUT	OUTPUT
a.	User exits microscope view	The view goes back to the workbench
b.	User selects the notebook	The page on the notebook appears on UI with a multiple choice for each dimension displayed
c.	User selects the data of the analyzed fragment for each quiz page and then clicks “check answer” to receive feedback	• If some questions are incorrect: a message will appear on the workbench monitor, suggesting trying again • If all questions are correct: a message of congratulations will appear on the workbench monitor completing the task

4.13.1 **Implementation**

I implemented the interaction and UI in two main scripts:

❖ *“NotebookQuizTrigger”* (on the notebook object): the notebook is an XR-interactable surface that listens for a “pen tap” (via the XRI UI pipeline). On tap it shows the Confirm Canvas; selecting “Yes” hides the confirm and enables the Quiz Canvas, meanwhile selecting “Go back” button dismisses it (this script only toggles canvases).

❖ *“NotebookQuizUI”* (on the Quiz Canvas root) builds and runs the quiz at runtime. At Awake it auto-discovers its child pages (“Page 1”, “Page 2”, “Page 3”), their Toggle groups, and footer buttons (Back, Next, Check Answer). Each page uses a ToggleGroup for single question selection.

For mass and size, the script queries the carried fragment’s *ItemProperties* (ID, mass, size) to produce the correct option, then procedural “plausible wrong options” are synthesized by perturbing the ground-truth within bounded percentages.

For the microscope page, the correct image is picked at random from the four sprites used in the microscope viewer scene, while the two wrong thumbnails come from a dedicated wrong sprites pool. The script also shows a brief on-screen result on "*Check answers*" and, when all three are correct, hides the quiz itself to reveal the final result on the workstation monitor.

The carry-over script guarantees that the same fragment used under the microscope is also the one queried by the quiz. The "*MicroscopeDock"* records the chosen fragment (via a ItemProperties check) before loading the microscope scene.

The confirmation and quiz canvases are world-space UI with a standard event system. The pen uses the XRI UI interaction so tapping feels natural. All canvases are initially inactive, they are shown strictly by the trigger flow, so earlier instruction canvases from prior tasks never overlap the quiz phase.

4.14 **Task 5: Emergency scenario**

This emergency scenario has not yet been implemented in the current simulation *APK* (*Android Package Kit*). It is intentionally scoped as a bonus extension to enrich the overall experience of the core workbench workflow (Tasks 1–4) and is planned to be added soon to deepen immersion and evaluate user readiness.

This fifth task extends the experience beyond the Scientific Workspace. A sudden rover system malfunction is announced on the wall monitor with an alarm sound, flashing red lights, and a short countdown overlay. The player exits the workspace, navigates inside the LPR, reaches the EVA and airlock area, and equips a space suit part-by-part. Once suited, the player returns to the monitor to identify the fault. After acknowledging the report, the alarm ends and the environment returns to normal (*Table 8*). A similar task is also planned for the Cockpit "story mode," where the alert originates from the rover MFD.

Table 8. Task #5 description

#5	INPUT	OUTPUT
a.		In the monitor above the workbench an alert of malfunction of rover system will appear alongside with an audio warning (also visible from cockpit section). An overlay UI countdown starts, and red lights begin to flash
b.	User leaves the scientific workbench area	User is in free roaming mode, so he is free to move around the rover
c.	User reaches the space suits and click on it to wear each part of it	If clicked in the right order, the different parts of the equipment are worn by the user; when all the elements are placed correctly, a success message appear
d.	User goes back to the monitor and checks the malfunction in the workbench monitor	In the monitor, after the space suit will be worn, the type of malfunction will be displayed
e.	User clicks on the monitor to end the emergency	The audio and the warning will disappear, and the emergency will end

Chapter 5 – Results

5.1 Main Menu

The Main Menu scene is the first scene that opens when the simulation starts. Here, the canvas UI lets the player choose between the two story modes (Scientific and Cockpit), as well as the Free Mode and the Single Task Mode. I created an Earth object using a custom high-resolution texture and combined it with a starfield rendered on a black skybox to give a strong sense of being in space. The lunar ground comes from a Blender model, imported with a bump map to keep the surface readable and detailed (*Figure 27*).

As a result, the main menu gives a strong visual feedback of the environment thanks to the setting and the VR immersion.

5.2 Free Mode

Free Roaming Mode let users interact with the environment without a predetermined series of task and the need of "learn by doing". In this mode users can freely grasp, move, rotate, and release objects; observe the object's behavior under reduced moon gravity; and try interacting with constraints such as walls and table surfaces. As a result, the user obtains three concrete benefits:

1. *Familiarization*: users calibrate their movements and controls and grasp the internal sense of mass, inertia and friction for later tasks, being ready to perform when accuracy is needed.

2. *Robustness checking*: unconstrained play will naturally exercise edge cases (rapid grabs, drops, collisions), helping reveal issues in object colliders, grab settings, or rigid-body parameters before task modes.

3. *Engagement and confidence*: starting the training part right ahead without gaining confidence with the environment lowers cognitive load and encourages exploration, so users enter task phases already comfortable with the controls and with the "feel" of lunar gravity.

The free mode also gives the possibility to be used as an engaging experience for students and scholars as a tool for public outreach events, letting people who are not astronauts or experts in this field, try the experience first-hand by interacting with the various objects in the scene, without any prior training or specialized knowledge.

5.3 Workbench Task 1

The selection panel reliably lists the available tools and samples, including the "Random Sample" option (*Figure 28, Figure 29*). This random option keeps replay fresh; every time the user can analyze a different rock on a new run. Selecting entries highlights their counterparts in the workspace; pressing Confirm consistently spawns the chosen set inside the sealed box scene. The script store transfers only string IDs, leading to deterministic spawns via the item database. The task ends with the confirmation of the item in the selection canvas but when the user is inside the box, he will have the possibility to go back to the first scene to bring other objects to the glove box and start again the experiment.

To summarize, selection is smooth and easy to organize, also the visual feedback helps the player to complete the task.

5.3.1 Task Workflow bullet points

- ❖ Touching the sealed box hides the introduction panel and opens a selection canvas with a scrollable list of available items.
- ❖ The Random Sample button selects one of five lunar rock prefabs; on *Confirm*, the scene switches to *InsideBoxScene* and spawns the chosen objects at predefined anchors.
- ❖ Visual selection feedback (glow and outline) on the bench objects mirrors the state in the UI list; state is preserved until *Confirm*.

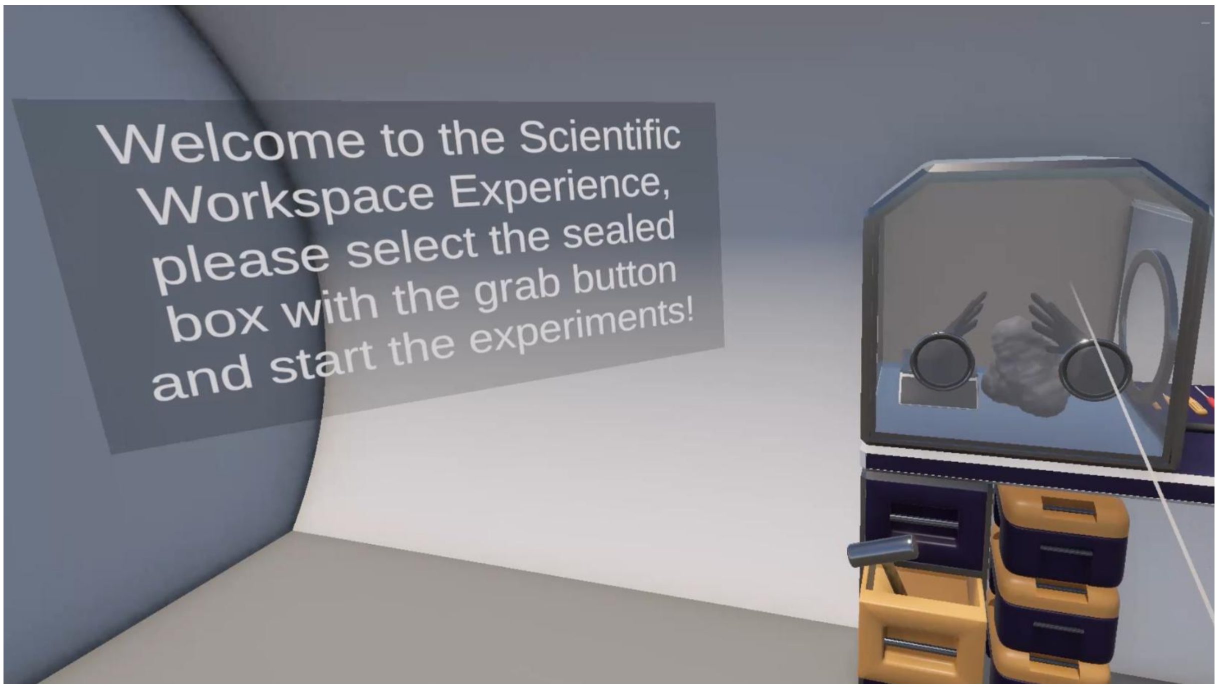

Welcome to the Scientific Workspace Experience, please select the sealed box with the grab button and start the experiments!

Figure 28. Introduction Canvas and story mode start

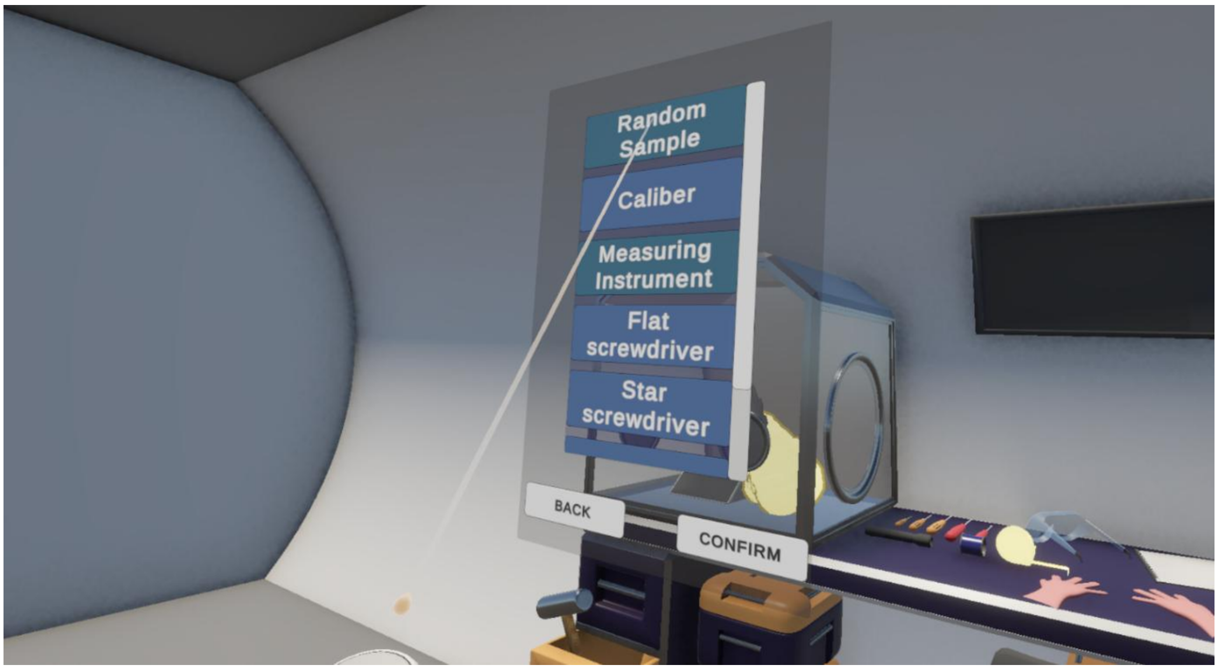


Figure 29. Selection Canvas and object outline/glow on selection

5.4 Workbench Task 2

Placing a sample or fragment on the balance plate triggers an immediate UI update over the plate's black display (*Figure 30*). With *ItemProperties* assigned to the object, mass is shown on the scale. The reading smoothly settles within milliseconds, aided by temporal smoothing to avoid flickering during minor vibrations. Removing the object clears the contact. Bringing the measuring instrument in contact with a sample displays length–width–height in millimeters (*Figure 31*). When authored dimensions are absent, bounds-based estimation yields stable, repeatable values. The readout appears at the instrument's display location and updates continuously as the user moves across different targets. For what regards the breaking action, on the first valid hammer hit (hammer prefab wit tag "*Hammer*") at the correct speed threshold (0.8 hit speed), the intact sample is replaced by a pre-fractured set of five pieces at the same world pose. Each piece is assigned its own *ItemProperties* (mass and size), enabling immediate, piece-specific readings on both instruments. A small impulse separates the shards while respecting the box walls (*Figure 32*). When a rock with an assigned total mass is broken, the per-fragment masses summed reproduce the original initial value.

In conclusion, the user can select items, see clear highlighting, and then weigh, measure, and break samples inside the sealed environment with reliable and readable feedback. The instrument readouts are immediate and consistent, and the fracture event is realistic.

5.4.1 Task Workflow bullet points

- ❖ When the scene loads, the *InsideBoxSpawner* places the selected sample and tools.

- The scale displays weight in grams with 10^{-2} precision (0.01 [g]); when objects leave the plate, the readout resets to 0.00 [g].
- The measuring instrument through its probe shows dimensions L×W×H (mm) for the touched object; values are correct for items with *ItemProperties*.
- The hammer breaks a sample into pre-fractured pieces; each fragment carries its own *ItemProperties*, so weighing and sizing post-break are consistent with their real weight and dimensions.
- The monitor inside the glove box functions as a step-by-step guide (multi-page UI), including the controlled navigation between pages (back, next and exit buttons) (*Figure 33*).

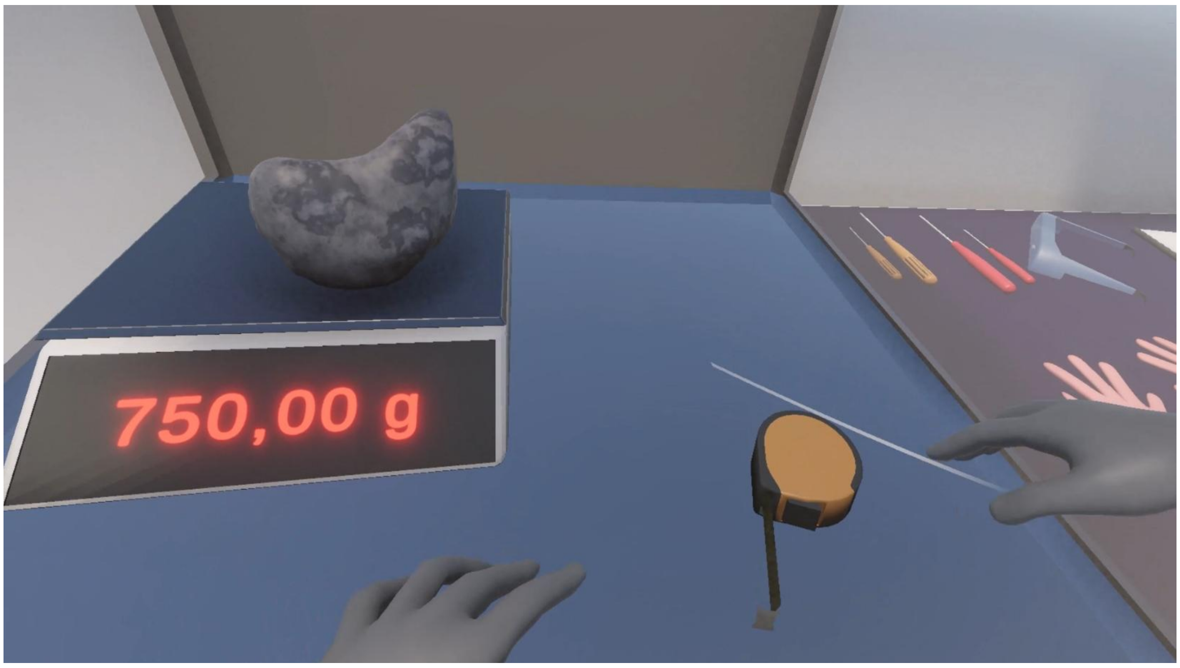


Figure 30. Scale and weighing phase

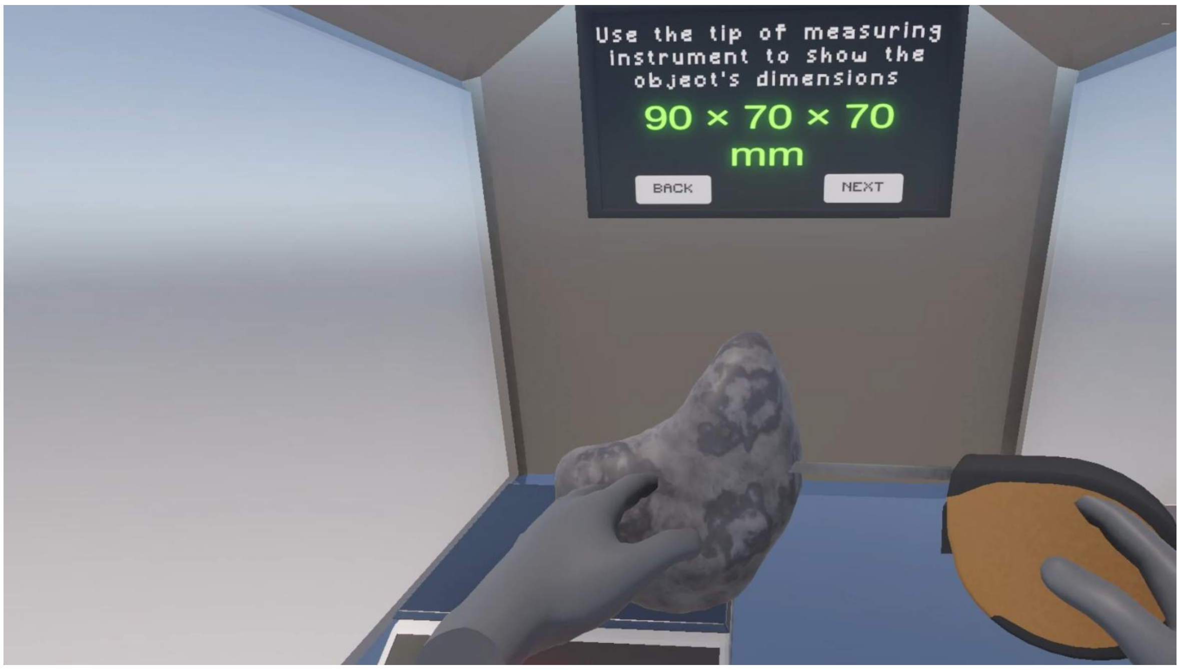

Use the tip of measuring instrument to show the object's dimensions
90 × 70 × 70 mm
BACK
NEXT

Figure 31. Measuring process

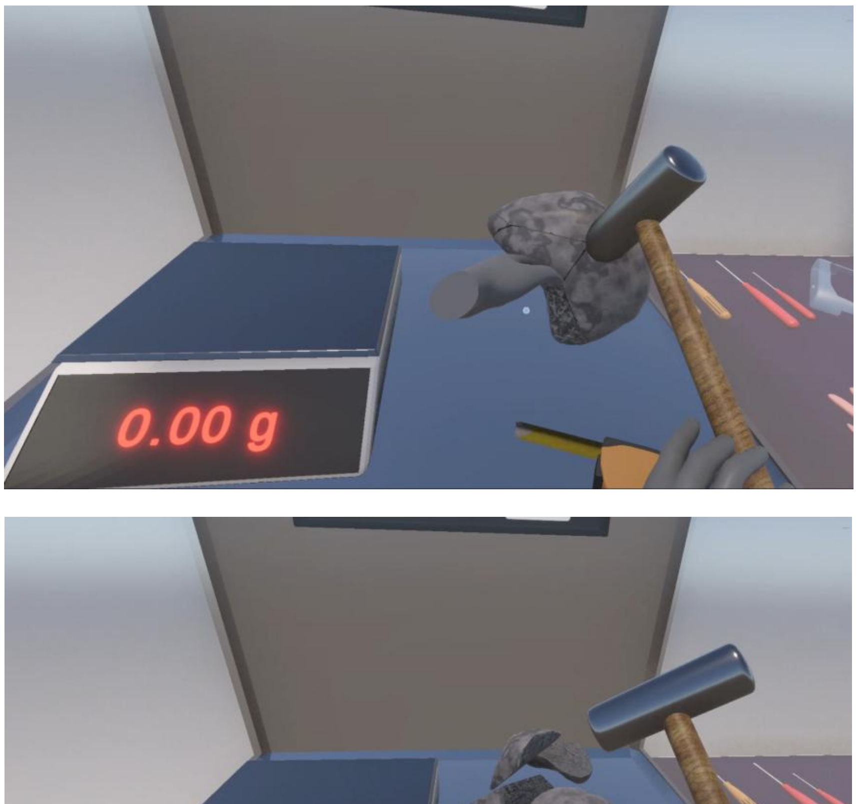


Figure 32. Sample Breaking

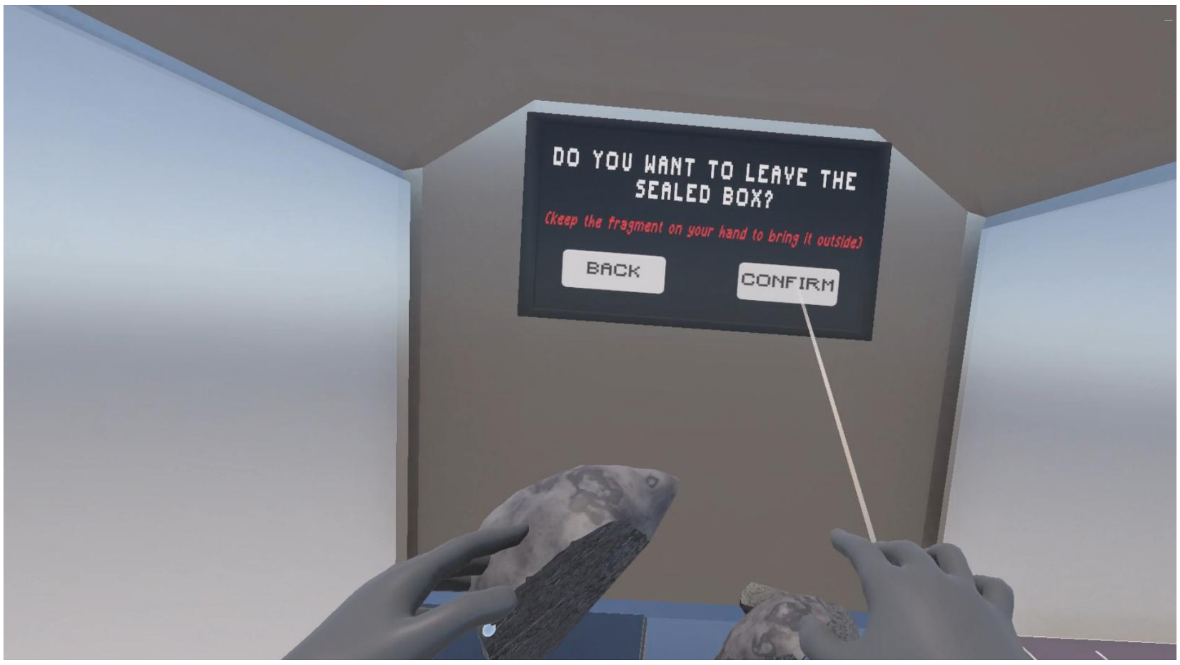

DO YOU WANT TO LEAVE THE SEALED BOX?
(Keep the fragment on your hand to bring it outside)
BACK
CONFIRM

Figure 33. Monitor UI instructions

5.5 Workbench Task 3

When the player drops the analyzed fragment into the microscope socket, a confirmation prompt appears (*Figure 34*). Choosing to analyze and confirming the selection switches the current scene to a 2D microscope viewer one (*Figure 35*). The image fills the field and behaves like a slide: the left stick steps through magnification levels with a brief fade that mimics objective changes, and a small overlay shows the magnification. The right stick pans the images within tight bounds, so the user never steps outside the texture. Pressing the exit input returns to the Scientific Workspace and cleanly swaps canvases (microscope instructions off, next task instructions on). The fragment is kept intact by the carry-over flow, ensuring continuity between analysis and subsequent assessment.

5.5.1 Task Workflow bullet points

- ❖ From the inside of the box, the analyzed fragment spawns on the desk of the workbench thanks to the *CarryOver* script and an anchor positioned in a pre-determined space.
- ❖ Placing the fragment in the microscope socket triggers a confirmation panel; accepting loads the Microscope Scene.
- ❖ The microscope scene locks head motion relative to the image and presents a 2D top-down viewport.

- ❖ Left stick steps magnification 5× → 20× → 50× → 400×, with a textual overlay (“20×”, etc.) for 2 seconds.
- ❖ Right stick pans within the oversized slide; panning is clamped at image bounds to avoid blank areas.
- ❖ Pressing the exit button returns to the Scientific Workspace; on return, previous instruction canvases are automatically hidden, and the Task 4 instruction canvas becomes active.

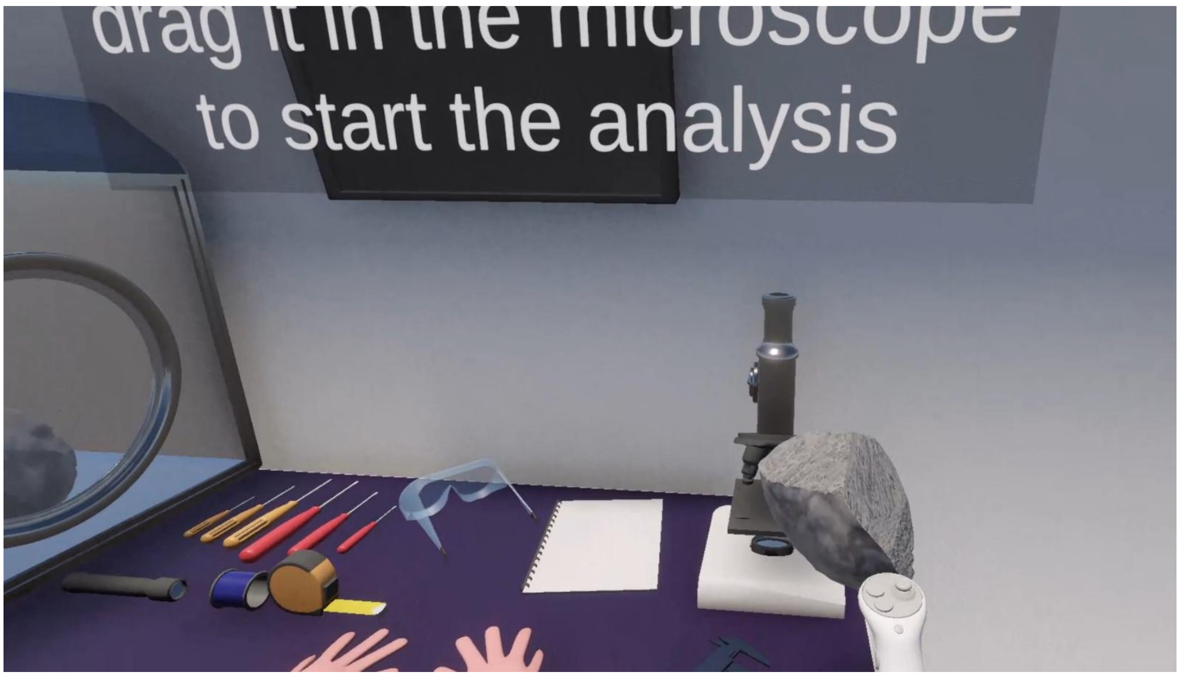


Figure 34. Microscope socket

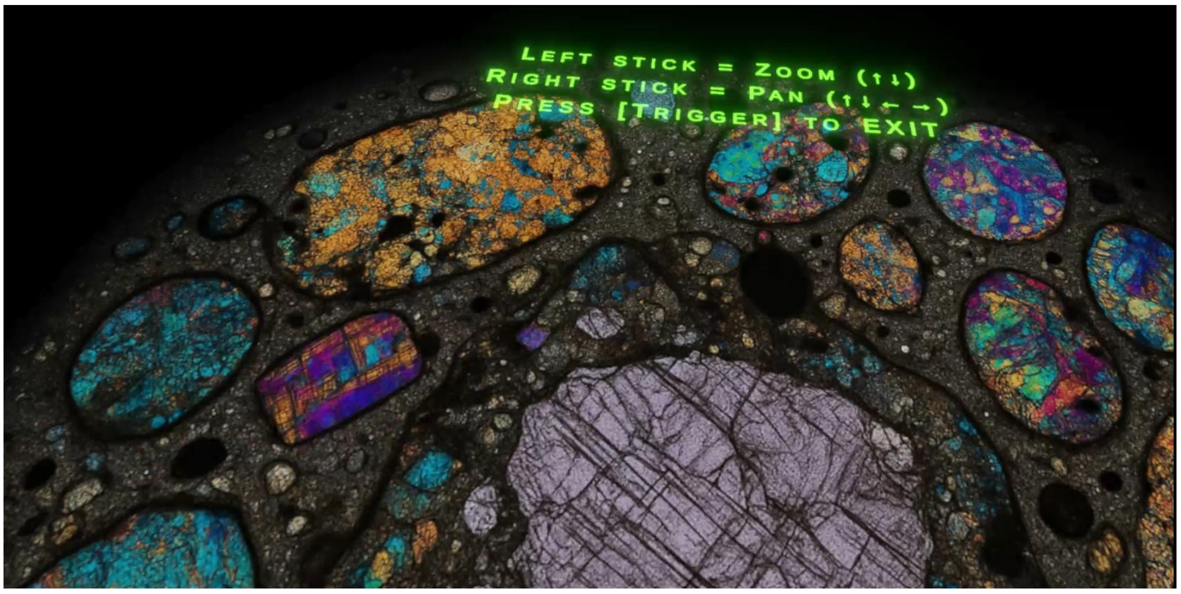


Figure 35. Microscope Scene

5.5.2 Microscope Images

After weighing, sizing, and controlled fracture inside the sealed box, a representative fragment was transferred to the microscope station and “examined” using simulated thin-section imagery rendered under transmitted polarized light (PPL/XPL). The dataset consists of four high-resolution images at increasing magnifications. Although no physical optical path was used, the renderings reproduce key optical behaviors (extinction, birefringence contrast, relief, opaques) and are presented at laboratory-grade resolution to support realistic training.

All figures are generated through AI and intentionally designed to approximate *typical* lunar textures for instructional and training use (*Figure 36*). I produced the images to mimic a petrographic microscope analysis with rotatable cross polarizers and a 30 [µm] thin section.

5x Magnification

❖ Whole fragment view: overall shape and main light (crystal rich) and dark (glass) zones.

❖ Round dark holes → Pores/vesicles.

❖ Bright blocky pieces → feldspar-rich clasts; smoother dark zones → glass/impact melt.

20x Magnification

❖ Long, light laths (thin crystal) → plagioclase (feldspar).

- ❖ Slightly darker crystals with straight extinction→ pyroxene.
- ❖ Jet-black grains→ opaques (because they are opaque in transmitted light).
- ❖ Clear or brownish "filler" between crystals → glass/fine groundmass.
- ❖ Fine shock cracks can cut through feldspar and terminate in glassy areas.

50x Magnification

- ❖ Sharp crystal–glass boundaries.
- ❖ Plagioclase twins appear as thin, parallel stripes.
- ❖ Pyroxene shows faint zoning (slight color/brightness rings).
- ❖ Curved, needle-like microlites in the glass bend around oxides, showing flow.
- ❖ Narrow, dark shock-melt veins cross multiple minerals.

400x Magnification

- ❖ Very fine stripes in feldspar twins; some zones where twins blur out (shock damage).
- ❖ Pyroxene lamellae (thin internal lines) become visible.
- ❖ Glassy veins have ultra-fine crystals
- ❖ It's possible to measure tiny features (lamella spacing, vein width) at this scale.

The microscope task in this thesis demonstrates how a virtual lab can deliver credible petrographic analysis practice while remaining cost-effective, safe, and reproducible. For users unfamiliar with a sample's internal structure, the microscope view also serves as a clear visual guide that deepens the simulation experience. Beyond training, these visuals are valuable for public outreach, helping broader audiences grasp what rock textures and minerals look like under the microscope.

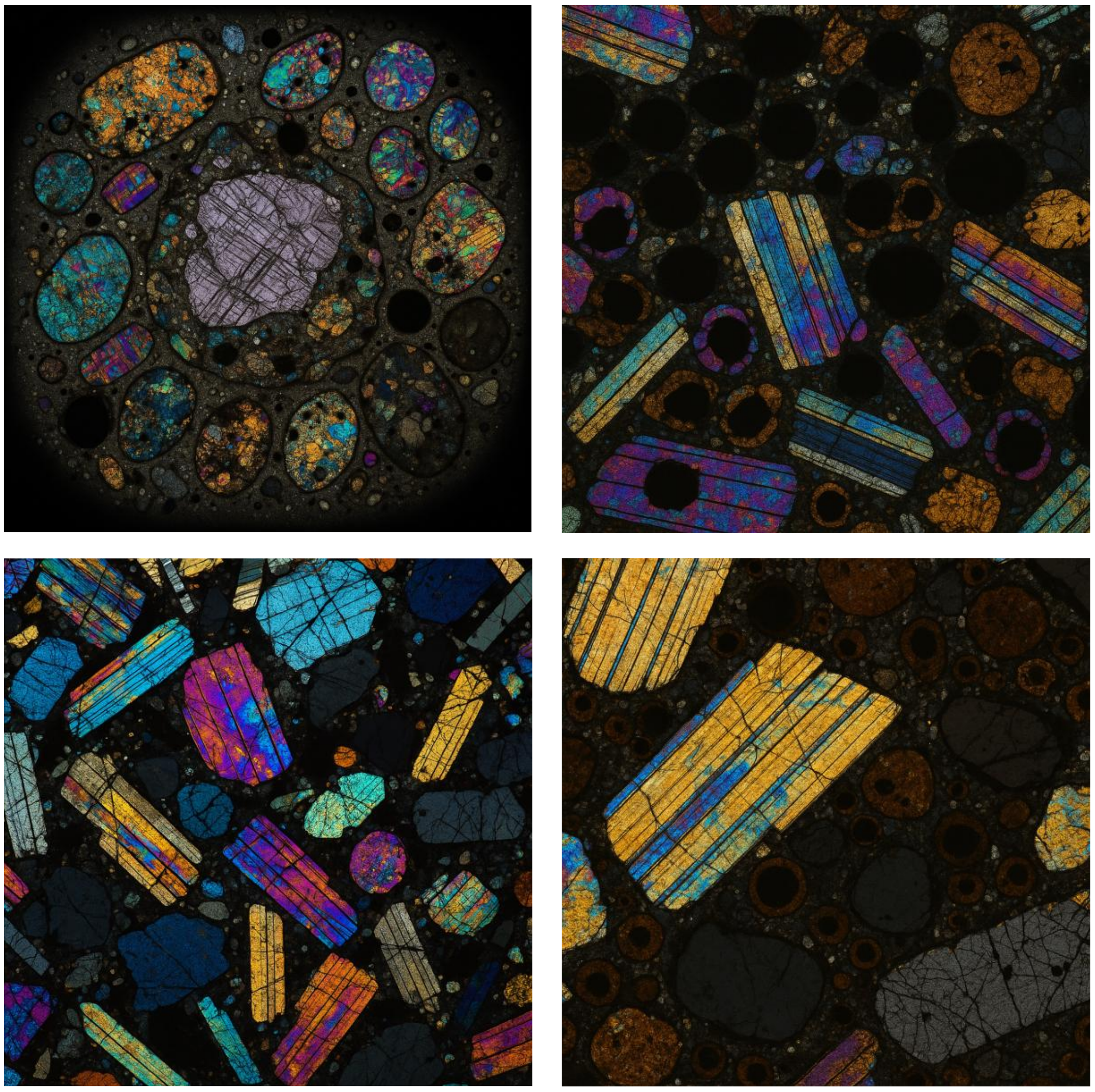


Figure 36. Microscope fragment's image. In order: a) 5x Resolution; b) 20x Resolution; c) 50x Resolution; d) 400x Resolution.

5.6 **Workbench Task 4**

The quiz starts with a small confirmation that can be triggered thanks to the collision between the pen and the notebook GameObjects (*Figure 37*), so users don't open it by accident while handling tools. Once accepted, a notebook-styled canvas appears with three pages (one question per page) and radio toggles that guarantee only one choice at a time. For Q1 and Q2, the correct values are pulled dynamically from the exact fragment the player carried out of the glove box; wrong answers are generated by the code within realistic values, so they look credible but remain distinguishable. For Q3, the system picks one of the four microscope sprites the user worked with as the correct answer and mixes it with two wrong images (also AI generated) to ensure all three thumbnails are different. The two incorrect images were created in such a way that they could appear to be plausible representations of the fragment’s internal structure, even though they actually show different materials (granite, schist, sandstone, oolite). These images get selected randomly from a specific pool that always put one of the four correct images as the right answer.

Hitting Check Answers validates the three selections; on success the quiz and any earlier instruction UIs are dismissed, and the workbench display shows a short congratulation message. This closes the sequence of sample analysis’s task from measurement to analysis to verification.

5.6.1 **Task Workflow bullet points**

- Touching the notebook with the pen opens a confirmation canvas (“Are you ready to start the Quiz?”) with Yes / Go Back.

- Selecting Yes reveals a three-page quiz:
  - Q1 – Mass: one correct value from the fragment’s mass database (*ItemProprieties*), plus two plausible distractors.

  - Q2 – Dimensions: one correct L×W×H (mm) from *ItemProperties.size*, plus two perturbed options within realistic tolerances.

  - Q3 – Microscope image: the correct option is one of the four microscope sprites at random; the other two options are different, wrong images to avoid duplicates.

- Each page enforces single selection; Back/Next navigates, and Check Answers appear and is selectable only on the last page.

- On submission, if all answers are correct the quiz closes itself and prior canvases; the bench monitor shows a congratulatory message. Otherwise, feedback prompts the user to revise choices (*Figure 38*).

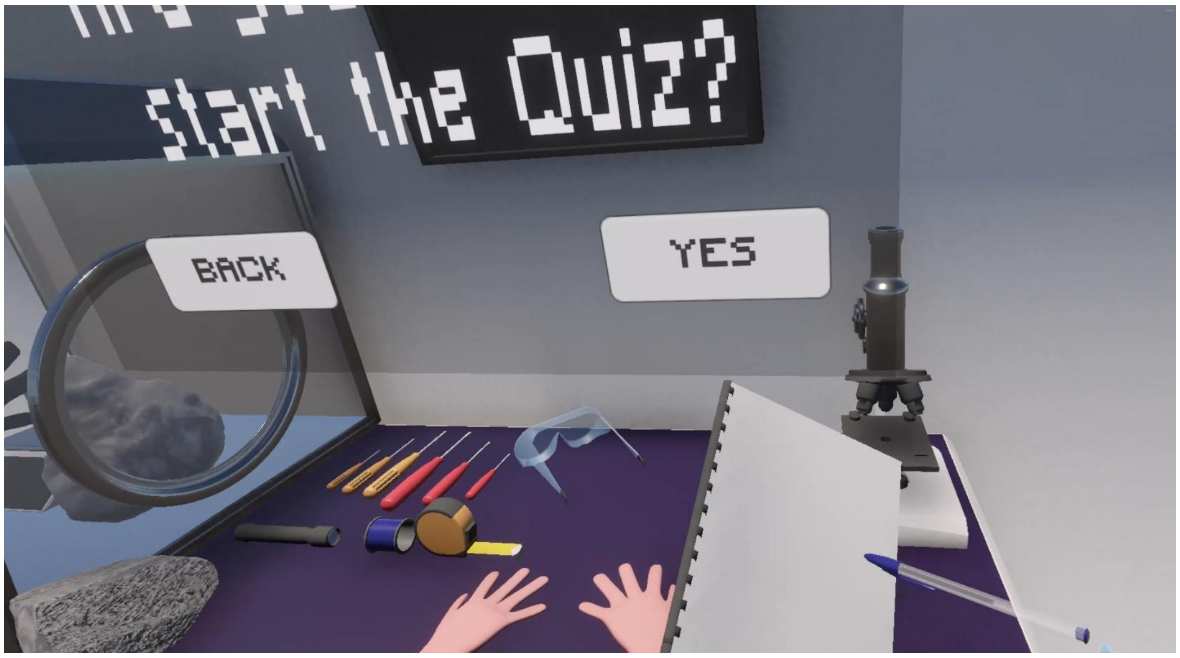
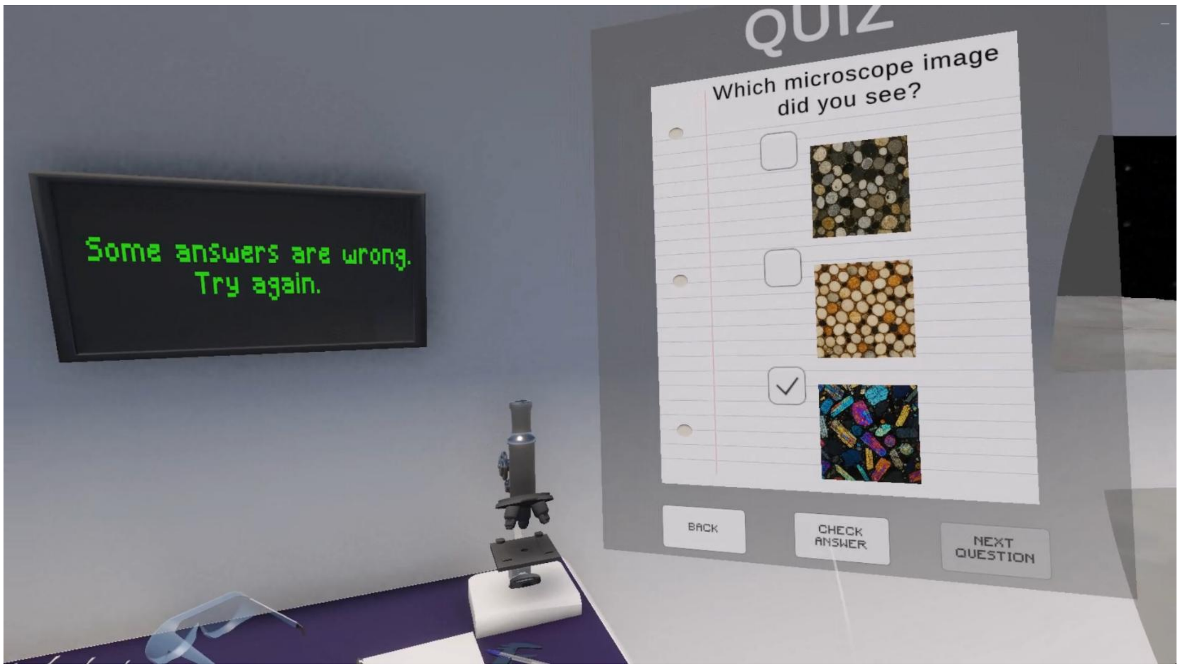


Figure 37. Pen and Notebook collider interaction



Some answers are wrong. Try again.

Which microscope image did you see?

Figure 38. Quiz

Chapter 6 – Conclusions and future developments

6.1 Conclusions

The main goal of this thesis was to design and implement *RoVR*, a Virtual Reality simulation of the internal environment of a Lunar Pressurized Rover, with a specific focus on the Scientific Workspace and a set of training tasks related to lunar sample analysis. The idea of the thesis is born from a clear gap: despite the growing interest in future lunar missions and in pressurized rovers, there are still very few training tools that allow astronauts or operators to become familiar with the interior layout and procedures of such vehicles in an immersive way.

The development followed a step-by-step process. I started by reviewing the current state of the art in XR technologies, Digital Twin concepts and existing VR applications in astronaut training which helped me to identify both opportunities and gaps. Research from an academic literature review showed that VR is already being used for EVA training, path planning or habitat design, but no dedicated platform was found for intra-vehicular activities inside a pressurized rover, especially for sample collection and analysis.

To address these gaps, I developed a virtual simulation of the rover interior. The LPR interior was designed and modelled in Blender, using hierarchical diagrams and the concepts of Level of Detail (LOD) and Level of Interaction (LOI) to organize the 3D assets. The rover includes five main functional areas (Cockpit, Scientific Workspace, Airlock and EVA, Crew and Utility, and Emergency), with the Scientific Workspace and Cockpit as the most relevant for operational tasks. Particular attention was given to ergonomics and to the arrangement of instruments, monitors and tools, so that the virtual environment would resemble credible future designs inspired by NASA's Space Exploration Vehicle and ISS-like habitats. For implementation, Unity was chosen as the game engine because of its component-based architecture, the Universal Render Pipeline and the availability of the XR Interaction Toolkit, which simplifies interaction design for VR. The Scientific Workspace story mode development was the core of this thesis. It consists of a sequence of four main tasks: sample selection; sample weighing, sizing and breaking; microscope analysis; and quiz and data verification. These tasks are inserted into a bigger framework that includes a Main Menu scene where it is possible to select three modes (Free Roaming Mode, Story Mode and Task Mode).

The development of RoVR manages to fulfill the initial goals of the project, demonstrating that VR can be used to recreate the interior of an LPR with enough realism to support procedure training and scenario exploration. In addition, it shows a coherent workflow for sample analysis, from selection to measurement, microscopic analysis and final data verification, remaining modular and extendable both in terms of content (new tasks, new models and tools) and in terms of environments (other planets).

On top of that, this kind of simulation offers training and outreach potential, since the same environment can be used both by future astronauts and by students or the general public to better understand lunar operations.

6.2 Future Development

The work done in this thesis can be seen as a starting point for the future space training market. Several future developments are already ready to be implemented in the current game architecture.

The first upcoming and reasonable development is the full implementation of the emergency scenario (task 5), which is already described at design level. Here the user has to leave the Scientific Workspace, move through the rover interior, reach the airlock area, equip a space suit, and then return to resolve the malfunction. Integrating this task will connect the Scientific Workspace to other sections of the rover and test how users behave under time pressure and simulated stress.

This thesis is created with the focus on the internal part of the rover. In future versions, the simulation could be extended to include EVA activities outside on the lunar surface. Users could exit the rover through the airlock, walk outside with lunar gravity, use tools to collect samples, deploy instruments and then re-enter following correct depressurization and pressurization procedures. Some models of EVA tools already have been created (tongs, rakes, scoops, drill) and could be used in this external scenario.

Interaction in *RoVR* is mostly visual and based on motion controllers. Future work could introduce haptic feedback, for example vibration when grabbing tools, hitting the sample with the hammer, or pressing buttons on panels or audio feedback and warning during the experiments and audio instruction from mission control that would make the environment more realistic.

An important step for validating the simulator as a training tool will be to conduct user evaluation testing. The prototype could be tested with groups of students in aerospace engineering, engineering professionals or astronauts to find out how quickly they learn the tasks and how comfortable they are following the instructions and executing the various operations. Experts' feedback would be also useful to assess the ergonomic and the scenario design.

6.3 Final Remarks

To summarize, the most important outcome is not only the VR simulation itself, but the fact that it shows how a pressurized lunar rover can be explored from the inside before it even exists. By building and testing RoVR step by step, I was able to see both the strengths and the limits of immersive simulations for astronaut training: they can't replace physical tests, but they can already help people understand procedures in an immersive and interactive way.

With further development and input from agencies and professionals, tools of this kind could eventually become a standard part of astronaut preparation.

Bibliography

[1] Milgram P., Takemura H., Utsumi A. and Kishino F., 1995, Augmented reality: a class of displays on the reality–virtuality continuum, Proc. SPIE 2351, Telemanipulator and Telepresence Technologies, 21 December 1995, pp. 282–292.

[2] Kraal J.C., 1996, An application of virtual reality to engineering design: Synthesis of spherical mechanisms, Master of Science Thesis, Iowa State University, Ames, USA.

[3] Fisher S.S., Wenzel E.M., Coler C. and McGreevy M.W., 1998, Virtual Interface Environment Workstation, Proceedings of the Human Factors Society Annual Meeting, Aerospace Human Factors Research Division, February 1998.

[4] Cruz-Neira C., Sandin D.J., DeFanti T.A., Kenyon R.V. and Hart J.C., 1992, The CAVE: Audio-Visual Experience Automatic Virtual Environment, Communications of the ACM, 35 (6), 64–72.

[5] Joskowicz J., 2023, A Historical and Current Review of Extended Reality Technologies and Applications, Universidad de la República, Montevideo, Uruguay.

[6] Meta, 2018, Introducing Oculus Quest — A New All-in-One VR System Coming Spring 2019, September 26, 2018, https://about.fb.com/news/2018/09/introducing-oculus-quest/

[7] Berg L.P. and Vance J.M., 2017, Industry use of virtual reality in product design and manufacturing: a survey, Virtual Reality, 21 (1), 1–17.

[8] Caudell T. and Mizell D., 1992, Augmented Reality: An Application of Heads-Up Display Technology to Manual Manufacturing Processes, Proceedings of the Twenty-Fifth Hawaii International Conference on System Sciences, Vol. 2, pp. 659–669, Kauai, HI, USA.

[9] Bohn D., 2019, Microsoft's HoloLens 2: a $3,500 mixed reality headset for the factory, not the living room, The Verge, February 24, 2019, https://www.theverge.com/2019/2/24/18235460/microsoft-hololens-2-price-specs-mixed-reality-ar-vr-business-work-features-mwc-2019

[10] Boschert S. and Rosen R., 2016, Digital twin – the simulation aspect, in Mechatronic Futures: Challenges and Solutions for Mechatronic Systems and Their Designers, pp. 59–74.

[11] Shafto M., Conroy M., Doyle R., Glaessgen E., Kemp C., LeMoigne J. and Wang L., 2010, Modeling, simulation, information technology & processing roadmap – technology area 11 (DRAFT), NASA, November 27 (2010).

[12] Rostami M., Pradhan P., Karki N., Omorodion J., Milani P., Kamoonpuri J., Liu H.Y. and Chung J., A comprehensive review of extended reality and its application in aerospace engineering, Department of Aerospace Engineering, Toronto Metropolitan University, Toronto, Ontario, M5B 2K3, Canada.

[13] Ahamed S., Das A., Tanjib S.M. and Nahar Eity M.Q., 2020, Study of an application development environment based on Unity game engine, International Journal of Computer Science and Information Technology, Vol. 12, pp. 43–62

[14] Garg V., Singh V. and Soni L., 2024, Preparing for space: How virtual reality is revolutionizing astronaut training, Proc. IEEE International Conference for Women in Innovation, Technology & Entrepreneurship (ICWITE), pp. 78–84.

[15] Hochberg L., Mulino J., Phillips D., Griffith C., Gupta A., Cecil J. and Lopez-Aramburo A., 2019, Exploring the role of Simulation-Based Design principles in support of transportation and material handling activities for the Lunar Mission, Proc. IEEE International Conference on Systems, Man and Cybernetics (SMC), Bari, Italy, October 6–9, 2019.

[16] Garcia A.D., Schlueter J. and Paddock E., n.d., Training Astronauts using Hardware-in-the-Loop Simulations and Virtual Reality, NASA Johnson Space Center, Virtual Reality Laboratory, Houston, USA.

[17] Franchini G., Tuberga B. and Chiaberge M., 2024, Advancing lunar exploration through virtual reality simulations: a framework for future human missions, Proc. International Astronautical Congress, Milan, Italy, 2024.

[18] Casini A.E.M., Maggiore P., Viola N., Basso V., Ferrino M., Hoffman J.A. and Cowley A., 2018, Analysis of a Moon outpost for Mars enabling technologies through a Virtual Reality environment, Acta Astronautica, 143, 353–361.

Ringraziamenti

Premessa: non sono bravo a fare queste cose e scusate se sembra più un elenco che un testo di ringraziamenti, siete tantissimi. Proverò a elencarvi tutti (se dimentico qualcuno mi tocca un’eterna infamia) e probabilmente sarà un po' cringe però fatevele andare bene perché ci ho provato :)

Volevo ringraziare tutte le persone che mi sono state vicine in questi anni qui a Torino e che sono state fondamentali nella mia vita e per i miei successi.

Innanzitutto, i miei genitori, mio fratello, i miei nonni, i miei zii, i miei parenti e, in generale, tutta la mia famiglia che mi ha sostenuto sia dal punto di vista emotivo e personale che da quello economico e che mi ha sempre permesso di fare ciò che desideravo.

Poi volevo ringraziare tutto il mio gruppo di amici storici di Palermo: Andrea, Dani, Matteo, Peppe, Pie, Simo, Orazinho, Rosario, Cla, Chiara, Giorgia, Alberto, Michelangelo, Melania, Roberta, Roberto, Marta, Martina, Alessia, Luca, Simona, Cris, Lollo. Con alcuni di voi ho condiviso questo percorso a Torino, con altri mi sono sempre tenuto in contatto a distanza su Discord o visto solo durante le pause e le vacanze, ma la cosa importante è che sono felice che facciate parte della mia vita e vi voglio bene.

Un ringraziamento anche a tutti i miei amici del gruppo di scherma di Palermo: Sofi, Alberto, Virginia, Alice, Ciccio, Stefano, Matilde, Sara, Riccardo, Dario, Giada, Adalberto, Chiara e Gaia, mi mancate e ci vediamo presto.

Non posso non citare un altro importante pezzo che si è aggiunto nella mia vita in questi due anni e mezzo qui a Torino. Tutti voi che ho conosciuto qui, mi avete fatto sentire come a casa e solo grazie a voi sono riuscito a completare questo percorso. Tutti i miei colleghi e amici del gruppo season e oltre: Giuseppe (comandante), Fry, Irene, Vints, Linda, il Sale (Pietro), Marco, Peppe (Rib), Mattia, Gilda, Tony, Cami, Matteuzzo, Gabri, Nicola, Salvo, Giorgio, Michela, Giulia, Laura, Nino, grazie per aver condiviso questa avventura con me; siete stati l’ancora di salvezza in questa tempesta di esami ed esaurimenti. Sono sicuro che ci vedremo sempre, sia che io rimanga qui a Torino sia che trovi lavoro altrove.

Volevo anche ringraziare tutti i colleghi e amici gli amici delle colazioni torinesi e tutti i falsinei: Elena, Michele, Elvira, Monica, Roberto, Alessandro, Daniele, Giulio, Ioana, le due Chiara, Leonardo, Lorenzo, Niccolò, Quintilla e tutti i membri del team Flamin-1A.

Uno speciale ringraziamento anche alle mie amiche Rebecca e Valeria, che nonostante mi abbiano conosciuto come “Ingegnere Della Selva”, hanno capito (forse) che sono più modesto di così e sono diventate parte della mia vita e insostituibili.

Volevo anche ringraziare tutti i miei amici e le persone che ho conosciuto fino ad oggi, tutti quelli che, anche se non ho citato perché altrimenti saremmo stati qui per sempre (palermitani, gruppo di torinesi, amici montaganesi e molisani, italiani, polacchi o americani che siano), hanno fatto parte della mia vita e mi hanno reso la persona che sono oggi.

Grazie,

Da Matteo