Tartu University

Faculty of Science and Technology

Institute of Technology

Audris Mihailovs

Development and Validation of a Lunar Rover Digital Twin Using Unreal Engine 5

Master's thesis (30 EAP)
Robotics and Computer Engineering

Supervised:

Self-supervised

Tartu 2025

Resumee/Abstract

Kuupkulguri digikaksiku arendamine ja valideerimine Unreal Engine 5-ga

See lõputöö annab lühikese ülevaate kaasaegsetest tehnoloogiatest, mis on seotud kosmosetööstusega, keskendudes robootikale ja selle arengule füüsiliste süsteemide koopiate kasutamise kaudu simulatsioonitarkvaras ehk digitaalkaksikute abil. Edasi kirjeldatakse kasutatud tarkvaravahendeid ja potentsiaalseid kandidaate mitmeotstarbeliseks kasutamiseks tulevikus ning nende vastavaid plusse ja miinuseid.

Teine osa räägib Lunar Roveri digitaalse kaksiku loomisest Unreal Engine 5-s. Käsitletakse kogu digitaalse kaksiku ehitamise, rakendamise ja kalibreerimise protsessi, et see vastaks võimalikult täpselt füüsilisele maismaasõidukile. Samuti võrrelda simuleeritud andurite andmeid füüsiliste anduritega, et mõista kasutatava tarkvara erinevusi ja piiranguid. See töö annab aluse täpsele digitaalsele kaksikule, millele saab tulevikus tugineda ja rakendada veelgi arenenumaid süsteeme.

CERCS: T120 Susteemitehnoloogia, arvutitehnoloogia; T125 Automatiseerimine, robootika, ¨ control engineering;

Märksõnad: digikaksik, simulatsioonid, Unreal Engine 5, Kuu rover, arvutid, kontroll, robootika

Development and Validation of a Lunar Rover Digital Twin Using Unreal Engine 5

This thesis will provide a concise report on the modern technologies connected with the space industry focusing on robotics and its development using copies of physical systems in simulation software or digital twins. It will go on to describe the software tools used and potential candidates for multipurpose use in the future and their respective pros and cons.

Second section will be about the creation of the Lunar rover's digital twin in Unreal Engine 5. It will discuss the whole process of building, implementing, and calibrating the digital twin to be as accurate as possible to the physical rover. As well as compare simulated sensor data with the physical sensors to understand the differences and limitations of the software used. This work provides the basis for a precise digital twin on to whom future work can be built upon and more advanced systems implemented.

CERCS: T120 Systems engineering, computer technology; T125 Automation, robotics, control engineering.

Keywords: digital twin, simulations, Unreal Engine 5, lunar rover

Contents

List of Figures

List of Tables and Graphs

Abbreviations, Constants, Terms

ROS - Robot Operating System

UE – Unreal Engine

DT – Digital Twin

PC – Personal Computer

NASA - National Aeronautics and Space Administration

BP – Blueprints in Unreal Engine

1 Introduction

Space exploration has been a uniting and a driving force for the humanity in the past century[1]. More and more missions are planned for surface rovers however their research and development path still follow the old ways of building and testing everything on-site[2], [3]. One of the most famous recent Mars rovers Perseverance has a copy of it on Earth named Optimism[4]. Everything that the Perseverance rover on Mars will do Optimism first tests it on ground. Whether that would be adjustments in communication, image capture or path planning. Even though this is an excellent way to ensure the success of a mission it is heavily limited by the necessity to devote large groups of scientist who will work relentlessly to ensure proper conditions and hazards that would resemble what Perseverance could be exposed to[5]. And this type of a trend is noticeable throughout the space industry. However, with Industry 4.0 advancing more and more possibilities of virtual testing are available. DAS, SPICE, JEOD are just a few software platforms that can be used for space mission simulations[6]. However, these platforms are mostly meant for satellite missions and trajectory planning.

1.1 Problem Statement

Lunar exploration and the quest to explore other planets and objects are the next goals for the space industry [7]. As more and more companies and countries plan their lunar missions they all face a similar challenge – how to test their rovers for these harsh conditions [8]. Simulations are one of the possible solutions for this problem. Creating an environment which mimics that of an extraterrestrial body is financially costly, logistically difficult and even sometimes impossible [9]. For example, to recreate the gravity for both lunar and space missions NASA uses their created pulley system – Active Response Gravity Offload System (ARGOS) [10]. But not all space agencies and companies have these types of budgets to create such solutions [11].

Simulations offer a viable option to recreate these conditions at a fraction of the cost[12]. Providing the option for a wider range of companies and countries to participate in such missions. However, wide usage of simulations for full scale mission planning and testing is a field that is yet to be explored thoroughly. And as technological capabilities develop at a pace higher than full scale missions happen it is difficult to utilize the full potential of them [13]. Unreal engine 5 is a potential solution for creating cost-effective, flexible, and scalable testing environments for lunar rover missions[11].

This thesis aims to prove the viability of Unreal Engine 5 as a simulations software for creating a lunar environment and testing rovers' capabilities. Creating a digital twin of an existing rover (Kuupkulgur Lunar Rover), implementing onboard sensors and parameters of the real rover, and testing its capabilities both in software and in real life [14]. Furthermore, by analysing the discrepancies and the capabilities of the two rovers it will be possible to deduct whether Unreal Engine 5 is a viable software for lunar mission planning and rover prototyping.

1.2 Objectives and Roadmap

Research Objectives

RO1. Develop a high-fidelity Digital Twin of the Kuupkulgur Lunar Rover in Unreal Engine 5 using SolidWorks and Blender. Implement blueprint functionality for its driving and suspension systems, calibrated based on real-world performance data.

RO2. Implement and validate the digital twin's measurement instruments (wheel encoder, IMU) by comparing simulated and real-world sensor outputs under controlled conditions, focusing on accuracy and response characteristics to better understand the differences between the digital twin and the real rover.

RO3. Perform a mission with current rover specifications on the lunar surface. Evaluate the rover's system performance in a simulated extraterrestrial environment.

RO4. Establish a validated framework for using Unreal Engine 5 in lunar rover prototyping and mission simulation. Provide reusable digital assets, performance benchmarks, insights, and recommendations to support future research in space robotics simulations and contribute to the successful planning and execution of the Kuupkulgur Lunar Rover's future mission.

2 State of the Art

This section will give an overview of digital twins for the purpose of prototyping and mission planning, as well as Unreal Engine 5 as a potential system for simulation creating focusing on its scientific aspects even though it has been mostly only used in the gaming industry.

Digital twins have revolutionised robotics development by representing physical assets either components or system of systems facilitating real-time simulation, analysis and optimization[15]. These advanced models integrate real-world data, predictive algorithms, and computational modelling to predict and simulate robots performance in diverse environments[16]. It acts as a live mirror of a physical robot, potentially evolving based on feedback from operational data[17].

One of the key advantages of digital twins is their cost-effectiveness. Unlike traditional prototyping methods requiring the production of physical components, their assembly and testing can be conducted virtually, significantly reducing development time and material cost. Furthermore their ability to support rapid iterations, real-time simulations, and scenario-based testing allows for more efficient design optimization without the need for constant physical modifications [18]. However, the development of high-fidelity digital twins can be a lengthy process. Based on the systems complexity, data integration requirements, and computational demands their implementation can take from months to even years [19]. On top of that, acquiring specialized knowledge of model simulations, sensor integration, and data-driven system optimization can be a hurdle for widespread adaptation as even obtained basic knowledge and proficiency of Unreal Engine 5 can take anywhere between 3-6 months [20].

2.1 Digital Twins in Space Robotics

The necessity of digital twins in space robotics grows each day as more and more countries and private organizations begin their journey on this path [21]. Due to the unique challenges of the environment these machines must operate in – vacuum, radiation, microgravity – testing whole systems and individual parts is a critical component for the success of the mission. And because of the lengthy process and high costs of transporting equipment to lower earth orbit and further failure of a mission can not only lead to significant losses but even bankruptcy for the private sector[22].

National Aeronautics and Space Administration (NASA) made the initial proposal of digital twin technology back in 2002 with the goal in mind to enhance the reliability and safety of mission critical equipment and systems by performing virtual tests in specific environments [23]. Till then most equipment testing took place either “on paper” by performing calculations and making assumptions on equipment capabilities and specifics or by building the actual systems and trying to recreate certain conditions to test the equipment longevity and effectiveness[24]. For example, ensuring component functionality in vacuum environments requires the usage of thermal vacuum chambers which simulate the temperature and pressure

of a space environment[25]. By applying principles such as “test as you fly” hardware is exposed to these harsh conditions and repeatedly and thus ensured of its capabilities to complete its missions. It is crucial to run such tests however they can also be very expensive and often times require specialized equipment to be made[26].

One of the most notable space applications is the James Webb Space Telescope (JWST). Being the largest and most powerful space telescope ever launched thorough and rigorous testing had to take place to ensure mission success [27]. Its currently orbits the sun is approximately 1.5 million kilometres away from earth. Due to the distance servicing this system is very complicated thus it had to undergo detailed pre-flight checks and testing to ensure all systems would be functional and operate as intended. One such crucial simulation testing was used for testing the heatshield on James Webb Space Telescopes[28]. Due to the delicate infrared detectors it was important to ensure proper working temperatures for these sensors and thus it was required to create a new type of heatshield which would unravel when at a specific moment after launch[29], [30].

Another important section that required simulation was the mirror alignment. Ansys Zemax OpticStudio was a software used to solve this issue due to its capabilities to evaluate mechanical stress effects and visualize ray tracing as well as operative with different lens designs[31]. JWST has 18 hexagonal segments mounted to a backplane structure which reach a size of approximately 6.6 m and each one of them has to be precisely aligned to achieve the best possible quality for the sensors as they mimic a single mirror[32]. Zemax software provided engineers the capabilities to design and test every step of the alignment process by starting with initial segment search and performing the final Fine Phasing thus ensuring mirror alignment would be optimal when the actual launch would happen [33].

Another interesting example to look at is Ingenuity Mars Helicopter. It was initially designed to perform five test flights in 30 days to research and demonstrate powered flight in the thin atmosphere of the Mars [34]. First successful mission was executed on April 19, 2021 and over time more and missions were performed slowly shifting from “Technology demonstration” to an “Operation Demonstration”[35]. Ingenuity Mars Helicopter prioritized safety and simplicity of the mission [35]. As communication was available only at certain moments mission had to be done autonomously and thus every sequence and flight plan had to be discussed, planned and most importantly simulated with a digital twin to better understand the limitations and mission success. Crucial factor about flight on Mars is the fact that its air density is only about 1% of what is on the surface of the Earth as well as one third of Earth’s gravity, even though reduced gravity helps in terms of achieving flight the extremely thin atmosphere is a big hurdle for helicopter typed drones as it is more difficult to achieve flight [34], [36]. Testing the drones capabilities for these environments is very difficult and costly on earth and thus the best solution for this problem is to use a digital twin in a simulated Mars environment and recording data that can be later on analysed and used for improvements or design changes to achieve a successful mission. One of the tools that was created specifically for this purpose is the Helicopter Control Analysis Tool (HeliCAT) which uses the Darts/Dshell multibody simulation framework which was created at JPL [37]. Most notable features of this simulation tool are: detailed modelling of actuators and sensors as well as camera imaging, modelling of ground contact dynamics capable of determining and functioning with varied terrain and surface properties, modelling of ground

support equipment, for example gimbals and force-torque sensors that allow to validate system identification, flight software integration and 3D visualization of the experiment. HeliCAT also offers the possibility to execue end-to-end mission simulations with flight software in the loop thus providing a full set of data about the digital twins performance and it's limitations [38]. One of the limitations for this software is the fact that it models the components as rigid bodies connected with hinges and thus there are limitations on the functionality of flexible bodies. However, one way to counteract this is to use CAMRAD II, another widely used software used in NASA, as it offers a higher-fidelity especially for flexible bodies [39]. This provided the researchers with better understanding of certain properties related to thrust and power performance and to fully understand the mission feasibility engineers had to simulate their designs in both software's.

Making a digital twin model requires to recreate the real system with certain specifications in mind to ensure the results gotten are valid and coincide with what one could expect from a real system in the specified conditions [40]. Examining more thoroughly how researchers at NASA approached this design the model consisted of a fuselage, the mast, blades and landing gear, and the connections of the blades were configurated with a sequence of hinges with different properties [41]. Main part of these simulations is to test the aerodynamic forces on the blades and this is achieved by dividing the blade into numerous individual sections each having their own individual lift, drag and pitch moment as a function of angle-of-attack and Mach number while the fuselage drag is modelled as a blunt-body drag model with no specific angle-of-attack dependency because of the low drag density of Mars it has a very small impact on the flight dynamics[42]. This type of an approach is one of the ways how digital twins can be simulated and tested in virtual environments as testing individual component efficiency can bring light to the overall efficiency of the system.

National Aeronautics and Space Administration being one of the top space exploration organizations have conducted numerous missions and used digital twins for testing and prototyping purposes and one such notable mission is the OSIRIS-REx and later OSIRIS-APEX mission [43]. This mission's goal was to collect a sample from an asteroid Bennu, deliver the sample capsule back to Earth and then continue its mission to explore the asteroid Apophis. One way digital twin simulation was used for this mission was to study the capsules entry to earths atmosphere and the aerodynamic effects, sequencing and triggering logic events for example parachute deployment [44]. Notable part of this was the collaboration of multiple companies each with different tools that allowed the scientists to compare discrepancies between each of the simulation data sets and understand fully the effects and mission readiness status. Due to the complexity of this mission another simulation test was used to to determine the trajectory the satellite would need to approach the asteroid and the place it would make contact with it and when it would happen [45]. This was all needed to gain knowledge of the place and angle the satellite would touch and when to arm the gas bottles which are responsible for collecting the samples. Most widely used satellite simulation usually are path planning which was also done for this mission. As OSIRIS-REx had completed it mains mission to the asteroid Bennu and delivered the samples back to Earth, its secondary mission could beggin, traveling and exploring the asteroid Apophis[46]. To achieve this scientists used a simulation software called MIRAGE which is used for navigation[47]. MIRAGE uses gravitational models of all solar system bodies which also includes solar, lunar and planetary masses thus providing an accurate estimation of the objects paths.

These and other examples showcase the capabilities and usefulness of digital twins and simulation software. But often times companies and organizations create specialized simulation software and digital twins to test certain capabilities and individual components of the systems which limits their reusability and effectiveness over time.

2.2 Simulation software choice

Given the steep increase of computational power and simulation software capabilities it was decided for this work to use a premade simulation software instead of creating something new and specialized. Thus, research and comparison was made between the currently available simulation software tools. Most notable ones are Unreal Engine 5, Gazebo, NVIDIA Omniverse. Each one's pros and cons will be discussed in next section as well as the ideology behind making the decision.

2.2.1 Gazebo

Gazebo was initially started in 2002 and is a collection of open-source software libraries whose goal is to simplify development of high-performance applications[48]. Main users of Gazebo would be robot developers, designer and educators, however as time has gone on and more and more libraries have been added it has been used in the professional industry aswell and can be a suitable simulation software for prototyping and mission testing. Being apart of the Open Source Robotics Foundation provides access to numerous tools and functionality as well as support for the development[48]. The possibility to integrate Robot Operating System (ROS) greatly enhances the development speed. The ease of use and wide support for this software has also benefitted the digital twin industry for space robotics. One such example is the work "Development of a High Fidelity Space Robot Gazebo-Based Simulator for Space Close-Proximity Operations Experiments"[49]. In this work researchers created a model of a system containing air bearing, nozzles, compressed air cylinders, solenoids valves, pressure regulator valves, electronic control board and most importantly a robotics arm, which was tasked to move around in the environment and grasp the object. Another compelling argument for this software would be its integration with OnOrbitROS and the development of Robonaut 2 which all focus on the usage of Gazebo simulation tool with extra packages for the purpose of space robotics [50], [51]. Even though certain steps are taken for research and further development these solutions still heavily realy on the support of community and outside libraries/pakcages which are often times specialized for a specific mission. Furthermore, a huge problem for this simulation tool would be its default basic physics engine which is meant only for simple simulation environments and systems[52]. Being library-based engine there have been certain advancements in this field by providing specialized libraries that increase the realism and precision of the physics engines, but this still relies on the community support. Gazebo has the potential to run on low computationally powerful machines and even on machines with no dedicated graphics cards if there are no environment obstacles[53]. But to achieve this Gazebo has had to reduce the quality and realism of its simulation designs which works for research purposes but might not be the most appealing look for the average person. As this tool is also meant for education its usability and skill requirements are low [54].

2.2.2 NVIDIA Omniverse

NVIDIA Omniverse was launch around 2021 and it is a platform of APIs, SDKs and other services that provide the tools for developers to integrate OpenUSD - Universal Scene Description which is an ecosystem for 3D world designs - NVIDIA RTX rendering technologies and generative physics AI with uses cases for industrial and robotic use cases [55]. NVIDIA Omniverse provides the support of additional plugins and app that provides certain functionality and capabilities such as Python and command line scripting and even apps providing AI integration[56]. Regarding physics engine NVIDIA Omniverse uses PhysX 5 which is an moderately advanced physics engine providing accurate data for industrial settings, robotics, autonomous driving and support for deep reinforcement learning[57]. Given the addition of NVIDIA Flow, NVIDIA Blast and NVIDIA Flex libraries this physics engine provides possibilities to simulate soft body dynamics such as liquids and cloth and more complex collisions. Given how new this simulation tool is there is a question on how accessible it is an how easy it is to use this application, furthemore as NVIDIA's focus is on professional application this could mean that to efficiently use this application the user would need sufficient background knowledge and previous experience[58]. NVIDIA promotes its realism and 3D world visualization which is a benefit in terms of appeal for the user however this also leads to heavy computational demands, as recommended graphics card for these is an NVIDIA RTX 4080 it would be difficult to efficiently use this simulation software on cheaper and more constrained computers[59]. Noticeable companies that use NVIDIA Omniverse are BMW Group, Amazon, Dyson[60]. However, research in the space industry sector has been scarce based on the paper published on 2022 "Exploring NVIDIA Omniverse for Future Space Resources Missions" by Xiao Li from the University of Luxembourg NVIDIA Omniverse was used as a tool for photorealistic visual effects while having MATLAB/Simulink for testing space system architectures[61]. This could be the result of NVIDIA's choice to focus on photorealism and visual effects for factories, product designs and virtual worlds on the cost of experimentation, prototyping and usage of digital twins in non-typical Earth environments.

2.2.3 Unreal Engine

Unreal Engine software has been created by the company Epic Games with the first iteration of this engine released around 1991 and was initially made for first person shooter games[62], [63]. As time went on it grew in popularity which also meant more and more advancements for the technology and more people working on. Unreal Engines newest version Unreal Engine 5 was launched in 2022 and marketed as the most advanced and open real-time 3D creation Tool[11]. With the new features such as Nanite - rendering technology for geometry systems enabling large scale visualisations of environments and objects - and Lumen - global illumination and reflection system creating more realistic environment luminations[64], [65]. Thus, providing game developers and engineers with the tools to make realistic and visually appealing environments and objects and their respective physical interactions. In terms of computational requirements Unreal Engine requires any graphics card that supports DirectX 12 although recommended would be NVIDIA RTX 3080 which is the typical system Epic Games engineers use[66]. Unreal Engine 5 uses Chaos physics engine which replaces PhysX engine

on previous version[67], [68]. Based on 2024 data Unreal Engine 5 was used by 4838 companies on numerous projects including games, visualizations and VR experiences and that number does not include all the independent projects or hobbies someone works on[69]. This means that the community support and guides on how to use this application is vast especially in the gaming industry because of its popularity. As the realism and efficiency of Unreal Engine has grown so has the amount of industries Unreal Engine is used in besides gaming for example, higher education, information and technology services, design, media, research[70]. Complexity of Unreal Engine heavily depends on the use of premade structures and interactions but in general it is estimated that to become a beginner takes approximately 1 to 2 weeks, however learning enough to be intermediate master of this software can take up to 4 months if you are developing a game[71]. Due to its popularity and efficiency numerous scientific research have been conducted on testing Unreal Engine as a tool for digital twin testing. For example, Unreal Engine has been used to setup a two robotic arm collaboration for testing their cooperation capabilities in a laboratory environment[72]. NASA is also one of the companies that has started to implement Unreal Engine in its research for example by creating a Lunar environment for making synthetic data to train image processing models which would be used for cameras when building the Tall Lunar Tower Assembly[73]. There have been multiple scientific studies on using Unreal Engines synthetic data creation to benefit the technology advancements for both on Earth usage as well as for more extreme environments[74], [75]. More notable use cases for Unreal engine would be its usage to prepare astronauts for the ISS station by creating a fully functional ISS station and immersing potential astronauts using VR technology into this environment[76]. Unreal Engine is also used to prepare for the next Lunar mission by creating a Lunar environment with detailed landscaping and using the Lumen technology scientists are researching how to use shadows for navigation on the south pole region[77]. It is also used to train airplane pilots and a project in collaboration with CAE using Unreal Engine created an aviation simulation which received level D qualification which is the highest one possible for full-flight simulators[78]. These and many more examples are proof of the efficiency and capabilities of Unreal Engine and sets it on the track to become more than just a game engine, but a tool for scientists to use and explore the capabilities of digital twins.

Efficiency, reduced costs and safety are all reasons for using robotics in the space industry and to achieve this digital twin bring significant benefits by improving testing precision and accuracy especially in hard to replicate environments while also reducing the costs that are linked to research and development[79]. As we approach Fourth Industrial Revolution such tools as mentioned above will become more widespread and even in some cases an industry standard thus researching them is a vital part of understanding their capabilities and limits[80]. Furthermore, due to the complexity of the environment that a Lunar rover must operate in and because of the limited funds and capabilities of the Tartu Observatory the best option to test mission success and equipment readiness is to utilize the effectiveness of a digital twin. Based on the studies described above it was concluded to choose Unreal Engine 5 as the simulation software due to its balance between graphics, performance and current widespread use in other fields and the efficiency of using nodes.

3 Methodology

This section will cover the steps taken to create and implement proper functionality of a digital twin in Unreal Engine 5. This will include – transforming file type, ensuring proper location of components, creating joints for the components, implementing keyboard functionality, simulating sensors and data collections. Computer specifications that were used for running the simulations: NVIDIA RTX 3060, 11th Gen Intel® Core(TM) I7 3.30GHz, 16GB ram.

3.1 SolidWorks and Blender

SolidWorks is one of the leading computer software's for 2D and 3D designing and CAD (computer-aided design) model creating and is an industry standard[81]. It was also used for this project to create the 3D models of components and their assembly. However, SolidWorks files are saved with the sldprt file extension which is not compatible with Unreal Engine 5. Thus, another application must be used to convert this file type into a usable one and in this research, Blender was used. Blender is an open-source software used for 3D visualization and animations, but it has also been growing in popularity for fields outside of animation[82]. For this work Blender provided the option to convert SolidWorks sldprt files to fbx which could then be imported into Unreal Engine 5.

Furthermore, Blender also has two other advantages – relocation, centring, assembly as well as reducing triangle count which will be described later. Firstly, all the components were imported into Blender (Figure 1).

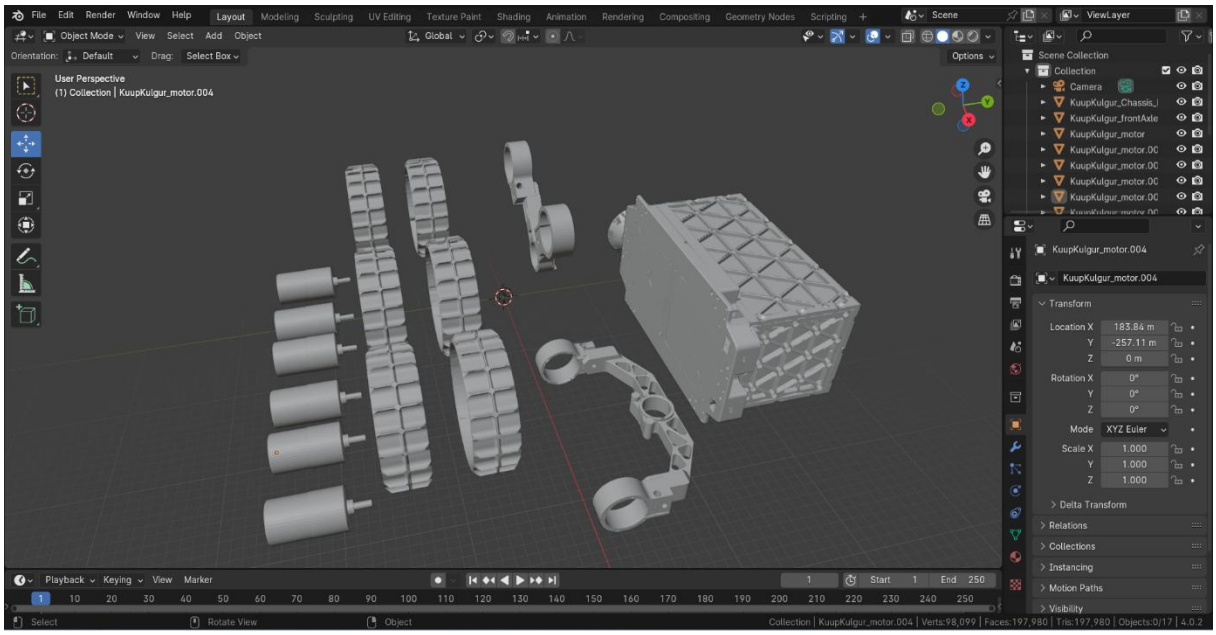


Figure 1. Lunar rover model disassembled in Blender

Next step in the process was to align and assemble all the components. This process was achieved by centring one part to another. In more detail for assembling the rear axel and main body. For real axel's mounting hole choose frontal face of said hole and centre the cursor in the between all the chosen points. Then this step is repeated for the main body's mounting section where the face points are chosen, then the centre point is marked. When both component centre points are marked, they can align by using the option of centring point to cursor thus achieving proper assembly of components. These steps were repeated for each component till the rover was fully assembled, for better separation of components they were also coloured (Figure 2).

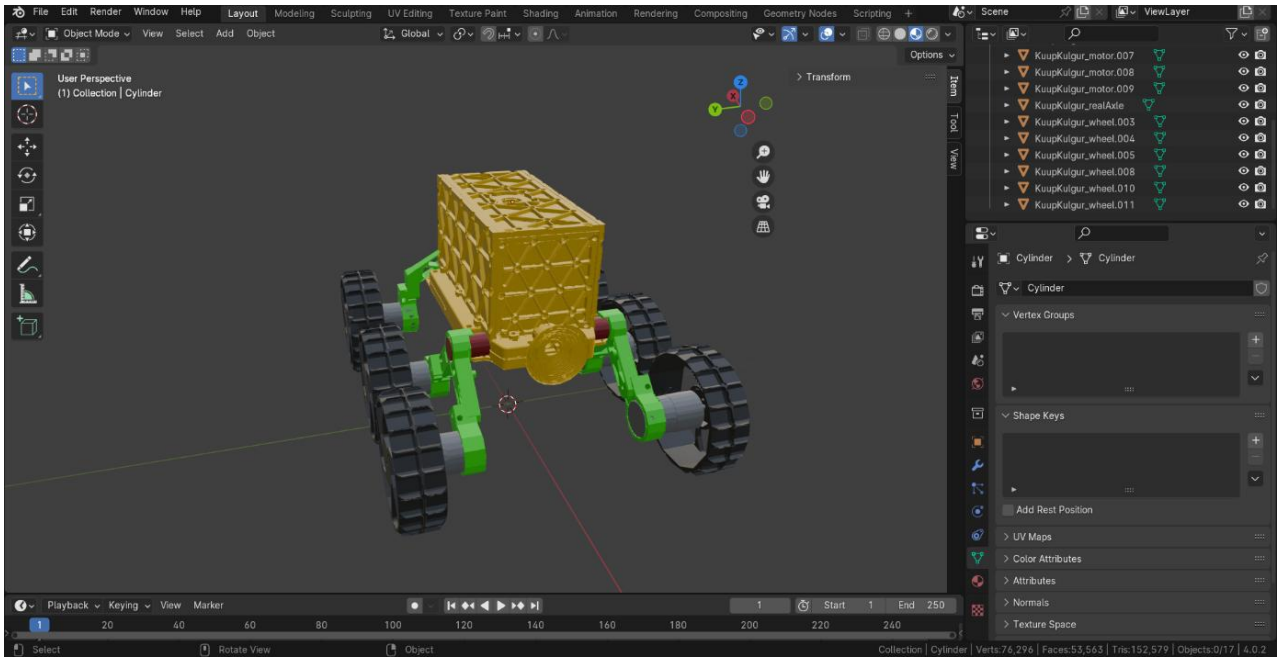


Figure 2. Assembled and coloured Lunar rover in Blender

A very important and useful feature of Blender is its capabilities to reduce point/triangle count of components. This is achieved by utilizing the options of "Decimate Geometry" which reduces the vertex/face count of meshes thus simplifying the structure[83]. Another option is to use "Degenerate Dissolve" which collapses any two edges which are shorter than certain length[84]. Third option for mesh simplification is "Limited Dissolve" which simplifies and merges vertices and edges that separate flat regions which is often especially useful for larger components as in this case for the main body if the Lunar rover[85]. The approach for this was to simplifies the mesh without causing too many discrepancies between the original and altered version. Thus, observing the frontal disc for the camera of the main body it is possible to notice that the difference between the meshes are very mildly noticeable however the number of points/triangles are reduced significantly (Figure 3).

By observing Table 1 it is possible to see the amount that the mesh has been simplified by. Overall count of vertices reduced after the mesh editing is 23069, number of edges reduced 134968, number of faces reduced 111895, number of triangles reduced 53699, thus the overall number of points reduced is 323631, which is a significant improvement. Main purpose of doing

this action is to reduce the computational requirements and load because a simulation software would have to calculate each component's location and interaction thus by reducing the complexity it is possible to improve the efficiency of simulations while not affecting the visual aspect of the components too much.

Number of points for component								
Component	Pre-edit				Post edit			
	Vertices	Edges	Faces	Tris	Verts	Edges	Faces	Triangles
Main body	61673	186873	124876	132422	59556	106694	46818	120631
Front axle left	3654	11034	7356	7356	1451	2068	593	2950
Front axle right	3654	11034	7356	7356	1451	2068	593	2950
Rear Axel	3258	9891	6600	6600	1418	1954	503	2916
Motor of rover (6x)	456	1362	908	908	280	441	163	556
Wheels (6x)	3761	6073	2300	7546	1486	2174	676	2996
Overall reduction	Vertices: 23069		Edges: 134968		Faces: 111895		Triangles: 53699	

Table 1. Comparison of pre-edit and post-edit of component mesh points

Last step of this process is to export the newly created assembly from Blender as a fbx file and import it into Unreal Engine 5 and reduce its size as Blender uses metres as its default measuring unit, while Unreal Engine 5 uses centimetres (Figure 4).

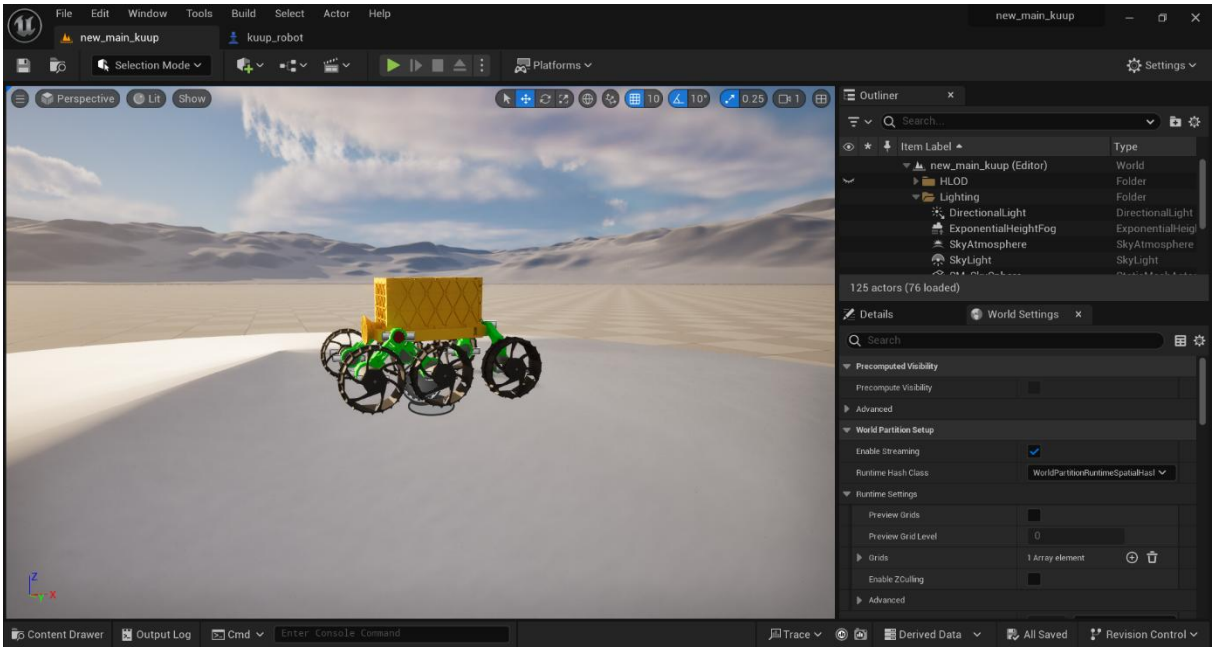


Figure 4. Lunar rover model imported into Unreal Engine 5 and resized to proper scale

3.2 Joint construction of components

After the importing of the file next step is to set up the correct joint dynamics between components otherwise the rover components will fall apart. Similarly to Gazebo, Unreal Engine also has different joint types. However, in Unreal Engine 5 these are called “physical constraints”. For components that require no movement between each other such as the motors and axel locations for said motors physical constraints are set up in a way to lock the two components together without any freedom of movement. Setting up a joint between two components can be achieved in two ways either by having skeletal meshes or by choosing two components. As this work doesn’t use skeletal meshes the setup is done by using “Constraint Actor 1” which refers to the main part for example the rear axle, while “Constraint Actor 2” refers to the component which will be joined to the main part for example the motor.

Furthermore, setup for moving components require additional steps, for the physical constraints between the front axels and main body an additional feature was used by enabling rotation on the X axis or in terms of Unreal Engine limited “Swing motion” of 15 degrees. The physical rover has a mechanical stop at this angle however there is no need for this type of a system as Unreal Engine supports a swing motion limit. Due to how the rear axle is positioned instead of swing motion rear axle requires limited “Twist motion” which refers to the Z axis. Similarly to front axels physical rovers also has mechanical stops for rear axle which were replaced by limits on the twist motion which was set to 25 degrees. Additionally, to ensure proper movement of axels an additional feature was used “Target Orientation Strength” for front axels for swing motion and for the rear axel twist motion and the unit strength for this was set to 1000 centinewtons which corresponded to the physical rover’s springs in the axels.

Wheels also require modified physical constraints. Being set between the motors and the wheels physical constraints need the option of free rotation around the Y axis. Additionally, another option specific for wheels is used called “Angular Motor Target Velocity” for swing motion. For this specific setup strength for target velocity was chosen as 5300 centinewtons and the target velocity on y axis rotation was 0.18 revolutions per second. The choice for these specific values will be described later in the work.

3.3 Keyboard control

This section will cover the steps taken to implement keyboard control for Lunar rover digital twin. Initial design of the controller consisted of two part – controller and main blueprint section. By assigning each keyboard keys to specific outputs and specific wheel angular motors (Figure 5).

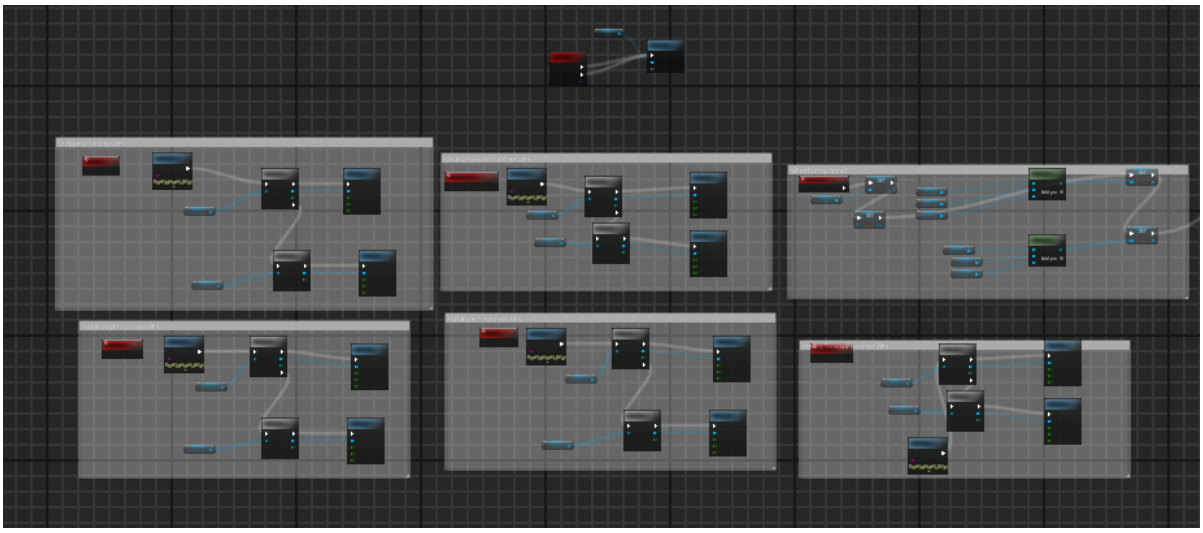


Figure 5. Initial design of keyboard controller overview

This allowed for simple setup consisting of only two blueprint sections and very direct action between keyboard button and the motor control (Figure 6).

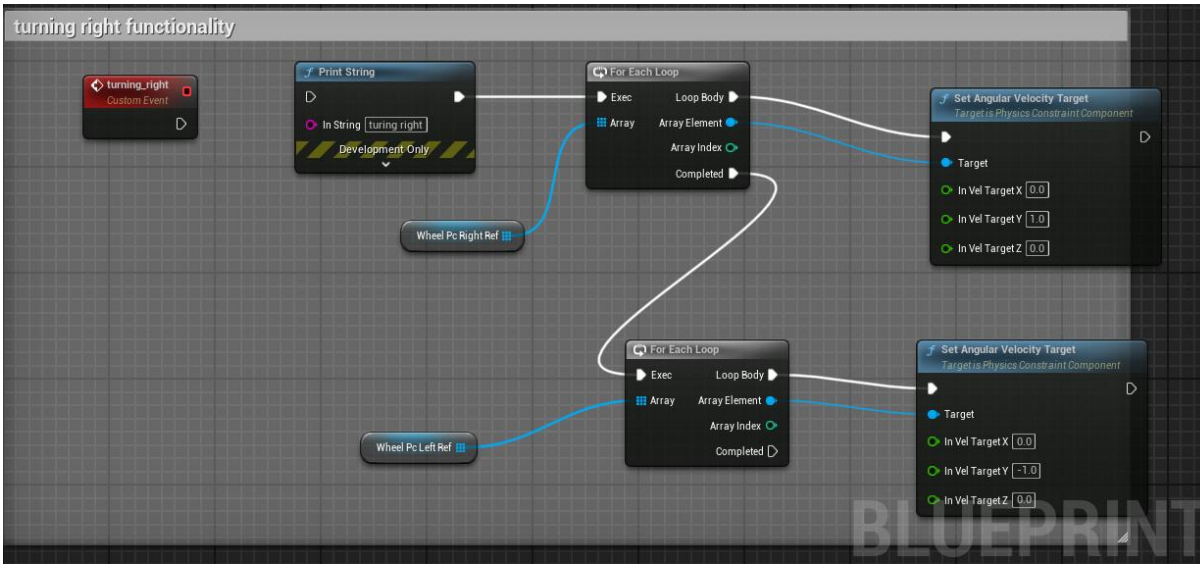


Figure 6. Zoomed in version of initial design of keyboard control

However, to utilize the full capabilities of Unreal Engine 5 and their new system of Enhanced Input Action the old design was completely reworked. Enhanced Input Action provides the possibility to remap inputs during runtime and create more complex and priority driven inputs. This also comes with its drawbacks such as overall complexity of input mapping and multiple blueprints that must be implemented. First step of this process is to create a Data Asset blueprint

which represents an abstract game action in which and important step for this work is to use the available option of “Action Value” to be “Axis2D(Vector2D)” this provides the necessary values that are used later (Figure 7).

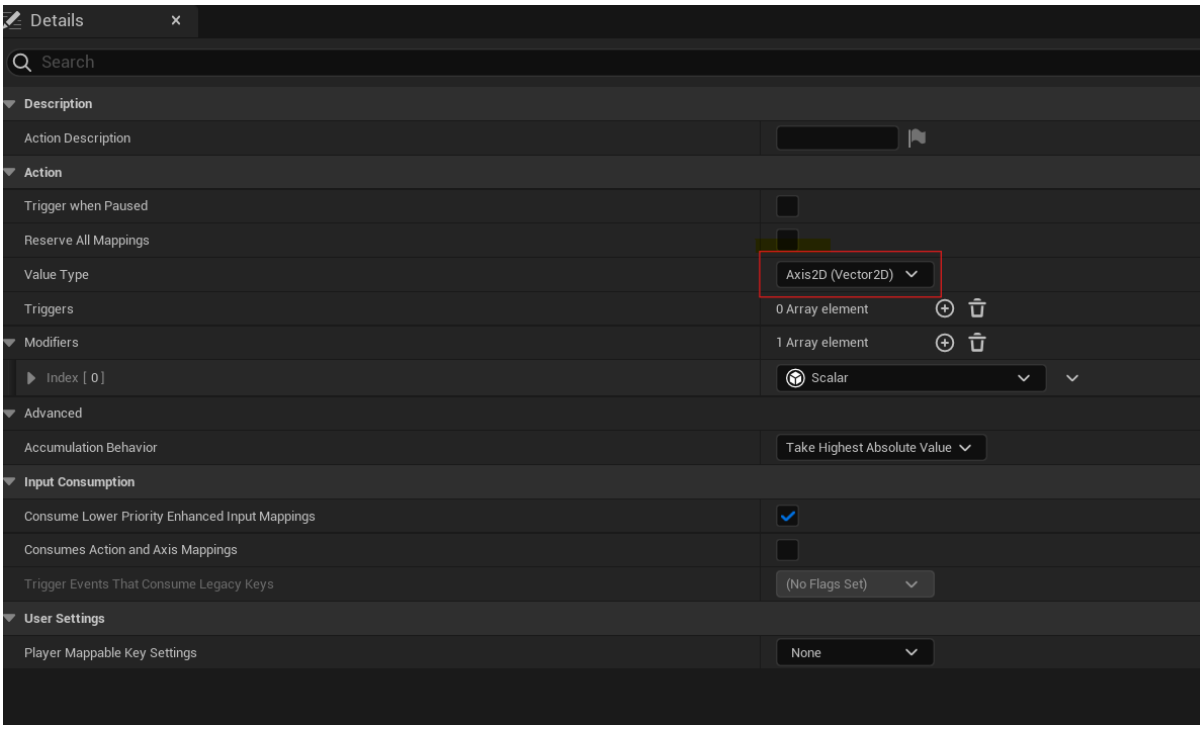


Figure 7. Data Asset of an abstract game input

Next step is to create Data Asset Group which is a collection of Data Asset inputs that are mapped to action. This section requires additional options to be selected. Data Asset group assigns certain keyboard buttons to specific values and controls and in this case keyboard button D has no modifiers and rest of the keyboard buttons are adjusted according to this button. Meaning button “A” values is negative compared to the letter “D”, keyboard button “W” needs the modifier of “Swizzle Input Axis Values” which means to switch the axis based against “D”, lastly letter “S” has the modifier same as “W” with the addition of being negative compared to D (Figure 8). These steps are required as a setup before implementing the Enhanced Input Action into the main blueprint.

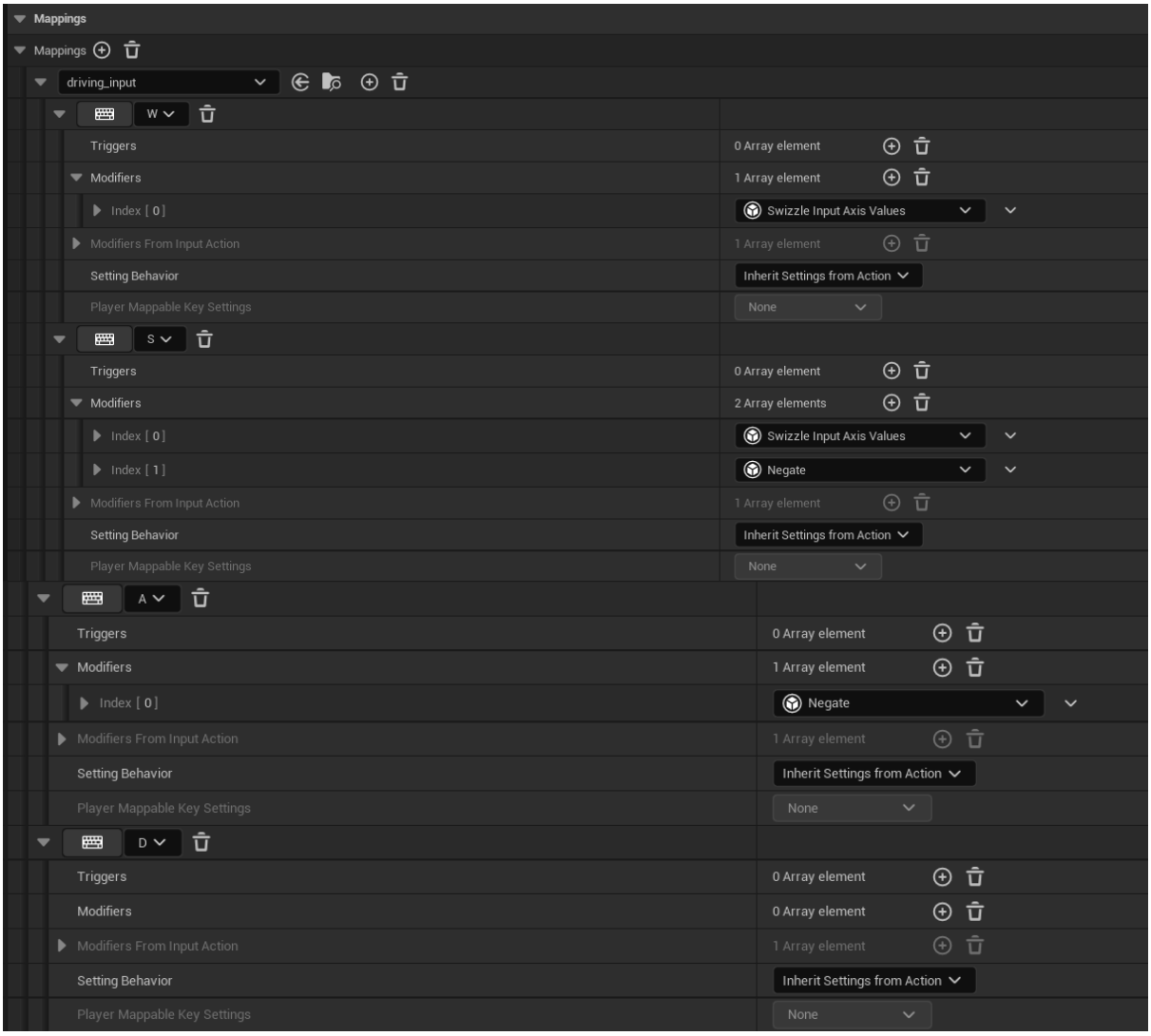


Figure 8. Data Asset group settings for individual keyboard inputs

To implement the Enhanced Input Action in the blueprints using nodes it needs to be connected to “Add Movement Node” as it triggers each time from a button press. Using Pawn setup, the values are generated each time as the chosen button is pressed and provides the direction in which the Pawn should move or an action to happen (Figure 9). This offers a simpler design in blueprints but requires more setup before.

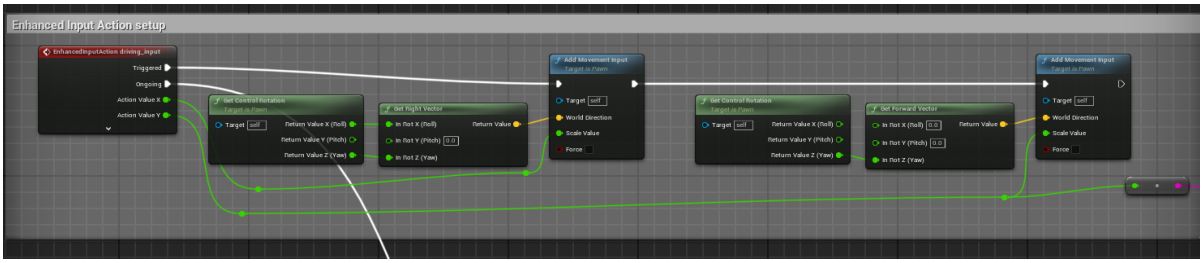


Figure 9. Enhanced Input Action node setup in blueprint

3.4 Additional Component setup

3.4.1 Camera implementation

Cameras are required to monitor the pawn during experiments, and the setup consists of two cameras – one placed on a spring arm providing third person view while the other is located at the front of the rover providing first person view (Figure 10). Spring arm provides the option of moving the camera around using a computer mouse by placing a set of nodes in the blueprint that provide relative rotation based on whether the mouse is moved on X or Y axis. Additionally, for switching between the camera modes a node called “Flip Flop” was implemented that switches between two options based on keyboard input.

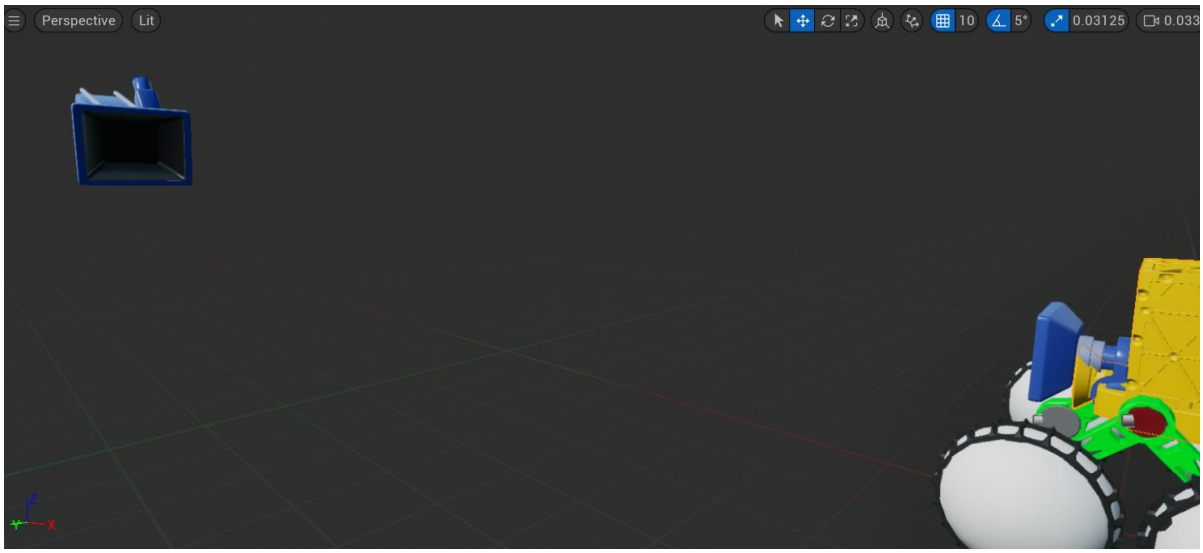


Figure 10. Two camera setup for the rover in Unreal Engine 5

3.4.2 Data saving

Due to how Unreal Engine operates a plugin was needed to collect data and save them in a text file. For this purpose, the plugin “FileSDK" was implemented. This provides the necessary nodes to incorporate a data saving functionality. After installing and enabling this plugin for the project the required node setup consists of creating nodes that are used for the naming of the file, choosing the location where to save them, collecting the data in an array of strings and using the “Write String to File” node (Figure 11).

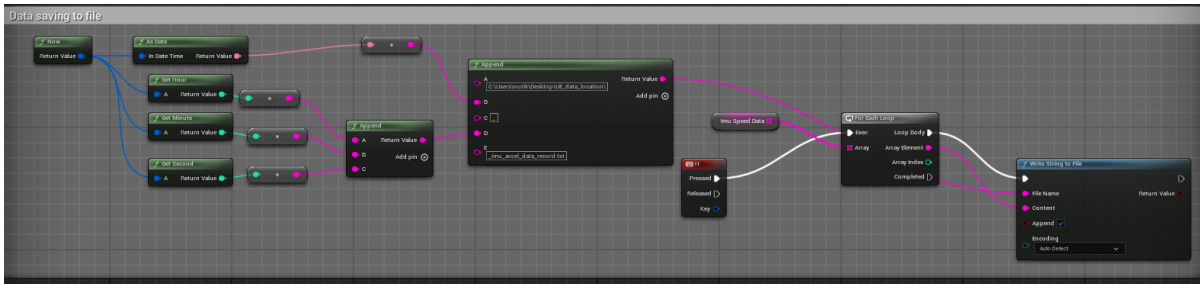


Figure 11. Data saving file node setup in blueprint

3.4.3 Simulating sensors

For the purpose of this work and the accuracy of a digital twin two sensors were implemented in this simulation model – sensor resembling a wheel encoder and an IMU sensors.

Simulating a wheel encoder requires an implementation of an "Event Tick" node which provides constant action at each tick which is connected to a "Get World Rotation" node that has the pawn's wheel connected to it, thus providing a rotational value at each tick. Last step of this process is to save the value of rotation and implement a node functionality to subtract the previous value from the new to get the rotational value change which is then saved to an array of strings and later written to a file (Figure 12). Wheel encoders provide the wheel rotational speed value in degrees per second[86].

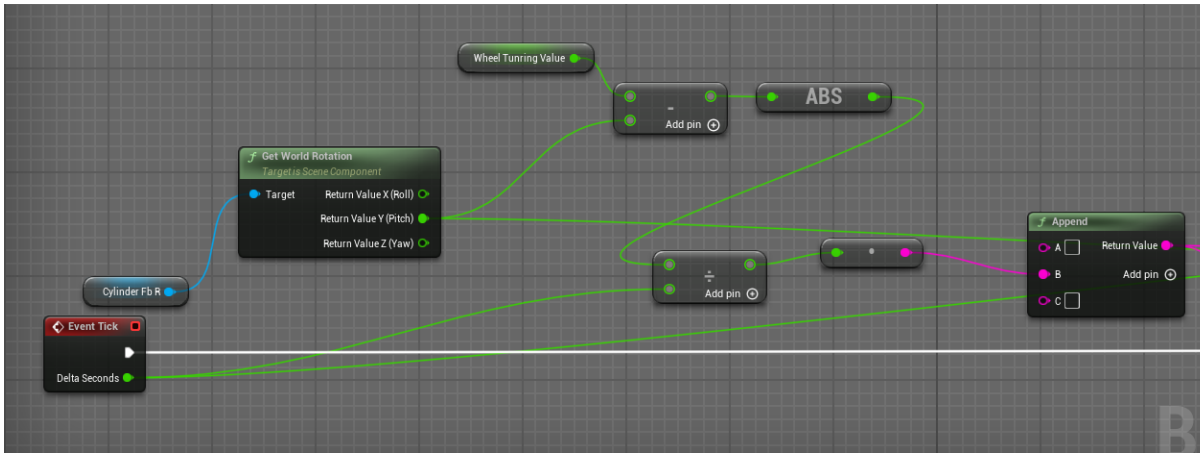


Figure 12. Simulated wheel encoder in Unreal Engine blueprint

IMU or inertial measuring unit is the combination of an accelerometer and gyroscopic sensor [87]. To implement a realistic IMU sensor a small rectangle was placed inside the main body of the rover digital twin. Furthermore, to receive the gyroscopic data "Get World Rotation" was used and similarly to wheel encoders setup previous value gets subtracted from the new value (Figure 13). Data gets saved to an array of strings and then written into a file.

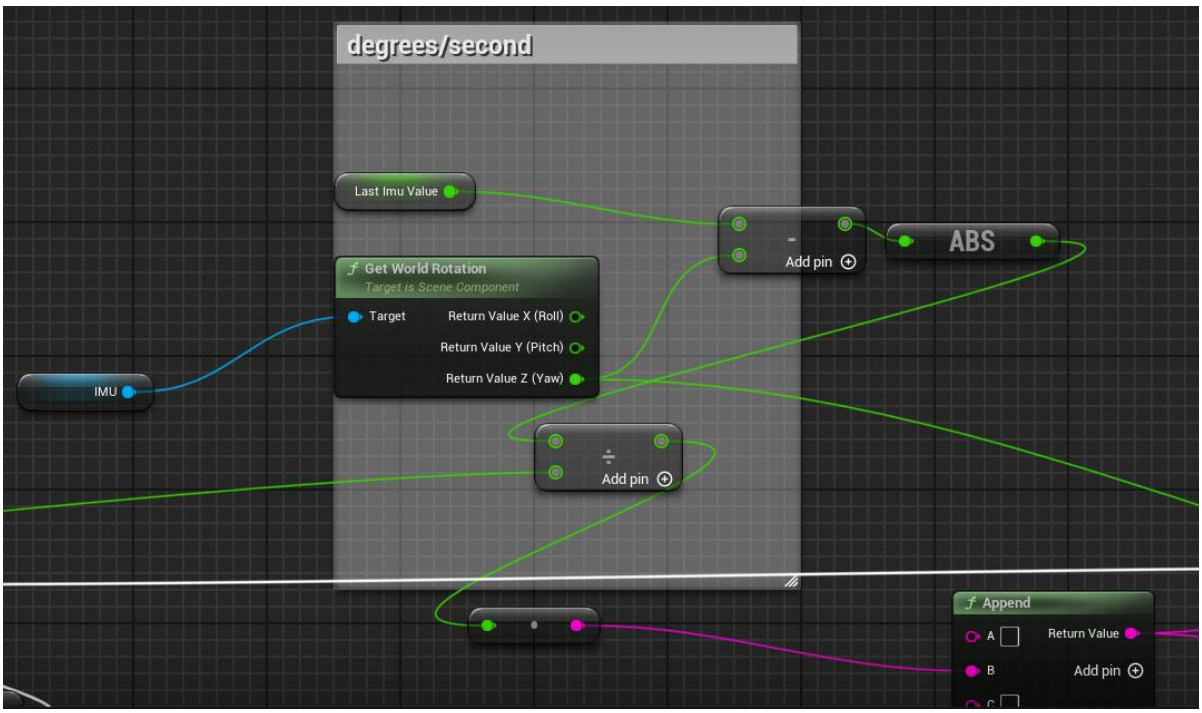


Figure 13. IMU gyroscope implementation in Unreal Engine 5

Accelerometer requires a slightly different use of nodes compared to gyroscope; it uses "Get Physics Linear Velocity" which provides the linear value on a chosen axis. Rest of the setup is similar to previous ones by subtracting the old data from the new, appending an array of strings and writing the strings to a file (Figure 14).

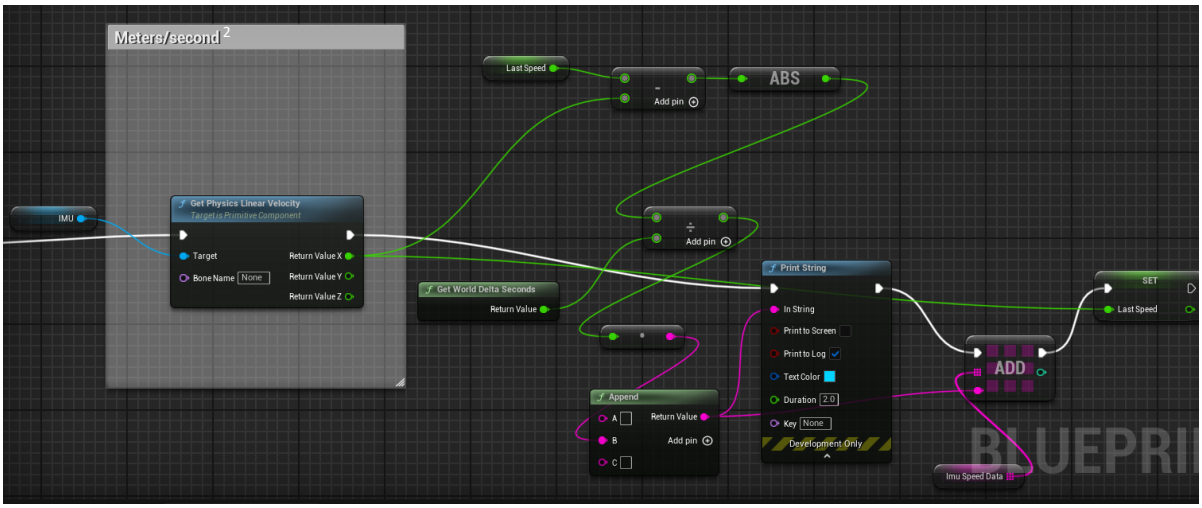


Figure 14. Accelerometer implementation in Unreal Engine

4 Experimental Results and model modifications

This chapter will cover the experiments conducted and how the obtained data affected the model and improvements made to achieve a more accurate digital twin model of a Lunar rover in Unreal Engine 5 as well as the value choices described in the previous section.

4.1 Weight measurement

To achieve an accurate digital twin model one of the first steps required was to measure the weight of the components of the physical rover. This required to disassemble the physical rover and weigh each part separately so these values could be inserted into Unreal Engine. For research purposes weight measurements were made in a more detailed manner and for each individual component that could have been disassembled however the data connected with Unreal Engine is visible in Table 2, as digital twin model had a simpler configuration and required the weights for component assemblies.

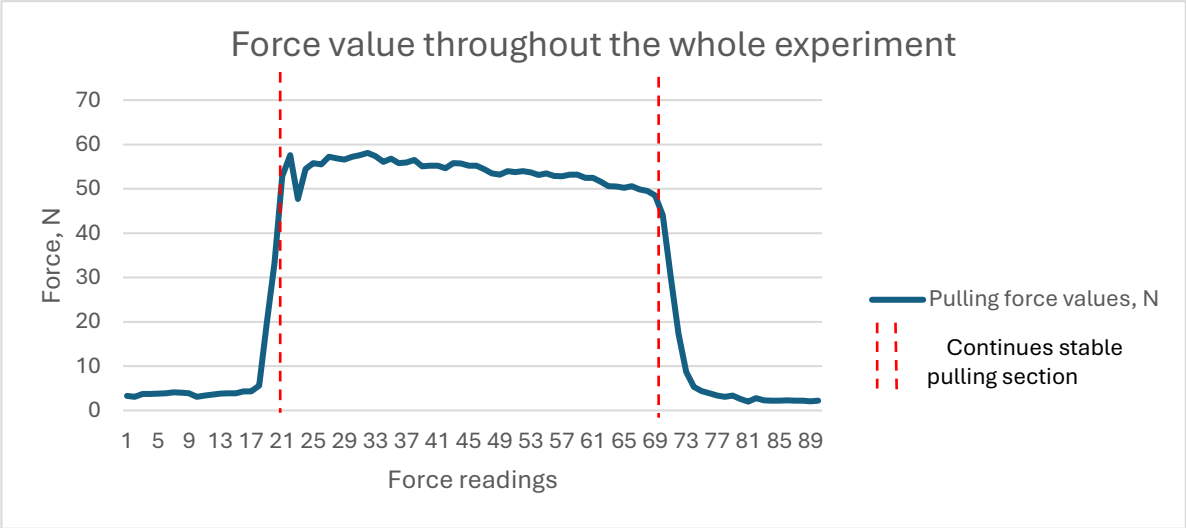
WEIGHTS FOR UNREAL ENGINE PARTS	
Component	Weight in grams
Wheel	146
Wheel motor	298
Back suspension	754
Front Suspension 2x	710
Main Body	780
Overall weight: 4908	

Table 2. Physical component weights used in Unreal Engine

After the measurements these values were used to adjust the component weights in Unreal Engine by adjusting the component weight in the pawn settings.

4.2 Dynamometer measurements

Dynamometer experiment was used to measure the pulling strength of the motors by tying one end of the string to the wheel while the other part is connected to the dynamometer. This test was conducted multiple times both in continuous pulling strength measurement setting and max force pulling strength (Graph 1).



Graph 1. Pulling strength of rover experiment data

To calculate the average pulling strength from all the experiments it was decided to use the max pulling strengths (peak force) measured as well as the pulling strengths of continues measurements were pulling could be accurately observed which is visible in Graph X in the section between red, dashed line. Overall average values from said experiments are visible in the Table 3 and the average value between all of these experiments is 55.5103 N.

Experiment number (continues pulling - C, peak force – P)	Force value, N
1 C	54.1917
2 C	49.8108
3 C	54.3869
4 C	50.5826
1 P	63.9000
2 P	58.1000
3 P	57.6000
Average pulling force value: 55.5103 N	

Table 3. Dynamometer experiment pulling strength measurement average data

4.3 Speed measuring experiment

Experiment for measuring the speed of the physical rover was conducted by marking a 4-meter distance on a solid, non-slippery floor and using a stopwatch to measure how long will it take for the rover to cross the distance. Rover was placed behind the line with enough distance for it to achieve maximum speed thus ensuring that the whole distance is covered with a stable speed and stopwatch was only started/stopped when the rover passed the line. Time in seconds from the 3 experiments is as follows: 22.39, 22.47, 22.48. Average time for the rover to cross the 4-meter distance was 22.45 seconds. Knowing the time it took for the rover to cross the distance, and the size of the wheels it was possible to calculate the degrees per second the wheel spins by calculating the circumference, then calculating the number of rotations the wheel does over the distance, then getting the time taken for a rotation and finally converting it to degrees per second which comes out to: 185.57 degrees per second.

This value was used to modify the digital twins speed by running numerous experiments in the simulation to a point where the output printed from "Get World Rotation" matched that of the calculations of the physical rover. When the correct value was found it was inserted into the "Target Velocity" section on physics constraints visible in Figure 15.

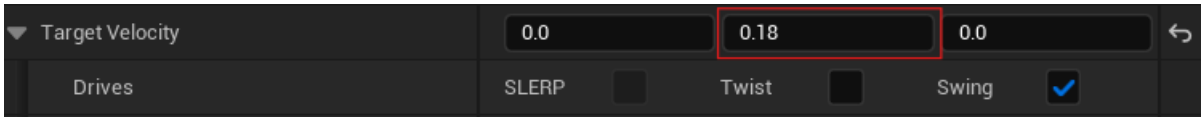


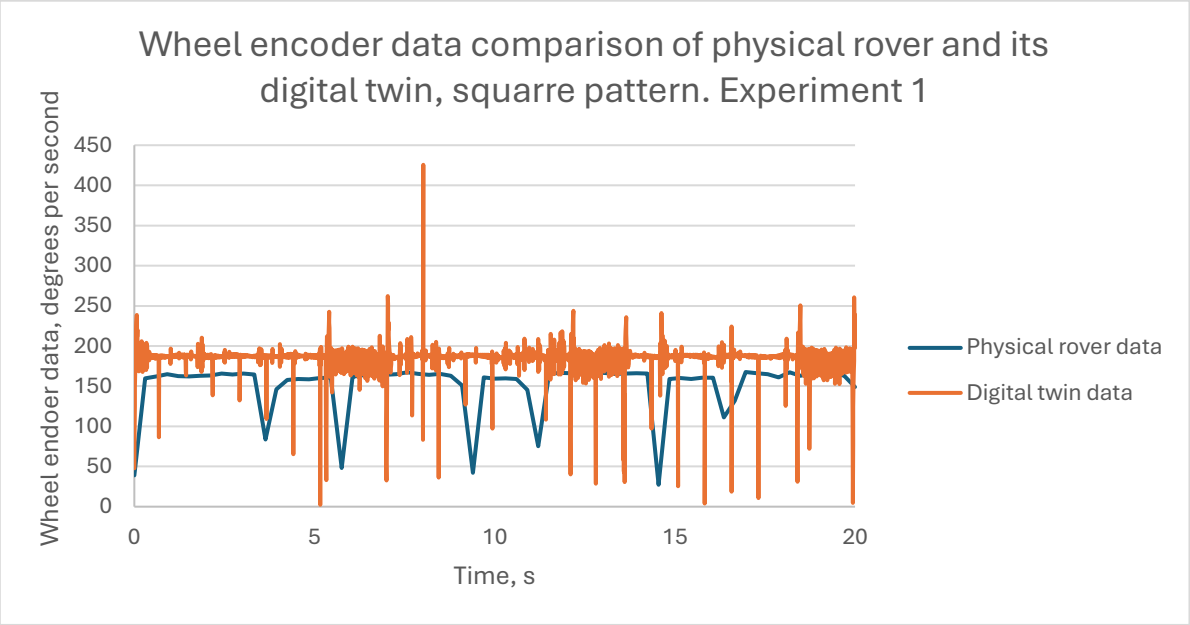
Figure 15. Physics constraints target velocity for the digital twin in Unreal Engine

5 Experimental result comparison between physical rover and the digital twin

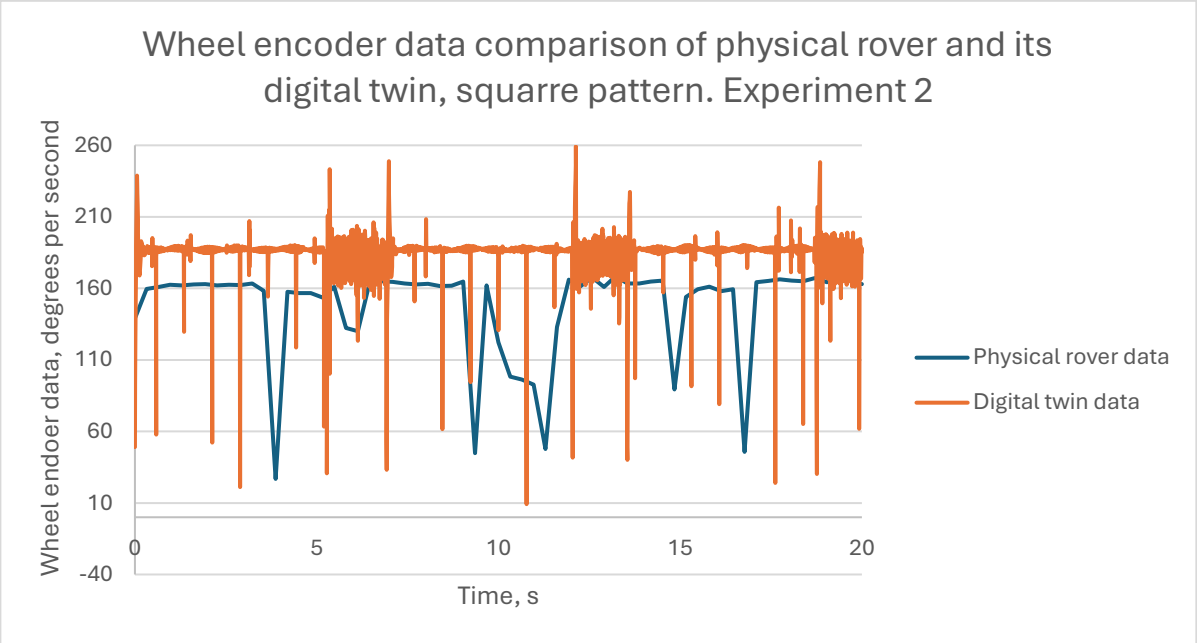
This section will cover the experimental results between the physical rover and its digital twin in Unreal Engine 5 from a set of experiments design to test the rover similarities.

5.1 Wheel encoder experiments

To test the similarities between the physical rover and its digital twin two sets of experiments were conducted. First one consisting of driving the robots in a square 1 meter by 1 meter turning 90 degrees. For the second experiment rovers were driven in a "lighting" shape which was 1 meter forward, turn 90 degrees right, 1 meter forward, turn 90 degrees left, and the last 1-meter forward drive. Both experiments were conducted multiple times. Physical rover was driven in the Tartu Observatories bunker which had specialized sand as the surface. Wheel encoder data is visible in the graph 2, graph 3.

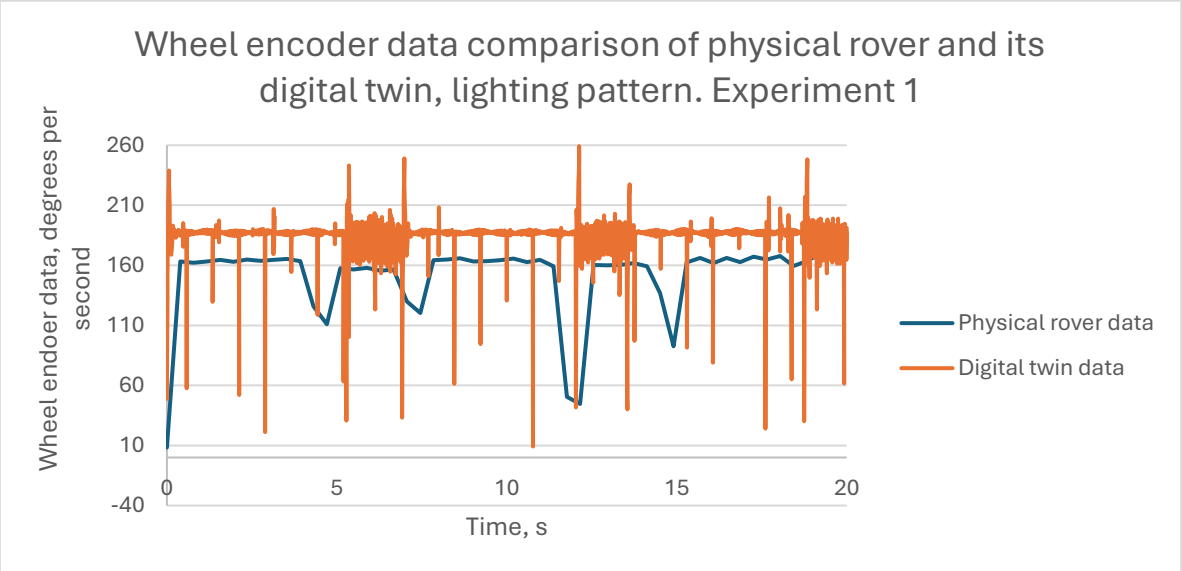


Graph 2. Wheel encoder data comparison, square pattern, experiment 1

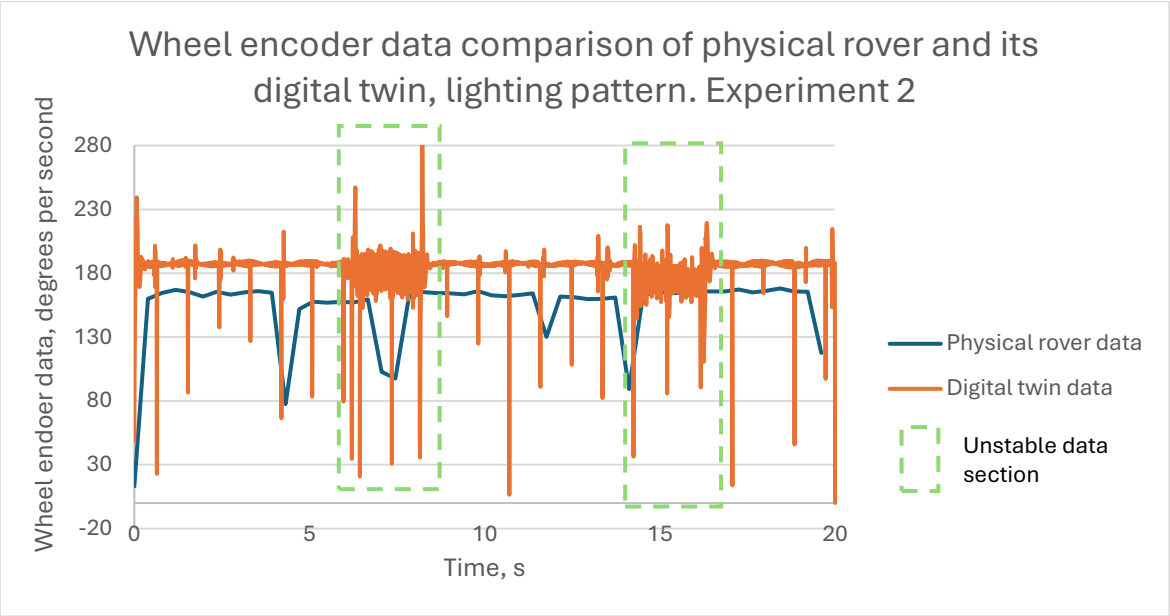


Graph 3. Wheel encoder data comparison, square pattern, experiment 2

Next set of data for the “lighting” pattern driving is visible in graph 4, graph 5.



Graph 4. Wheel encoder data comparison, lighting pattern, experiment 1



Graph 5. Wheel encoder data comparison, lighting pattern, experiment 2

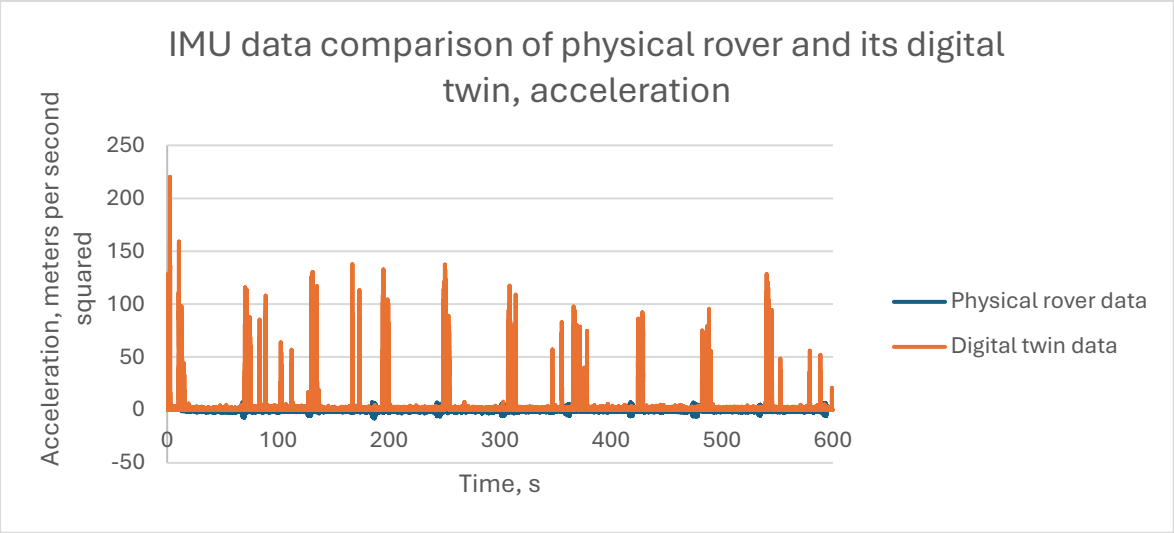
Data in these graphs show a lot of details about how the physical rover and its digital twin operate. Firstly, the data has a significant difference in the amount of data points, physical rover had approximately 50 data points while the digital twin had about 3000 data points. Both of the model data is not perfectly stable and the reasons for this can be many. Focusing on the physical rover data discrepancies could have been caused by slippage or getting stuck in sand. However, the digital twin did not experience such problems as the surface in the simulations is perfect, but the spikes in the data are either problems with the alignment of the components or computational efficiency of the used system. Furthermore, there are noticeable sections in the graphs similar to the one marked in graph X as "Unstable data section" which are caused by two main reasons. First reason is how the rover turns, which happens by spinning the wheels in opposite directions and thus turning either left or right, but such actions cause the rover to shake in simulations and due to that the data becomes more unstable. Other reason is due to minor misalignment between the components which causes small collisions, this can be slightly lessened by enabling a setting called "Async physics tick enabled" which runs the physics simulation on a separate thread and thus improving the stability of the simulations however as this option is experimental it does not solve all the problems.

Most importantly a very interesting observation is made by examining all of the data. Physical rovers' data and its digital twins on average seems similar to one another with one distinct difference. Degrees per second differ by approximately 20 points which has been caused by the fact that the digital twin was calibrated based on data when the physical rover drove over solid ground, but the experiments were conducted on special sand. While driving over the sand rovers' wheel slightly sunk and thus caused a higher resistance for the motors which did not have the power to sustain the regular rotational speed of wheels. Unfortunately, creating such an effect in simulations is not possible because one would need to create a multi-level floor with different forces working against the wheels and this would be very computationally difficult as well as Unreal Engine has not been designed to work is such a way. But thinking about the application and the mission plan of the physical rover such a problem would not be

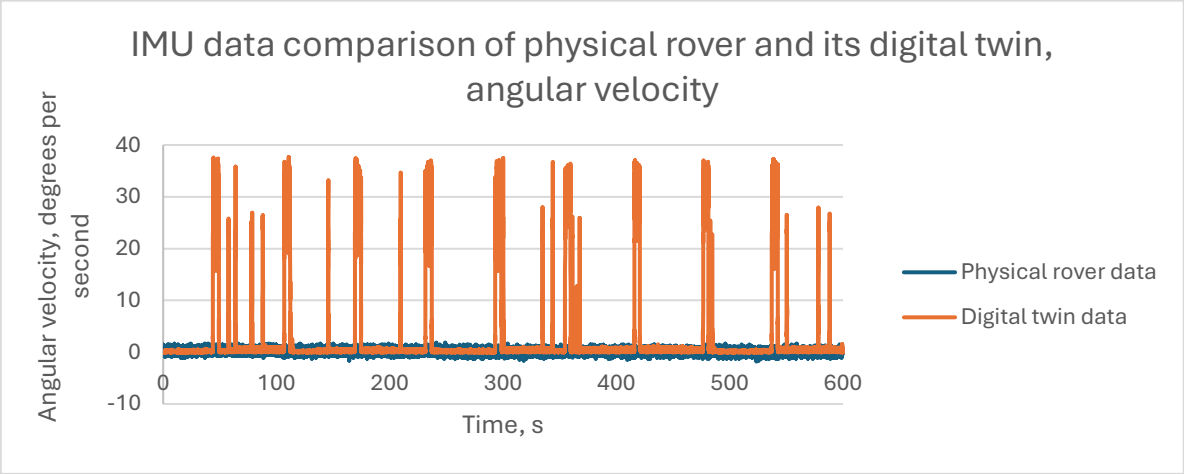
present on the Moon due to its reduced gravity which would cause the rover not to sink in the sand so much.

5.2 IMU experiment

To observe the difference between the physical rover and the digital twin in terms of IMU data an experiment was conducted, where both rovers had to drive back and forth for a distance of 10 meters five times, in total covering the distance of 50 meters on a solid, non-slippery surface. This was meant to show how the data differs between in a long-term experiment as the overall driving time was approximately 10 minutes and collecting between 45000-60000 data points. Data is visible in graph 6, graph 7.



Graph 6. IMU data comparison between the physical rover and the digital twin of acceleration



Graph 7. IMU data comparison between the physical rover and the digital twin of angular velocities

Looking at both of the graphs certain things can be noticed. Digital twins' data is a lot more unstable having high peaks at certain moments in the experiment. This was most likely caused by previously discussed misalignments and overall instability of the system as well as computational load. Another factor for this could have been the fact that Unreal Engine takes into account the frequency of the monitor the images are displayed on, but by running a longer experiment more data is accumulated in the system which could cause the a heavier computational load and as the system tries to maintain a stable frequency of image outputs it lacks the necessary computational power and starts to become unstable. This effect most likely happened during the turning of the rover because as the rover is turning the background imagery must be rendered causing a heavier load for the computer.

5.3 Lunar gravity experiment

This experiment was conducted to see how the reduced gravity would affect the rovers' driving capabilities. It was achieved by changing the world settings in Unreal Engine and setting the value of gravity to -162 cm/s^2 . Most noticeable thing was the fact that one of the front wheels would lift up in the air when starting to accelerate visible in figure 16. This could possibly be solved either by reducing the rovers speed or increasing the spring strength for the frontal suspension.

Figure 16. Digital twin's front wheels lift from ground when accelerating in a lunar environment

5.4 Discussion of the results

Looking over at the results certain aspects become apparent. Firstly, implementing real world physical rover characteristics into a digital twin is a very time consuming and difficult process as even the smallest misalignment can cause severe problems with the stability of the simulation. However, Unreal Engine provides good tools to deal with most issues to a certain degree.

Although dynamometer experiments were successful the data implementation in the digital twin was unfortunately not possible due to the very limited data on how Unreal Engines Chaos engine functions especially how it uses the physics constraints "Strength" and how to connect this to the physical rover. Numerous experiments and setups were conducted to understand this functionality, but no reasonable conclusions were made. On the other hand, rest of the properties were implemented into the digital twin such as the speed of the rover, its overall build and the mass of it.

Focusing on the wheel encoder data it is possible to assume that this experiment was successful and helped to better understand the differences between both models and the possible alternations that would have to be made for future experiments, especially in terms of motor strength. Another aspect to increase fidelity would be to limit the rate at which Unreal Engine measures, thus when performing the analysis data would be more similar. In the future adding random jitters or noise would also be beneficial.

Going over to the IMU experiments. Unreal Engines synthetic data is very unstable for both gyroscopic measurements as well as acceleration, even though filtering could be applied to better see what the majority of the data looks like, understanding and solving the stability issues is crucial to achieve a stable data output from which conclusions could be drawn. Furthermore, because Unreal Engine provides raw data, but the physical sensors usually have a filter inside is something to think about in the future as well. After achieving stability for the digital twin noise would also have to be introduced to better simulate real world conditions.

Lastly, conducting a simple driving experiment in the Lunar environment yielded results by showing two possible problems: either the speed of the rover was too high, or the spring strength would need to be increasing to ensure the front wheels of the digital twin do not lift of from the ground when accelerating. As it is very difficult to replicate Lunar gravity on Earth such experiments can give insights on the possible mission outcomes however this data can't be tested until the rover traverses the Moon.

6 Conclusion

In conclusion, the initial research done on the space industry and current simulation tools used as well as possible alternatives brought inside on the tools that could be used in practical applications one such being Unreal Engine 5. As it showed balance both from the computational requirement standpoint as well as its capabilities.

Implementing an accurate digital twin model requires significant testing and data to achieve as close as possible digital twin to the physical rover. This was successfully achieved by implementing most of the data and characteristics from the real rover, however limitations of information accessibility and overall system prevented from fully incorporating the available data. Furthermore, due to the novelty of this simulation software difficult problems have yet to be discovered and answered in the forums which required a significant investment of time in testing and prototyping which often yielded very little fruits. Achieving a truly multipurpose simulation software is a very difficult task and thus mostly the industry standard is to use very specialized programs build for specific systems and their requirements.

Overall, examining the results obtained from wheel encoder and IMU experiments it is possible to conclude that the set-out tasks were completed and provided valuable insights on the future problems and improvements that could be made, as this works main purpose which was to create an initial model of the physical rover in Unreal Engine 5. Which could be later used as a base for improvements and prototyping. Thus, the final experiment was conducted to see how the rover would act in Lunar gravity which provided valuable insight and concluded this work.

Possible future paths for this digital twin would require a thorough understanding of how to improve the stability of the whole system and by achieving that it would be ready to be used as a prototyping base to test out new systems and instruments.

Acknowledgements

I would like to thank everyone who has helped me throughout this journey. Special thanks to Davis Krumins for the support, Andres Aleksander Tammer for helping me with running the experiments, Karl Kruusamäe for being a friendly yet a just professor and Heiki Kasemägi for displaying support in times of need and always having answers to tough questions.

And of course, everyone else, friends, family, who supported and helped me through these years.

Bibliography

[1] 'Why Go to Space - NASA'. Accessed: May 07, 2024. [Online]. Available: https://www.nasa.gov/humans-in-space/why-go-to-space/

[2] 'Every mission to Mars ever', The Planetary Society. Accessed: May 08, 2024. [Online]. Available: https://www.planetary.org/space-missions/every-mars-mission

[3] W. Wu, W. Liu, D. Qiao, and D. Jie, 'Investigation on the development of deep space exploration', *Sci. China Technol. Sci.*, vol. 55, Apr. 2012, doi: 10.1007/s11431-012-4759-z.

[4] 'Twin of NASA's Perseverance Mars Rover Begins Terrain Tests - NASA'. Accessed: May 08, 2024. [Online]. Available: https://www.nasa.gov/centers-and-facilities/jpl/twin-of-nasas-perseverance-mars-rover-begins-terrain-tests/

[5] D. Gaines *et al.*, 'ONBOARD PLANNING FOR THE MARS 2020 PERSEVERANCE ROVER'.

[6] 'Simulation Tools - NASA'. Accessed: May 08, 2024. [Online]. Available: https://www.nasa.gov/general/simulation-tools/

[7] ISECG, 'Global Exploration Roadmap'. ISECG, Aug. 2024. [Online]. Available: https://www.globalspaceexploration.org/wp-content/isecg/GER2024.pdf

[8] Dr. David R. William, 'Lunar Exploration Timeline', *NASA*, [Online]. Available: https://nssdc.gsfc.nasa.gov/planetary/lunar/lunartimeline.html

[9] Patrick Labouré, 'Feasibility Study on the development of Lunar Environment Simulator', Jul. 2021, [Online]. Available: https://nebula.esa.int/sites/default/files/neb_study/2524/C4000130476ES.pdf

[10] O. Bekdash *et al.*, *Development and Evaluation of the Active Response Gravity Offload System as a Lunar and Martian EVA Simulation Environment*. 2020.

[11] Epic Games, 'Unreal Engine 5', May 20, 2025. [Online]. Available: https://www.unrealengine.com/en-US/unreal-engine-5

[12] NASA eyes, 'Mars Perseverance pre-landing simulation', 2020. [Online]. Available: https://eyes.nasa.gov/apps/mars2020/#/home?id=entry_interface_point

[13] E. Gent, 'Computing Power Can Keep Growing as Moore's Law Winds Down. Here's How', *singularityhub*, Jun. 2020, [Online]. Available: https://singularityhub.com/2020/06/08/computing-power-can-keep-growing-as-moores-law-winds-down-heres-how/

[14] Q. Saimoon, 'Kuupkulgur - the estonian lunar rover concept', May 20, 2025. [Online]. Available: https://tospexgroup.space/projects/kuupkulgur/

[15] MathWorks, 'Digital Twins for Predictive Maintenance', *MathWorks*. [Online]. Available: https://se.mathworks.com/campaigns/offers/next/digital-twins-for-predictive-maintenance.html

[16] R. R. Corsini, A. Costa, S. Fichera, and J. M. Framinan, 'Digital twin model with machine learning and optimization for resilient production–distribution systems under disruptions', *Comput. Ind. Eng.*, vol. 191, p. 110145, May 2024, doi: 10.1016/j.cie.2024.110145.

[17] Graham Kenny, 'Digital Twins Can Help You Make Better Strategic Decisions', Sep. 23, 2024. [Online]. Available: https://hbr.org/2024/09/digital-twins-can-help-you-make-better-strategic-decisions?utm_medium=paidsearch&utm_source=google&utm_campaign=intlcontent_strategy&utm_term=Non-Brand&tpcc=intlcontent_strategy&gad_source=1&gclid=CjwKCAiAt4C-

BhBcEiwA8Kp0CYJWvDxfzjvkjWi1tmhPFhdBTtLxtwt0CmQX-F6w2_voNmCsfTFqSRoCc2AQAvD_BwE
[18] M. Attaran and B. G. Celik, 'Digital Twin: Benefits, use cases, challenges, and opportunities', *Decis. Anal. J.*, vol. 6, p. 100165, Mar. 2023, doi: 10.1016/j.dajour.2023.100165.
[19] Ankitha VP, 'Digital Twin Development Time', Jan. 15, 2025. [Online]. Available: https://www.toobler.com/blog/digital-twin-development-time
[20] Unreal University, 'How to Learn Unreal Engine In 2025 (Beginners Guide)', Dec. 28, 2024. [Online]. Available: https://www.unreal-university.blog/how-to-learn-unreal-engine-in-2025-beginners-guide-2/
[21] Space Insider, 'Countries with Space Programs: An Overview', Nov. 2023, [Online]. Available: https://spaceinsider.tech/2023/11/27/countries-with-space-programs-an-overview/
[22] Harry Jpnes, 'Take Material to Space or Make It There?' Ascend, Oct. 23, 2023. [Online]. Available: https://ntrs.nasa.gov/api/citations/20230015110/downloads/Take%20or%20Make%20ASCEND%20charts.pdf
[23] W. Liu, M. Wu, G. Wan, and M. Xu, 'Digital Twin of Space Environment: Development, Challenges, Applications, and Future Outlook', *Remote Sens.*, vol. 16, no. 16, 2024, doi: 10.3390/rs16163023.
[24] C. Floyd, 'A Systematic Look at Prototyping', in *Approaches to Prototyping*, R. Budde, K. Kuhlenkamp, L. Mathiassen, and H. Züllighoven, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 1984, pp. 1–18.
[25] ETD's Mechanical Systems Division, 'Thermal Engineering', Jan. 2025, [Online]. Available: https://etd.gsfc.nasa.gov/capabilities/capabilities-listing/thermal-engineering/
[26] Jon Homan, 'Thermal Vacuum Chamber A', Jan. 06, 2025. [Online]. Available: https://www.nasa.gov/setmo/facilities/thermal-vacuum-chamber-a/
[27] Marty McCoy, 'James Webb Space Telescope', *NASA*, May 15, 2025. [Online]. Available: https://science.nasa.gov/mission/webb/
[28] 'Observing galaxies: Explore the software behind the James Webb Space Telescope', *MayaHTT*, 2025. [Online]. Available: https://www.mayahtt.com/blog/explore-software-james-webb-space-telescope/
[29] Stephen Sabia, 'Infrared Detectors', *NASA*, Aug. 12, 2024. [Online]. Available: https://science.nasa.gov/mission/webb/infrared-detectors/
[30] Kan Yang, 'Thermal Model Performance for the James Webb Space Telescope OTIS Cryo-Vacuum Test'. Jul. 12, 2018. [Online]. Available: https://ntrs.nasa.gov/api/citations/20180004588/downloads/20180004588.pdf
[31] 'Ansys Zemax OpticStudio Comprehensive Optical Design Software', *Ansys*, May 04, 2025. [Online]. Available: https://www.ansys.com/products/optics/ansys-zemax-opticstudio#tab1-2
[32] J. Howard and R. Ohl, 'Optical alignment of James Webb Space Telescope breaks new ground', *SPIE Newsroom*, Apr. 2011, doi: 10.1117/2.1201103.003570.
[33] ANSYS BLOG, 'Designing the James Webb Space Telescope with Simulation', Dec. 25, 2021. [Online]. Available: https://www.ansys.com/blog/designing-the-james-webb-space-telescope-with-simulation
[34] Stephen Carney, 'Ingenuity Mars Helicopter', Feb. 14, 2025. [Online]. Available: https://science.nasa.gov/mission/mars-2020-perseverance/ingenuity-mars-helicopter/

[35] T. Tzanetos *et al.*, 'Ingenuity Mars Helicopter: From Technology Demonstration to Extraterrestrial Scout', in *2022 IEEE Aerospace Conference (AERO)*, 2022, pp. 01–19. doi: 10.1109/AERO53065.2022.9843428.
[36] Pilot Institute, 'Ingenuity and the Challenges of Flying A Helicopter on Mars', Aug. 29, 2021. [Online]. Available: https://pilotinstitute.com/ingenuity-drone/
[37] Håvard Fjær Grip, 'FLIGHT DYNAMICS OF A MARSHELICOPTER*'. 2017. [Online]. Available: https://rotorcraft.arc.nasa.gov/Publications/files/ERF2017_final.pdf
[38] A. Jain, 'DARTS - Multibody Modeling, Simulation and Analysis Software', in *Multibody Dynamics 2019*, A. Kecskeméthy and F. Geu Flores, Eds., Cham: Springer International Publishing, 2020, pp. 433–441.
[39] Cuyler Dull, 'Hover and Forward Flight Performance Modeling of the Ingenuity Mars Helicopter'. Jan. 10, 2022. [Online]. Available: https://rotorcraft.arc.nasa.gov/Publications/files/Cuyler_Dull_10-Jan-22_06-00-24.pdf
[40] IBM, 'What is a digital twin?', Aug. 05, 2021. [Online]. Available: https://www.ibm.com/think/topics/what-is-a-digital-twin
[41] Wayne Johnson, 'Mars Science Helicopter Conceptual Design'. Apr. 22, 2024. [Online]. Available: https://core.ac.uk/download/pdf/323104186.pdf
[42] K. Zhu *et al.*, 'Conceptual design and aerodynamic analysis of a Mars octocopter for sample collection', *Acta Astronaut.*, vol. 207, pp. 10–23, Jun. 2023, doi: 10.1016/j.actaastro.2023.02.033.
[43] Lonnie Shekhtman, 'OSIRIS-REx', Jan. 30, 2025. [Online]. Available: https://science.nasa.gov/mission/osiris-rex/
[44] Thomas Ajluni, 'OSIRIS-REx, Returning the Asteroid Sample'. Aug. 09, 2024. [Online]. Available: https://ntrs.nasa.gov/api/citations/20150000809/downloads/20150000809.pdf
[45] C. D. Norman, 'Autonomous Navigation Performance Using Natural Feature Tracking during the OSIRIS-REx Touch-and-Go Sample Collection Event'. The Planetary Science Journal, May 10, 2025. [Online]. Available: https://iopscience.iop.org/article/10.3847/PSJ/ac5183
[46] Jeff Foust, 'NASA to repurpose OSIRIS-REx for second asteroid encounter', Apr. 26, 2025. [Online]. Available: https://spacenews.com/nasa-to-repurpose-osiris-rex-for-second-asteroid-encounter/
[47] Brian Sutter, 'OSIRIS-REx Extended Mission Trajectory Design & Target Search'. NASA, May 02, 2021. [Online]. Available: https://ntrs.nasa.gov/api/citations/20210025297/downloads/OSIRIS_REx_Extended_Mission_Trajectory_Design_Target_Search_final.pdf
[48] 'About Gazebo', Jan. 10, 2025. [Online]. Available: https://gazebosim.org/about
[49] L. Santana, M. Maximo, L. Goes, and I. da Fonseca, *Development of a High Fidelity Space Robot Gazebo-Based Simulator for Space Close-Proximity Operations Experiments*. 2021. doi: 10.26678/ABCM.COBEM2021.COB2021-0620.
[50] J. L. Ramón, 'A ROS/Gazebo-based framework for simulation and control of on-orbit robotic systems'. Sep. 22, 2022. [Online]. Available: https://dspace.lib.cranfield.ac.uk/server/api/core/bitstreams/fb2059e5-068b-4b34-ba3d-80c5b0079782/content
[51] M.A. Diftler, 'Robonaut 2 – The First Humanoid Robot in Space'. NASA, Apr. 09, 2010. [Online]. Available: https://ntrs.nasa.gov/api/citations/20100040493/downloads/20100040493.pdf

[52] scpeters@openrobotics.org, ‘Gazebo Development Libraries physics’, Feb. 12, 2025. [Online]. Available: https://gazebosim.org/libs/physics/
[53] MATLAB, ‘Gazebo Simulation Environment Requirements and Limitations’, 2025. [Online]. Available: https://se.mathworks.com/help/robotics/ug/gazebo-simulation-requirements.html
[54] D. Krumins *et al.*, ‘Open Remote Web Lab for Learning Robotics and ROS With Physical and Simulated Robots in an Authentic Developer Environment’, *IEEE Trans. Learn. Technol.*, vol. 17, pp. 1–14, Jan. 2024, doi: 10.1109/TLT.2024.3381858.
[55] NVIDIA, ‘NVIDIA Omniverse’, 2025. [Online]. Available: https://www.nvidia.com/en-eu/omniverse/
[56] NVIDIA, ‘Omniverse Developer Guide: Get Started’, May 17, 2025. [Online]. Available: https://docs.omniverse.nvidia.com/dev-guide/latest/get-started.html
[57] Adam Moravanszky, ‘Open Source Simulation Expands with NVIDIA PhysX 5 Release’, Nov. 08, 2022. [Online]. Available: https://developer.nvidia.com/blog/open-source-simulation-expands-with-nvidia-physx-5-release/
[58] NVIDIA, ‘Build Beautiful, Custom UI for 3D Tools on NVIDIA Omniverse’, Nov. 08, 2025. [Online]. Available: https://learn.nvidia.com/courses/course-detail?course_id=course-v1:DLI+S-OV-08+V1
[59] PROX, ‘System Hardware Requirements for NVIDIA Omniverse in 2025’, Jan. 30, 2025. [Online]. Available: https://www.proxpc.com/blogs/system-hardware-requirements-for-nvidia-omniverse-in-2025
[60] HGInsights, ‘Companies Currently Using NVIDIA Omniverse’, May 20, 2025. [Online]. Available: https://discovery.hgdata.com/product/nvidia-omniverse
[61] X. Li *et al.*, *Exploring NVIDIA Omniverse for Future Space Resources Missions*. 2022. doi: 10.13140/RG.2.2.27498.12481.
[62] K. Thor Jensen, ‘25 Years Later: The History of Unreal and an Epic Dynasty’, May 22, 2023. [Online]. Available: https://www.pcmag.com/news/25-years-later-the-history-of-unreal-and-an-epic-dynasty
[63] A. Ciekanowska, A. Gliński, and K. Dziedzic, ‘Comparative analysis of Unity and Unreal Engine efficiency in creating virtual exhibitions of 3D scanned models’, *J. Comput. Sci. Inst.*, vol. 20, pp. 247–253, Sep. 2021, doi: 10.35784/jcsi.2698.
[64] Epic Games, ‘Nanite Virtualized Geometry’, Apr. 01, 2025. [Online]. Available: https://dev.epicgames.com/documentation/en-us/unreal-engine/nanite-virtualized-geometry-in-unreal-engine
[65] Epic Games, ‘Lumen Global Illumination and Reflections’, Apr. 01, 2025. [Online]. Available: https://dev.epicgames.com/documentation/en-us/unreal-engine/lumen-global-illumination-and-reflections-in-unreal-engine
[66] Epic Games, ‘Hardware and Software Specifications’, May 20, 2025. [Online]. Available: https://dev.epicgames.com/documentation/en-us/unreal-engine/hardware-and-software-specifications-for-unreal-engine
[67] Epic Games, ‘Unreal Engine - physics’, May 20, 2025. [Online]. Available: https://dev.epicgames.com/documentation/en-us/unreal-engine/physics?application_version=4.27
[68] Epic Games, ‘Chaos physics in Unreal Engine’, May 20, 2025. [Online]. Available: https://dev.epicgames.com/documentation/en-us/unreal-engine/physics-in-unreal-engine
[69] Oren, ‘Which Companies Use Unreal Engine 5?’, Feb. 05, 2025. [Online]. Available: https://achivx.com/which-companies-use-unreal-engine-5/?utm_source=chatgpt.com

[70] Enlyft, ‘Companies using Unreal Engine’, May 20, 2025. [Online]. Available: https://enlyft.com/tech/products/unreal-engine
[71] Animost Team, ‘Is Unreal Engine 5 Easy? How Long Does It Take To Learn UE5?’, Dec. 14, 2023. [Online]. Available: https://animost.com/ideas-inspirations/is-unreal-engine-5-easy/
[72] Christian König, ‘Implementation Analysis of Collaborative Robot Digital Twins in Physics Engines’. Arxiv, May 28, 2025. [Online]. Available: https://arxiv.org/html/2504.18200v2
[73] Brian Notosubagyo, ‘Unreal Engine Testbed for Computer Vision of Tall Lunar Tower Assembly’, NASA, Jan. 2023. [Online]. Available: https://ntrs.nasa.gov/api/citations/20230013170/downloads/UE_Testbed_for_CV_of_TLT_Assembly_FInal.pdf
[74] C. Ellul, N. Hamilton, A. Pieri, and G. Floros, ‘Exploring Data for Construction Digital Twins: Building Health and Safety and Progress Monitoring Twins Using the Unreal Gaming Engine’, *Buildings*, vol. 14, p. 2216, Jul. 2024, doi: 10.3390/buildings14072216.
[75] I. Rasmussen, S. Kvalsvik, P.-A. Andersen, T. Aune, and D. Hagen, ‘Development of a Novel Object Detection System Based on Synthetic Data Generated from Unreal Game Engine’, *Appl. Sci.*, vol. 12, Aug. 2022, doi: 10.3390/app12178534.
[76] Keef Sloan, ‘How NASA Trains Astronauts with Unreal Engine’, *NASA*, Mar. 11, 2016. [Online]. Available: https://www.unrealengine.com/fr/developer-interviews/how-nasa-trains-astronauts-with-unreal-engine
[77] a16z speedrun, ‘How NASA’s Using Unreal Engine 5 to Prep for its Next Moonshot’, Aug. 07, 2024. [Online]. Available: https://speedrun.substack.com/p/nasa-unreal-engine-5
[78] CAE, ‘First full flight simulator with gaming-engine-powered CAE Prodigy image generator achieves level D qualification’. CAE, Mar. 26, 2024. [Online]. Available: https://www.cae.com/media-centre/press-releases/first-full-flight-simulator-with-gaming-engine-powered-cae-prodigy-image-generator-achieves-level-d-qualification/
[79] R. Geronel and M. Silva, ‘Digital twins in robotic applications’, 2025, pp. 123–142.
[80] S. Vaidya, P. Ambad, and S. Bhosle, ‘Industry 4.0 – A Glimpse’, *2nd Int. Conf. Mater. Manuf. Des. Eng. ICMMD2017 11-12 Dec. 2017 MIT Aurangabad Maharashtra INDIA*, vol. 20, pp. 233–238, Jan. 2018, doi: 10.1016/j.promfg.2018.02.034.
[81] SolidWorks, ‘The Solution for 3D CAD, Design and Product Development’, May 20, 2025. [Online]. Available: https://www.solidworks.com/
[82] Alec Chillingworth, ‘The pros & cons of creating 3D content with Blender software’, *EpidemicSounds*, Mar. 30, 2023. [Online]. Available: https://www.epidemicsound.com/blog/blender-software/
[83] Blender, ‘Decimate Modifier - Blender’, May 10, 2025. [Online]. Available: https://docs.blender.org/manual/en/latest/modeling/modifiers/generate/decimate.html#decimate-modifier
[84] Blender, ‘Clean Up - Blender’, May 10, 2025. [Online]. Available: https://docs.blender.org/manual/en/latest/modeling/meshes/editing/mesh/cleanup.html
[85] Blender, ‘Limited Dissolve - Blender’, May 10, 2025. [Online]. Available: https://docs.blender.org/manual/en/latest/modeling/meshes/editing/mesh/delete.html#bpy-ops-mesh-dissolve-limited
[86] T. Anteneh, *Realistic Simulation and Kalman Filter-Based Estimation of Wheel Encoder Signals in Noisy Environments*. 2024. doi: 10.13140/RG.2.2.24864.80644.

[87] I. Faisal, T. Purboyo, and A. Ansori, ‘A Review of Accelerometer Sensor and Gyroscope Sensor in IMU Sensors on Motion Capture’, *J. Eng. Appl. Sci.*, vol. 15, pp. 826–829, Nov. 2019, doi: 10.36478/jeasci.2020.826.829.

Appendices

Experimental results and models are available at: https://drive.google.com/drive/folders/1TqDH8FT92gZnmiNtmyYIIShYIl8oLAcH?usp=sharing

Non-exclusive licence to reproduce thesis and make thesis public

I, Audris Mihailovs

1. herewith grant the University of Tartu a free permit (non-exclusive licence) to reproduce,for the purpose of preservation, including for adding to the DSpace digital archives until the expiry of the term of copyright,

   "Development and Validation of a Lunar Rover Digital Twin Using Unreal Engine 5"

   supervised by self-supervised

2. I grant the University of Tartu a permit to make the work specified in p. 1 available tothe public via the web environment of the University of Tartu, including via the DSpace digital archives, under the Creative Commons licence CC BY NC ND 3.0, which allows, by giving appropriate credit to the author, to reproduce, distribute the work and communicate it to the public, and prohibits the creation of derivative works and any commercial use of the work until the expiry of the term of copyright.

3. I am aware of the fact that the author retains the rights specified in p. 1 and 2.

4. I certify that granting the non-exclusive licence does not infringe other persons' intellectual property rights or rights arising from the personal data protection legislation.

Audris Mihailovs
20.05.2025