EXPLORING THE SOLAR SYSTEM USING DEEP REINFORCEMENT LEARNING VIA ML AGENTS IN UNITY

A PROJECT REPORT
Submitted By

KRISHNA KANT PANDEY[RA2011026010288]
A.BHAVANI SHANKAR [RA2011026010300]

Under the Guidance of
Dr.G.VADIVU
Professor, Department of Data Science and Business Systems

in partial fulfillment of the requirements for the degree of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE AND ENGINEERING

with specialization in Artificial Intelligence and Machine Learning

DEPARTMENT OF COMPUTATIONAL INTELLIGENCE

COLLEGE OF ENGINEERING AND TECHNOLOGY

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

KATTANKULATHUR– 603 203

MAY 2024

Department of Computational Intelligence
SRM Institute of Science and Technology
Own Work Declaration Form

Degree/ Course	: B.Tech Computer Science and Engineering with specialization in Artificial Intelligence and Machine Learning
Student Name	: Krishna Kant Pandey, A.Bhavani Shankar
Registration Number	: RA2011026010288, RA2011026010300
Title of Work	: **Exploring The Solar System using Deep Reinforcement Learning via ML Agents in Unity**

We hereby certify that this assessment compiles with the University's Rules and Regulations relating to Academic misconduct and plagiarism, as listed in the University Website, Regulations, and the Education Committee guidelines.

We confirm that all the work contained in this assessment is our own except where indicated, and that We have met the following conditions:

- Clearly referenced / listed all sources as appropriate
- Referenced and put in inverted commas all quoted text (from books, web, etc.)
- Given the sources of all pictures, data etc. that are not my own
- Not made any use of the reports or essays of any other students either past or present
- Acknowledged in appropriate places any help that We have received from others (e g.fellow students, technicians, statisticians, external sources)
- Compiled with any other plagiarism criteria specified in the Course handbook / University website

We understand that any false claim for this work will be penalized in accordance with the University policies and regulations.

DECLARATION:

I am aware of and understand the University's policy on Academic misconduct and plagiarism and I certify that this assessment is our own work, except where indicated by referring, and that I have followed the good academic practices noted above.

Student 1 Signature: [illegible] 30/4/24

Student 2 Signature: [illegible] 30/04/24

SRM
INSTITUTE OF SCIENCE & TECHNOLOGY

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY
KATTANKULATHUR - 603203

BONAFIDE CERTIFICATE

Certified that 18CSP109L project report titled **"Exploring The Solar System using Deep Reinforcement Learning via ML Agents in Unity"** is the bonafide work of **KRISHNA KANT PANDEY [RA2011026010288] and A.BHAVANI SHANKAR [RA2011026010300]** who carried out the project work under my supervision. Certified further, that to the best of my knowledge the work reported here does not form part of any other thesis or dissertation on the basis of which a degree or award was conferred on an earlier occasion for this or any other candidate.

[illegible] 25/4/24
SIGNATURE

Dr. G.VADIVU
SUPERVISOR
Professor
Department of Data Science and Business Systems

[illegible] 30/04/24
SIGNATURE

Dr. R. ANNIE UTHRA
HEAD OF DEPARTMENT
Department of Computational Intelligence

[illegible] 15/5/24
EXAMINER I

[illegible] 15/05/2024.
EXAMINER II

ACKNOWLEDGEMENT

We express our humble gratitude to **Dr. C. Muthamizhchelvan**, Vice-Chancellor, SRM Institute of Science and Technology, for the facilities extended for the project work and his continued support.

We extend our sincere thanks to Dean-CET, SRM Institute of Science and Technology, **Dr. T. V. Gopal**, for his invaluable support.

We wish to thank **Dr. Revathi Venkataraman**, Professor & Chairperson, School of Computing, SRM Institute of Science and Technology, for her support throughout the project work.

We are incredibly grateful to our Head of the department**, Dr. R. Annie Uthra** Professor, Department of Computational Intelligence, SRM Institute of Science and Technology, for her suggestions and encouragement at all the stages of the project work.

We want to convey our thanks to our program coordinators**,** and panel head, **Dr. S. Karthik**, Associate Professor, Department of Computational Intelligence, SRM Institute of Science and Technology, for his inputs during the project reviews and support.

We register our immeasurable thanks to our Faculty Advisor**, Dr. K. Moorthi**, Assistant Professor**,** Department of Computational Intelligence, SRM Institute of Science and Technology, for leading and helping us to complete our course.

Our inexpressible respect and thanks to my guide, **Dr. G. Vadivu,** Professor, Department of Data Science and Business Systems, SRM Institute of Science and Technology, for providing me with an opportunity to pursue my project under her mentor-ship. She provided me with the freedom and support to explore the research topics of my interest. Her passion for solving problems and making a difference in the world has always been inspiring.

We sincerely thank the Computational Intelligence staff and students, SRM Institute of Science and Technology, for their help during our project. Finally, we would like to thank parents, family members, and friends for their unconditional love, constant support, and encouragement.

KRISHNA KANT PANDEY(RA2011026010288)
A.BHAVANI SHANKAR(RA2011026010300)

ABSTRACT

In the novel study "Exploring the Solar System with Machine Learning Agents via Deep Reinforcement Learning," scientists use state-of-the-art artificial intelligence in a realistic virtual world to simulate the complexity of space exploration as they set out on a historic space voyage. This project, called "Solar System Adventure," creates an autonomous spaceship agent by utilizing deep reinforcement learning (DRL) and Unity's development framework. Outfitted with advanced machine learning algorithms, this agent makes its way across an intricately drawn virtual solar system, showcasing deft navigation between planetary destinations and celestial threats.

The core of this project is the deployment of a revolutionary incentive system in which the spaceship continuously learns and modifies its direction through dynamic interactions within the virtual cosmos. This demonstrates not only how flexible AI may be in intricate settings, but also how it has the possibility to completely transform how we go about discovering the enormous cosmos. The spaceship is put through a rigorous testing process in the simulated environment, demonstrating its decision-making speed and flexibility in response to changing conditions, going beyond simple programming success.

This project is a major step forward in the integration of artificial intelligence with virtual reality, providing a window into immersive and interactive experiences that will go beyond the limits of educational software and traditional digital entertainment. Here, the virtual solar system transforms from a flat background into a dynamic environment full with chances for interaction, education, and discovery, all made possible by a perceptive, learning agent figuring out the nuances of space. This exploration not only establishes a new standard for educational simulations, potentially directing future space missions, but it also perfectly illustrates how AI's adaptability can be combined with the endless possibilities of space exploration.

TABLE OF CONTENTS

LIST OF FIGURES

ABBREVIATIONS

AI: Artificial Intelligence

DRL: Deep Reinforcement Learning

QC: Quantum Computing

ML: Machine Learning

EMTG: Evolutionary Mission Trajectory Generator

HPC: High-Performance Computing

DSN: Deep Space Network

TESS: Transiting Exoplanet Survey Satellite

ESA: European Space Agency

GUI: Graphical User Interface

DL: Deep Learning

CHAPTER 1

INTRODUCTION

1.1 Introduction

Within the virtual constraints of a digitally generated solar system, the "Solar System Adventure" project represents a groundbreaking initiative at the nexus of artificial intelligence (AI) and space exploration. This project is a monument to the creative use of AI in recreating the vastness and complexity of space: it uses deep reinforcement learning (DRL) and the Unity platform to create an autonomous spaceship agent. Such advanced simulation technologies are extremely important at a time when mankind is on the verge of interstellar exploration. This project aims to revolutionize how we perceive and engage with the universe, going beyond simple technology demonstration to become a full teaching tool.

The creation of a cutting-edge spacecraft, an AI agent navigating a painstakingly constructed virtual cosmos, is at the heart of this inquiry. Driven by advanced machine learning algorithms, this spacecraft sets off on an exploration and education mission, adapting its trajectory and approach as it encounters different celestial occurrences. Unlike conventional space exploration simulations, which depend on preset routes and results, this project makes use of DRL to create a learning environment in which the spaceship learns from every choice it makes and every obstacle it faces.

At the forefront of this huge effort is Deep Reinforcement Learning, which will allow the spacecraft to automatically determine its approach to celestial entities. This illustrates how machine learning can emulate and even outperform natural navigational skills, paving the way for intelligent, autonomous space travel. The following sections of this paper explore the theoretical and practical foundations of combining virtual environments and machine learning, providing a window into the minds of the pioneers of this interdisciplinary subject and outlining future directions for research.

.

1.2 Motivation

The "Solar System Adventure" project is a shining example of innovation at the intersection of artificial intelligence (AI) and space exploration. This is especially true when looking at it from the perspective of deep reinforcement learning (DRL) used in a virtual universe. This project goes beyond scholarly research; it represents a quantum leap in the direction of opening up the infinite reaches of space through virtual simulations powered by the most recent AI developments. This ambitious project has a wide range of motivations, including the advancement of cognitive computing, technological advances, educational enrichment, exploratory initiatives, and the merger of virtual and tangible realities.

The project's primary goal is to transform space education. Despite their value, traditional teaching resources frequently fail to completely engage the digital native generation. "Solar System Adventure" promises an immersive learning experience where users may interact with the complexities of space, cultivating not just knowledge but a deep-seated enthusiasm for astronomy and space travel. This is achieved by utilizing DRL within a dynamic virtual solar system. It is anticipated that this approach to learning through engagement and immersion would pique interest and motivate upcoming scientists, perhaps changing the course of space science education.

In terms of technology, the project is a monument to the strength of DRL, demonstrating its capacity to maneuver through intricate and uncertain surroundings. This breakthrough expands its usefulness beyond space travel by providing insightful information on autonomous navigation and uncertain situation decision-making, a feature that can be applied to robots, autonomous vehicles, and other fields. Thus, the project serves as a testing ground for new technologies, expanding the limits of artificial intelligence both inside and outside the field of space travel.

This project has enormous exploration potential. While logistical and economical constraints limit real space travel, artificial intelligence (AI)-enabled virtual exploration opens up new possibilities. It makes space mission simulation possible and provides an affordable platform for learning, practicing, and getting ready for the real space flight problems. This feature benefits scientific research as well as democratizing space travel, opening out the wonders of space to a wider audience.

Additionally, "Solar System Adventure" makes a substantial contribution to the field of AI and cognitive computing. The study pushes the boundaries of DRL algorithms by tackling the difficulties of traversing a digitally recreated solar system. It also adds valuable insights into adaptive learning and decision-making to the field of artificial intelligence. This development is crucial because it might have a significant impact on a wide range of AI applications in many industries and usher in a new era of intelligent, self-governing systems.

In the end, this project embodies the merging of the virtual and the physical, with the goal of bridging the gap between the virtual and the vast, uncharted space. It embodies the collaboration between astronomy, computer science, and education, with the goal of developing a teaching tool that is accessible to all. The idea behind "Solar System Adventure" is that everyone should have access to the mysteries and wonders of the universe, not just the wealthy and well-off who can literally travel into space. With this endeavor, the project imagines a time where everyone is a cosmologist exploring the universe from the comfort of their own homes, motivated by an insatiable curiosity and directed by the combined powers of human intellect and artificial intelligence.

1.3 Challenges and Limitations in the Existing System

Despite their advances, the space exploration simulation systems currently in use are unable to accurately capture the complexity and unpredictable nature of the universe due to a number of problems. These difficulties vary from the pedagogical limitations in providing engaging and productive learning environments to the technical limitations of the simulation technologies available today.

A primary technological obstacle is ensuring that the simulations are realistic and accurate. The vastness of space, with its plethora of celestial entities, complex gravitational forces, and dynamic events, calls for very advanced modeling approaches and enormous computational resources. Current systems frequently oversimplify these intricacies, leading to simulations that may fail to capture the subtle dynamics of actual celestial physics. This simplification may reduce the simulation's quality and result in less accurate or significant instructional and exploratory findings.

Furthermore, the use of deep reinforcement learning (DRL) and other forms of artificial intelligence (AI) in space simulations is still very new. There is enormous potential for autonomous decision-making and flexibility in space navigation using DRL. Nevertheless, there are obstacles to DRL application in the form of data accessibility and developing reward systems that truly represent space exploration goals. Although useful, the model's reliance on simulated data raises concerns about its applicability to real-world circumstances, where unknown conditions and unpredictable factors abound.

From a pedagogical standpoint, current technologies frequently fail to produce genuinely immersive learning environments. While simulations can offer an interactive and visual depiction of space, more is needed to engage students in a way that fosters curiosity, critical thinking, and a thorough understanding of astrophysics than just visual accuracy. In order to guarantee that users engage with the simulation and derive relevant insights from it, it is imperative that educational frameworks be integrated in a way that is consistent with learning objectives. In order to ensure inclusivity in educational outreach, this entails designing experiences that are accessible to users with diverse degrees of prior knowledge.

From an accessibility perspective, the utilization of excellent space simulations is restricted to environments with the required technology infrastructure because they frequently call for powerful hardware and software. This limits the educational tools' reach and keeps out students who don't have access to them. In addition, these simulations need a large amount of human and financial resources to construct and maintain, which can be a barrier to innovation and iterative system improvement.

Finally, scalability is a problem that current systems have to deal with. Simulations need more data, more processing power, and more sophisticated algorithms as they get more comprehensive and attempt to encompass wider regions of the universe. This scaling problem is not simply a technical but also an educational one, since as simulations get more sophisticated, learners may find them more difficult to navigate and comprehend, which could overwhelm them instead of creating an environment that is favorable to learning.

In conclusion, even while systems for modeling space exploration now in use have advanced significantly, they still have many obstacles and restrictions. These range from the pedagogical, such as the requirement for more immersive and accessible learning environments, to the technological, such as the need for enhanced realism, accuracy, and AI integration. To overcome these obstacles, interdisciplinary collaboration is needed to harness technological, pedagogical, and design advances to produce simulations that are not only cutting edge in terms of technology but also highly accessible and effective in the classroom.

1.4 Objectives of Each Library

The goals of an AI-powered equity research project, such as the one outlined in the codebase that is provided, are closely related to the features offered by the several Python libraries and tools that are used. By combining the strengths of each component to meet the diverse needs of equity research analysts, these libraries seek to improve the efficiency, accuracy, and insight of the equity research process. I'll outline each library's contribution to the project's overall objectives here.

Unity ML-Agents

An open-source project called Unity ML-Agents Toolkit makes it possible to train intelligent agents in simulations and games. By offering a platform where researchers and developers can train agents in rich, complex environments utilizing deep reinforcement learning, imitation learning, and other machine learning approaches, Unity ML-Agents hopes to close the gap between machine learning and game creation. By enabling the development, testing, and implementation of AI-driven characters and systems within the Unity engine, it seeks to promote AI innovation.

TensorFlow

Google created TensorFlow, an end-to-end open-source machine learning platform. It offers a vast, adaptable ecosystem of tools, libraries, and resources that enable developers to create and implement ML-powered applications with ease and for researchers to advance the field of machine learning. TensorFlow aims to make it easier to create sophisticated machine learning models and algorithms as well as more accessible and scalable for use in a variety of sectors by streamlining the data acquisition, model training, prediction, and refinement processes.

PyTorch

Based on the Torch library, PyTorch is an open-source machine learning library used for tasks like natural language processing and computer vision. It is mostly created by the AI Research lab of Facebook. PyTorch aims to offer robust GPU acceleration support for tensor computation and deep learning. As flexible and user-friendly as possible, it offers dynamic computational graphs that make developing and modifying models simple and straightforward. PyTorch seeks to expedite the process of moving from research prototyping to actual production implementation.

1.5 Innovative Idea

This study offers a novel paradigm for combining artificial intelligence (AI) and quantum computing (QC) to advance space exploration and analysis. The suggested concept aims to solve challenging space exploration challenges that are now beyond the capability of traditional AI techniques and classical computing by combining the unmatched processing capacity of quantum computers with the adaptability and predictiveness of AI. With the goal of revolutionizing our understanding of the universe, this multidisciplinary approach will analyze cosmic data with unparalleled accuracy and efficiency, as well as optimize mission trajectories.

a. Objective

Investigating possible overlaps between AI and quantum computing in the context of space travel is the main goal of this study. The project intends to develop unique methods for navigating interstellar environments, evaluating astronomical data, and modeling cosmic occurrences by utilizing the advanced pattern recognition and decision-making capabilities of AI with the parallel computation capabilities of quantum processors.

b. Quantum-Enhanced Machine Learning Models

Among the most significant advances of this research is the creation of machine learning models with quantum enhancements. Because these models are made to run on quantum processors, they can handle complicated datasets—like those from deep-space observations and astrophysical event simulations—much more quickly. The investigation of celestial bodies and events will be improved by the quantum algorithms, which are specifically designed to optimize machine learning tasks like clustering and classification.

c. Optimized Spacecraft Trajectory Planning

The project will introduce algorithms for optimal spaceship trajectory planning by utilizing QC and AI. To determine the most fuel-efficient routes, this entails the quantum computation of gravity forces and other cosmic variables. Artificial intelligence (AI) systems will forecast changes in the space environment and modify trajectories in real-time to avoid obstacles and maximize mission results. This method is expected to boost the success rate of exploratory missions while drastically lowering mission expenses.

d. Deep Space Communication

Deep space communication is predicted to undergo a revolution as a result of the QC and AI integration. The present speed-of-light barrier to information sharing may be broken by instantaneous communication over great distances thanks to quantum entanglement principles. AI models will control and interpret quantum signals, facilitating data transfer and mission control by guaranteeing dependable communication between Earth and interstellar spacecraft or probes.

e. Cosmic Data Analysis and Simulation

One new use that has been suggested is the advanced analysis and simulation of cosmic phenomena using QC-AI. Because of their better capacity to process large amounts of data and do intricate simulations, quantum computers will be used to simulate events like star formation and black hole mergers. AI will analyze these simulations, offering insights into the principles of physics in harsh environments and making extremely accurate predictions about cosmic events.

f. Scalability and Accessibility

The paper's final discussion point will be the suggested QC-AI framework's scalability and accessibility. Even though there are now little resources available for quantum computing, the study will provide a path for the democratization of quantum-AI tools for space exploration. This includes AI algorithms that can be scaled down to work on ordinary hardware and cloud-based quantum computing services, opening up the technology to a wider scientific community.

g. Enhanced Exoplanet Discovery and Analysis

This research adds to the creative framework by suggesting that the QC-AI integration be leveraged for improved exoplanet detection and analysis. Potential exoplanets can be found with previously unheard-of speed and accuracy thanks to quantum computing's capacity to sort through massive volumes of data from space telescopes and observatories. In parallel, these discoveries can be analyzed by AI systems to forecast the makeup, atmosphere, and possible habitability of these far-off worlds. By taking a dual strategy, the pace of exoplanet discovery might be significantly increased, providing fresh information about their creation, evolution, and potential for extrasolar life. This aspect of the study seeks to increase the number of objects in our cosmic catalogue while also improving our knowledge of planetary systems in the hopes of finding hints about the universal conditions that support life.

1.6 Scope and Application of Project

The suggested research study proposes a transformative method with broad implications and applications for the integration of Quantum Computing (QC) with Artificial Intelligence (AI) in space exploration. This multidisciplinary project aims to broaden the scope of what is possible in the cosmic realm in addition to redefining existing space research approaches. The following five points can be used to examine the scope and application of this novel framework:

a. Revolutionizing Space Exploration Techniques :

The potential to completely transform conventional space exploration methods lies at the heart of the paper's scope. Through the integration of AI's predictive analytics and pattern recognition powers with QC's computational prowess, the framework seeks to improve the efficacy and efficiency of exploratory missions. This entails creating new models to comprehend cosmic occurrences in addition to maximizing spacecraft paths for fuel economy and mission success. More ambitious space missions, such as manned trips to far-off planets or in-depth investigations of the outer solar system and beyond, may result from the application of this knowledge.

b. Advancements in Cosmic Phenomena Simulation and Understanding:

The study suggests using quantum computing (QC) to carry out intricate simulations of cosmic events that are presently outside the purview of classical computer, such the dynamics of galactic formations or the subtleties of black hole mergers. AI's ability to decipher these simulations may offer previously unheard-of insights into the underlying principles of physics and the beginnings of the universe. This could hasten scientific findings, advancing our theoretical knowledge and providing fresh insights into the universe's evolution.

c. Improvement in Deep Space Communication :

Deep space communication technologies promise to benefit greatly from the proposed QC and AI combination. Transcending the current constraints imposed by the speed of light through instantaneous information sharing based on quantum entanglement principles could transform communication with spacecraft or deep space probes. Better mission control and monitoring are made possible by this improvement, which also guarantees the prompt return of scientific data to Earth, expanding our knowledge base as soon as possible.

d. Exoplanet Discovery and Characterization :

The study report emphasizes how QC-AI can be used to speed up exoplanet detection and in-depth investigation. Using this method, exoplanets might be more correctly identified and their properties, such their atmospheric composition and possible habitability, could be more accurately analyzed through sorting through the vast datasets that telescopes have gathered. Such developments could have a major impact on astrobiology, the study of the genesis and evolution of planetary systems throughout the galaxy, and the hunt for planets similar to Earth.

e. Democratizing Space Science :

The article concludes by talking about the QC-AI framework's scalability and accessibility, with the goal of democratizing space science by making cutting-edge technology accessible to a larger scientific community. Through the research, a path toward cloud- based quantum computing and flexible AI algorithms is proposed, opening doors for academics and institutions around the globe to conduct advanced space exploration research without having to make expensive infrastructure investments. This might result in a more diverse scientific community that encourages cooperation and creativity across national and financial boundaries.

CHAPTER 2

LITERATURE SURVEY

2.1 Introduction

A landmark study by Timothy P. Lillicrap et al. [1], "Continuous control with deep reinforcement learning," introduced the Deep Deterministic Policy Gradient (DDPG) algorithm. This development enabled more effective handling of continuous action spaces, applicable in complex environments such as robotics and autonomous vehicles. The introduction of DDPG marked a significant step forward in applying deep learning techniques to real-world continuous control problems.

Following this, John Schulman et al. in "Proximal Policy Optimization Algorithms" [2] proposed the Proximal Policy Optimization (PPO) algorithm. This method balanced performance with computational efficiency, making it ideal for high-dimensional control tasks. PPO has since become a standard tool for training reinforcement learning agents, thanks to its robustness and straightforward implementation.

Similarly, the foundational work by Volodymyr Mnih et al. [5], "Human-level control through deep reinforcement learning," presented the Deep Q-Network (DQN) algorithm. Achieving human-level performance on many Atari video games, DQN demonstrated the potential of deep reinforcement learning to master complex tasks from high-dimensional sensory inputs like images.

Another significant contribution came from Arthur Juliani et al. [18], who developed the Unity Machine Learning Agents Toolkit. This platform has facilitated the experimentation and development of intelligent agents across diverse environments, significantly accelerating the pace of research and practical applications in AI.

Lastly, Kevin Reuer et al. [12] explored the integration of deep reinforcement learning with quantum computing in "Realizing a deep reinforcement learning agent for real-time quantum feedback." This study demonstrated the application of deep learning techniques in optimizing real-time decisions in quantum feedback systems.

2.2 Existing Approaches

The integration of artificial intelligence (AI) into space exploration marks a pivotal chapter in our quest to understand the cosmos. Advanced AI technologies, particularly machine learning and deep learning algorithms, are increasingly being employed to enhance the autonomy and efficiency of exploratory missions beyond Earth. Inspired by the work of Juliani et al. [18], which provides a framework for developing intelligent agents, these AI systems are now fundamental in navigating unfamiliar terrains on distant planets.

For instance, NASA's Mars rovers utilize AI to autonomously detect obstacles and plot safe routes across the Martian landscape, while massive volumes of astronomical data are processed using deep learning techniques, reminiscent of the deep Q-networks discussed by Mnih et al. [5]. This not only facilitates the identification of celestial objects with unprecedented accuracy but also optimizes mission operations in real-time.

Simultaneously, the application of traditional computing methods in the simulation and modeling of cosmic phenomena has proven to be invaluable. High-performance computing (HPC) is extensively used to replicate intricate astrophysical processes, such as black hole mergers and supernova explosions.

This aspect of space science, where robust computational power is a necessity, resonates with the advancements in AI processing efficiency presented by Schulman et al. [2] through their development of Proximal Policy Optimization (PPO) algorithms. These simulations are critical as they enhance our understanding of the universe, providing detailed insights into the physics governing celestial bodies and their interactions.

In the realm of deep space communication, technology continues to evolve to meet the challenges of transmitting data over increasing distances. The use of radio waves, although traditional, has been augmented by technological advancements that allow for higher data transmission rates. The robust data processing capabilities required for such communications draw conceptual inspiration from the Deep Deterministic Policy Gradient (DDPG) algorithm introduced by Lillicrap et al. [1].

Spacecraft Trajectory Optimization: In order to conserve fuel and increase mission capabilities, spacecraft trajectories have been designed using traditional optimization approaches, utilizing the gravity assist maneuver. Evolutionary mission trajectory generator software (EMTG) from NASA is an example of how evolutionary algorithms can be applied to determine spacecraft best trajectories that minimize fuel consumption and trip time.

Exoplanet Discovery: By using the transit method, the Kepler Space Telescope and the Transiting Exoplanet Survey Satellite (TESS) have found exoplanets by using traditional computing techniques. The capacity to go through the data and find possible exoplanets from the faintening of stars when planets pass in front of them has been greatly improved by machine learning techniques.

These current methods have greatly increased space exploration, laying the groundwork for the next big advancement that comes from combining AI with quantum computing. The shift to these cutting-edge technologies should solve the drawbacks of traditional computer techniques and pave the way for further exploration of the cosmos.

2.3 Why we are better

In space exploration, the combination of Quantum Computing (QC) and Artificial Intelligence (AI) represents a paradigm shift, with substantial benefits over current approaches that depend on traditional AI techniques and classical computing. Using the unmatched computational power of quantum computers and the advanced analytical skills of artificial intelligence, this innovative method is set to surpass the constraints of existing technology, offering more profound understandings and effective solutions for space exploration.

Computational Superiority: The quantum leap in computational speed and power is one of the most important benefits of merging QC with AI. Due to superposition and entanglement, qubits—as opposed to classical bits, which can only be 0 or 1—can represent and process enormous amounts of data concurrently. This is how quantum computers work. Due to this, sophisticated algorithms and simulations of cosmic events can be carried out at speeds that are not possible for traditional computers. For example, it becomes possible to forecast celestial events with complex variables or simulate particle interactions in the inside of stars, creating new opportunities for astrophysical research and exploration.

Enhanced Data Analysis: The processing and analysis of astronomical data is revolutionized by the combination of QC and AI. Petabytes of data from telescopes and space missions can be effectively sorted through by quantum algorithms, which can reveal patterns and insights that would be invisible to classical means. This method can significantly enhance the discovery and study of exoplanets, star systems, and other astronomical objects, leading to a deeper comprehension of the structure and dynamics of the cosmos when combined with AI's predictive analytics.

Optimized Space Missions: The viability and effectiveness of space missions depend heavily on the accurate and effective design of spacecraft trajectories. Trajectory planning is improved by the QC-AI integration, which uses quantum algorithms to determine the most fuel-efficient routes and AI to dynamically modify these plans in response to space conditions. Deeper space exploration is made possible and mission expenses are significantly reduced as a result.

Revolutionized Communication: The concept of quantum communication refers to the prospect of instantaneous communication over great distances, made possible by the principle of quantum entanglement in quantum physics. We can accomplish real-time data transmission and control over interstellar distances by combining this with AI. This will improve the safety and responsiveness of both manned and unmanned missions by doing away with the latency problems that plague current deep space communication systems.

Collaborative and Inclusive Research: Lastly, a larger community of scientists and researchers can participate in space exploration initiatives thanks to the democratization of space research made possible by cloud-based quantum computing and AI platforms. This cooperative strategy guarantees a more broad participation in the process of deciphering the mysteries of the cosmos while also encouraging innovation and quickening the pace of discovery.

In conclusion, space exploration benefits greatly from the incorporation of quantum computing and artificial intelligence, which offer a revolutionary framework that surpasses current methods. It not only gets above the constraints of traditional AI and classical computing, but it also opens the door for novel discoveries and breakthroughs in our mission to explore space.

2.4 Future Direction and Challenges

The integration of Artificial Intelligence (AI) and Quantum Computing (QC) in space exploration offers a frontier full of opportunities and inherent difficulties. This intersection has the potential to profoundly alter how we think about the universe, how to get around interplanetary travel, and how to handle all of the data that comes from studying the stars. As we set out on this creative trip, there are many bright opportunities ahead of us, but there are also many challenges that call for creative problem-solving and strategic navigation.

The integration of Artificial Intelligence (AI) and Quantum Computing (QC) in space exploration offers a frontier full of opportunities and inherent difficulties. This intersection has the potential to profoundly alter how we think about the universe, how to get around interplanetary travel, and how to handle all of the data that comes from studying the stars. As we set out on this creative trip, there are many bright opportunities ahead of us, but there are also many challenges that call for creative problem-solving and strategic navigation.

The expectation for AI-driven automation in space missions is growing in tandem with the development of quantum algorithms. The idea of self-governing spacecraft that can navigate the universe and make judgments in real time with little assistance from humans is revolutionary. This change envisions AI leading exploratory missions, skillfully managing unforeseen obstacles and maintaining mission integrity with its sophisticated problem-solving abilities. This automation could usher in a new era of interplanetary travel by greatly improving the effectiveness and safety of space missions.

Deep space communication's latency problems can be resolved with quantum communication, which makes use of the quantum entanglement principle. The advancement of quantum communication networks could transform spacecraft command and control by enabling instantaneous data transmission over great distances. This technology has the ability to protect data transmission from eavesdropping, guaranteeing the integrity of mission-critical information, in addition to facilitating real-time contact with far-off spacecraft.

The finding and study of exoplanets will be revolutionized by the fusion of QC and AI technology. Through the combined power of quantum computing and artificial intelligence, researchers can sort through data from space observatories with never- before-seen accuracy. This combination could help identify future prospects for human colonization by speeding up the process of identifying planets that are habitable and providing fresh insights into the circumstances that support life.

This exciting new frontier is not without its difficulties, though. One major obstacle to quantum computing's full potential is its current technological maturity. Before QC can reach its full potential, problems with qubit stability, error rates, and scalability must be resolved. Furthermore, the amount and complexity of data in space exploration demand AI models that can effectively handle and analyze data—a challenge that calls for ongoing innovation in analytical methodologies and data management.

Another set of logistical and technical issues is integrating and guaranteeing compatibility between QC and AI technologies with current space exploration frameworks. To prevent disruptions and guarantee mission success, the smooth integration of these technologies into spaceship systems and mission regulations demands careful preparation and careful thought.

When it comes to integrating QC and AI into space exploration, ethical and security concerns are equally crucial. Because of these technologies' dual-use potential, care must be taken in both development and implementation to guarantee that their use in space science is guided by moral standards and protected from abuse.

Constrained resources and finances make the process of incorporating QC and AI technology into space exploration even more difficult. These cutting-edge technologies come with a high cost, especially when developing, deploying, and maintaining them in an environment with conflicting priorities and constrained funds.

Finally, because space exploration is a worldwide endeavor, it necessitates international cooperation as well as the creation of common guidelines and standards for the application of AI and QC. Leveraging these technologies to their full potential on a global scale requires navigating geopolitical difficulties and building a cooperative framework.

In conclusion, the application of AI and quantum computing to space exploration portends a revolutionary era that promises to expand our knowledge of the universe and improve our capacity for interstellar travel. But in order to make this vision a reality, a number of obstacles must be overcome, including budgetary limitations, ethical issues, technological barriers, and the requirement for international cooperation. The global scientific community's combined creativity, cooperation, and tenacity will be crucial in forging ahead into this exciting but difficult frontier and paving the way for a new era of space exploration.

2.5 Conclusion

Following an era on the verge of a technological Renaissance, the application of AI and Quantum Computing (QC) to space exploration represents a turning point in the quest to solve the mysteries of space with previously unheard-of accuracy and efficiency. As discussed in the research report, this fusion represents a new strategy that pushes the frontiers of what is possible in space exploration, not just an improvement of current methodologies. As we approach this pivotal moment in history, the consequences of this multidisciplinary collaboration go well beyond the recent gains in space travel, signaling the beginning of a new age of scientific discovery and technological growth.

The combination of QC and AI presents a promising light that points the way to solutions for some of the trickiest and longest-standing problems that astronomers and space scientists confront. The combination of these state-of-the-art technologies promises to push the boundaries of our understanding to the limits of the observable universe and accelerate our search for knowledge, from improving the fidelity of cosmic simulations to opening up new channels for deep space communication. When combined with artificial intelligence's ability to adapt and forecast, quantum algorithms' accuracy and efficiency open up new avenues for research into astronomy, spacecraft navigation, and other celestial phenomena.

But, as the discussion has shown, there are obstacles in the way of this bright future. There are many technological, moral, and practical obstacles to overcome when integrating AI into space travel, especially given the embryonic state of quantum computing technology. To successfully manage these challenges, the international scientific community must put in a concentrated effort that demands creativity, teamwork, and persistence. Furthermore, the deployment of these technologies raises ethical and security concerns that highlight the need for a responsible and cautious approach to ensure that knowledge is pursued in a way that is consistent with humanity's larger interests.

The realization of the complete potential of this integration appears to depend on the democratization of access to QC and AI technology. We can create a more diverse and collaborative atmosphere for scientific study by granting access to these tools to a wider range of researchers and institutions, thereby promoting cooperation across disciplinary and geographic divides. This inclusiveness broadens the pool of potential solutions and ideas while quickening the speed of creation and bringing us one step closer to the answers that exist beyond our planet.

CHAPTER 3

PROXIMAL POLICY OPTIMIZATION AND ITS LEARNING

3.1 Introduction to PPO and its Relevance

Proximal Policy Optimization (PPO) stands as a significant advancement in the field of reinforcement learning, striking a balance between computational efficiency and robust performance. Building on the foundational principles of Trust Region Policy Optimization (TRPO), PPO enhances scalability and simplifies implementation, making it more accessible and practical for a wide range of applications. A key innovation in PPO is the introduction of a clipping mechanism in the optimization process. This mechanism is ingeniously designed to prevent policy updates from deviating too drastically from the current policy, thereby maintaining stability across the learning process and minimizing the risk of significant performance degradation.

This stabilization is particularly critical in environments characterized by high degrees of uncertainty and complexity, such as simulations involving intricate celestial mechanics. For instance, in the dynamic and unpredictable realm of a simulated solar system, PPO’s ability to provide stable and consistent learning updates becomes invaluable. The agent, equipped with PPO, navigates through tasks such as asteroid avoidance, maneuvering through planetary gravitational fields, and achieving stable orbits around celestial bodies. These tasks require a nuanced balance between rapid, reactive maneuvers for short-term challenges and strategic, long-term planning for sustained activities such as orbit maintenance.

Implementing PPO involves careful consideration and tuning of several hyperparameters that significantly influence the algorithm's performance. These include the clipping range, which governs the extent to which policy updates are allowed to deviate from the baseline; the learning rate, which affects the speed and convergence of learning; and the number of training epochs per data batch, which determines how thoroughly the data is utilized for learning before updating the model. The selection and tuning of these parameters are crucial for optimizing the algorithm’s ability to effectively balance the exploration of new strategies against the exploitation of previously successful behaviors.

The practical implementation of PPO in such a complex simulation not only demonstrates the algorithm's robustness and efficiency but also highlights its versatility in managing diverse and dynamic challenges inherent in space navigation. This capacity to handle different aspects of space exploration—from avoiding immediate hazards like asteroids to executing precise orbital insertions—underscores the potential of PPO to serve as a foundational technology for autonomous systems in advanced space operations and beyond. As simulations grow in complexity and the demands on autonomous agents increase, the role of PPO in ensuring effective learning and operational capabilities becomes increasingly crucial, positioning it as a key player in the future of space exploration and other complex autonomous navigation tasks.

As simulations become more intricate, encompassing multiple interacting elements like planets, moons, and asteroid belts, PPO’s role in ensuring the agent’s ability to continuously adapt to changing conditions becomes paramount. This adaptive capability is critical for maintaining the knowledge the agent has accumulated over time while simultaneously integrating new information. Such adaptability is a cornerstone of PPO's design, facilitated by its conservative policy updates which help prevent the agent from drastic shifts in strategy that could lead to operational failures. These incremental adjustments are vital in environments where even minor errors can have significant repercussions, such as navigating through a densely populated asteroid field or adjusting the approach trajectory to a planet with complex gravitational fields.

The effectiveness of PPO in these scenarios hinges on its sophisticated handling of complex decision spaces, where the agent must continually assess multiple variables. These variables include the spacecraft's orientation, velocity, proximity to objects, and fuel levels—each of which can dramatically influence the success of a mission. PPO's architecture is particularly suited for such high-dimensional decision-making processes because it can efficiently manage the exploration of new strategies while exploiting known successful behaviors, ensuring a balanced approach to risk and safety.

Moreover, the unpredictable nature of space, characterized by varying conditions and potential emergencies, demands a learning algorithm that can maintain consistent performance under a wide range of scenarios.

To further enhance the performance and reliability of PPO in simulated solar system environments, careful hyperparameter tuning is essential. This tuning involves refining the algorithm's settings based on the specific characteristics and demands of the simulation. The clipping range, learning rate, and the number of epochs per training batch must be adjusted to find the optimal balance that maximizes learning efficiency and minimizes risk. This process often requires extensive testing and iteration, as the ideal settings can vary significantly based on the dynamics of the simulation and the specific tasks the agent is expected to perform.

The scaling and evolution of PPO models as simulation environments grow in complexity is another critical area of focus. This may involve enhancing the neural network architectures that underpin the agent's policy and value functions, ensuring that they are capable of processing increasingly complex inputs and making more nuanced decisions. Additionally, integrating more sophisticated feedback mechanisms from the simulation can help refine the quality of training, allowing the model to better adapt to the evolving challenges of space exploration. As the simulation expands to include more agents or more complex interactions, such as fleet operations involving coordinated or competitive behaviors, PPO's flexibility and adaptability make it well-suited for these advanced scenarios. This adaptability is vital for ensuring that each agent not only learns effectively on its own but also in conjunction with others, potentially leading to more sophisticated and successful strategies in space exploration missions.

The application of Proximal Policy Optimization (PPO) in simulated solar system environments underscores its capacity to support continual learning and adaptation, which are crucial for the dynamic and often unpredictable realm of space exploration. The continual learning process is facilitated by PPO's clipping mechanism, which effectively balances the need for stability and the pursuit of incremental improvements in agent performance. This mechanism is particularly beneficial in long-duration space missions simulated in Unity, where the agent might encounter diverse conditions over extended periods.

By enabling the agent to make progressive adjustments to its strategy based on real-time feedback, PPO helps maintain a consistent level of performance while avoiding drastic shifts that could compromise mission objectives.

Moreover, the unpredictable nature of space presents unique challenges that demand a robust learning algorithm capable of delivering consistent performance. Space environments can vary dramatically, from the relative predictability of established orbits to the high-risk scenarios of crossing paths with cometary tails or navigating near volatile planetary atmospheres. PPO's conservative approach to policy updates ensures that the agent's learning process remains stable across different environmental conditions, thus providing a safety net against potential malfunctions or miscalculations that could lead to mission failure.

The practical implementation of PPO in these environments involves not only the initial configuration of the algorithm's hyperparameters but also ongoing adjustments based on performance data. This iterative tuning process is critical for optimizing the algorithm’s effectiveness in specific mission scenarios. For instance, the appropriate setting for the clipping range might differ when transitioning from a densely populated asteroid field to a more stable orbital path, requiring adjustments to maintain optimal performance without sacrificing the safety and integrity of the mission.

Furthermore, as simulation scenarios increase in complexity, the scalability of PPO becomes a key factor. This involves not only enhancing the neural network architectures to accommodate more complex input data but also refining the interaction mechanisms between multiple agents within the simulation. For example, fleet operations involving coordinated maneuvers or competitive interactions between agents can introduce additional layers of complexity that require more sophisticated strategies. PPO's ability to adapt to these complex interactions makes it particularly suitable for scenarios that involve multiple agents, where the dynamics of agent interaction can significantly influence the overall success of the mission.

3.2 Simulation Environment setup in Unity

Unity stands out as a highly effective simulation tool for complex systems, including the vast and intricate environment of a solar system. Known for its strong rendering capabilities, Unity is adept at producing high-fidelity graphics, making it ideal for visualizing space scenarios. Its rendering engine facilitates a range of lighting and shading techniques that are crucial for achieving realistic depictions of celestial bodies. These visual capabilities enable the creation of striking portrayals of planets, stars, and other astronomical objects with intricate textures and dynamic effects such as atmospheric scattering and solar flares.

At the core of Unity's effectiveness in simulating a solar system is its robust physics engine, which provides precise modeling of gravitational forces, orbits, and the interactions among various celestial bodies. Leveraging NVIDIA PhysX technology, Unity delivers accurate physical simulations essential for ensuring that the movements and interactions within the solar system adhere to the laws of physics. This level of precision is vital for simulating the complex gravitational pulls between multiple bodies and maintaining the integrity of their orbital paths.

Moreover, Unity is highly customizable and extensible, supporting custom scripts and plugins that can significantly enhance its functionality. This flexibility is particularly beneficial in a solar system simulation, where developers might need to introduce new physics rules or modify existing ones to meet specific scientific requirements. For instance, while Unity does not natively support n-body physics—which are crucial for simulating the gravitational interactions between multiple bodies—developers can implement this through custom scripts based on Newton's law of universal gravitation. These scripts are designed to calculate the gravitational forces exerted among all bodies in the simulation and adjust their velocities and trajectories to reflect these forces accurately.

Unity's capacity for real-time interaction and updates stands as one of its key strengths, essential for maintaining a dynamic and responsive solar system simulation. This feature enables the incorporation of new astronomical events, such as supernovas or planetary discoveries, into the simulation without interrupting its operation. This real-time capability ensures that the simulation remains both current and engaging, reflecting the ever-changing nature of space.

Continuing with the capabilities of Unity in simulating a solar system, the platform's sophisticated collision detection and physics simulation tools play a crucial role in adding realism and scientific accuracy to the model. Unity’s advanced collision detection allows for the meticulous simulation of events like asteroid impacts on planetary surfaces. These collisions can be modeled to reflect actual physical consequences, including changes to a planet's atmosphere, surface topology, and even its orbital path. Such detailed simulations are vital for studying potential disaster scenarios and for educational purposes, providing a vivid depiction of the dynamic and sometimes violent nature of space.

Additionally, Unity’s ability to simulate soft body physics is particularly relevant when dealing with phenomena such as gas clouds and nebulae in space simulations. These aspects require a more nuanced approach to physics that goes beyond the rigid body dynamics typically used for solid objects. By accurately modeling the fluid-like behavior of gases and dust, Unity helps create more immersive and scientifically plausible space environments. These simulations not only serve to enhance the visual appeal of the model but also contribute significantly to its educational and research utility by providing a closer approximation to real-world space phenomena.

The customization options in Unity extend further through its support for real-time programming and data handling, which are indispensable for creating an interactive simulation environment. Developers can integrate external data sources, such as real-time astronomical data or hypothetical scenarios based on current scientific research, to continuously update and evolve the simulation. This integration capability makes Unity not just a static simulation platform but a dynamic tool that can adapt and grow in response to new scientific findings or educational needs.

Unity's scripting and API capabilities allow for the integration of advanced artificial intelligence and machine learning algorithms, which can be used to automate certain aspects of the simulation or to simulate intelligent behaviors. For example, using AI to control the motion of autonomous spacecraft within the simulation can provide insights into the challenges of space navigation and the potential strategies to overcome them. These AI-driven components are particularly useful in testing and developing new technologies for space exploration within a controlled and safe virtual environment.

Its ability to handle complex simulations in real-time, combined with the potential for customization and expansion, makes it a powerful tool for educators, researchers, and developers interested in exploring the vast possibilities of space.

Expanding on Unity's functionality for creating a solar system simulation, the development environment also emphasizes interactivity and scalability, which are essential for both educational modules and scientific explorations. This interactivity is facilitated by Unity’s user interface capabilities, which allow developers to create intuitive controls and informative dashboards that users can interact with to manipulate the simulation. For instance, educators can design interfaces that enable students to adjust the orbits of planets, simulate the effects of gravitational forces, or visualize the impact of celestial events like solar eclipses and meteor showers in real-time. This hands-on interaction enhances learning by allowing users to see the immediate effects of their adjustments, fostering a deeper understanding of complex astronomical concepts.

Scalability is another significant aspect of Unity's environment, supporting the expansion of simulations from a single solar system to an entire galaxy. This feature is particularly beneficial for projects aiming to provide a broader perspective of the universe. Developers can incrementally add more celestial bodies and phenomena, adjusting the level of detail and the physical simulations to maintain performance without sacrificing accuracy. This scalability ensures that the platform can handle the increased computational demands of larger simulations, making Unity ideal for ambitious astronomical projects that aim to capture the vastness and diversity of the cosmos.

Moreover, Unity's graphics and performance optimizations are key to maintaining fluid simulation experiences. Even as the complexity of the simulation grows, Unity's efficient rendering techniques ensure that the visual output remains smooth and visually appealing. These optimizations are crucial for maintaining user engagement, especially in educational settings where the clarity and quality of visual information can significantly impact learning outcomes.

The combination of Unity’s extensive customization options, robust physics modeling, real-time update capabilities, and integration with cutting-edge technologies like VR and AR, positions it as an exceptional tool for creating comprehensive and immersive solar system simulations.

Delving deeper into the practical applications of Unity in the field of space exploration and education, the platform serves not just as a simulation tool but also as a testing ground for space navigation technologies. By integrating deep reinforcement learning algorithms like Proximal Policy Optimization (PPO), researchers can use Unity to develop and refine AI-driven navigation systems within the dynamic and complex environment of a simulated solar system. These AI technologies can learn and adapt to various scenarios within the simulation, from simple orbital maneuvers to complex interplanetary travel, providing valuable insights that can be transferred to real-world space missions.

The significance of Unity in educational settings extends beyond typical classroom learning. It acts as a bridge between theoretical knowledge and practical application, providing students with a platform to experiment with and understand astronomical principles through direct manipulation and observation. This experiential learning approach is invaluable in helping students grasp abstract concepts like gravity, orbital dynamics, and planetary science in a more engaging and intuitive manner.

Furthermore, Unity's simulation capabilities are instrumental in public outreach and engagement. Museums, planetariums, and science centers can leverage Unity-based simulations to create captivating exhibits that allow visitors to explore the solar system interactively. These exhibits can simulate celestial events, provide tours of planets and moons, and even allow users to control space missions. Such interactive experiences are not only educational but also inspiring, potentially sparking interest in astronomy and science among a broader audience.

The future possibilities for Unity in astronomy and space science are boundless. As the platform continues to evolve, it could incorporate more advanced scientific algorithms and even more realistic physical models. Upcoming enhancements could include better support for n-body simulations, improved VR/AR integration for more immersive experiences, and greater scalability for modeling even larger sections of the universe or detailed simulations of space weather phenomena.

3.3 Agent Design

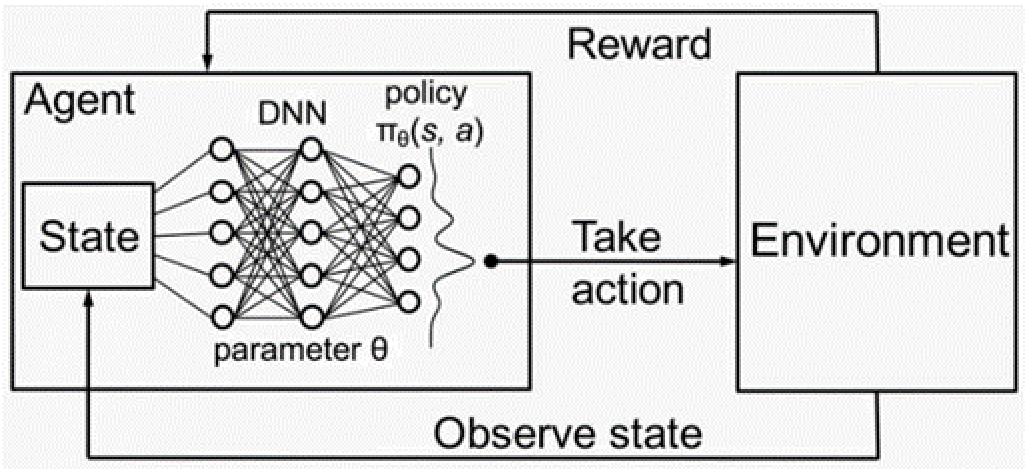


Figure 3.3.1: Agent Environment Interaction Diagram

In the dynamic and intricate environment of a simulated solar system within Unity, the design of the spaceship agent is meticulously crafted to achieve autonomous navigation and exploration. This agent is equipped with an array of sophisticated sensors and advanced navigation tools, essential for interacting intelligently with various celestial bodies and avoiding potential space hazards. The sensor suite on the agent is a critical component, encompassing spatial, spectral, and collision detection sensors. Spatial sensors, such as radar and lidar, are integral for mapping the surroundings by measuring distances to nearby objects, which is crucial for navigation and avoiding obstacles. Spectral sensors play a pivotal role in determining the chemical composition of celestial bodies, helping identify planets and asteroids of interest for scientific research. Furthermore, collision detection sensors are vital for ensuring the safety of the spacecraft, alerting the agent to potential collisions with debris or other space objects and enabling it to perform timely evasive maneuvers.

The navigation tools of the agent consist of integrated systems that utilize the data from these sensors to execute complex maneuvers and maintain the spacecraft's trajectory towards its designated targets. Star trackers are employed to maintain and adjust the spacecraft’s orientation in space by observing stars and other celestial markers. The propulsion systems, comprising various thrusters and advanced propulsion technologies, allow for precise movements and trajectory adjustments as dictated by the navigation strategy.

The implementation of the Proximal Policy Optimization (PPO) algorithm, known for its effectiveness in managing environments with high uncertainty and complexity, is a cornerstone of the agent’s design. Using Unity's ML-Agents toolkit, PPO is integrated to allow the agent to interact with the Unity environment effectively. This setup necessitates a continuous influx of environmental and performance data, which includes observations from the agent’s sensors, the actions taken, and the rewards received from these actions. PPO employs a policy iteration method based on a clipped surrogate objective function, which is crucial for ensuring that updates to the agent's policy are substantial enough to be effective, yet moderate enough to prevent destabilizing the model. This balance is vital for the continuous learning and adaptation of the agent as it navigates through the simulated solar system, dynamically tuning its responses to optimize the balance between exploring new strategies and exploiting known successful behaviors.

The neural network architecture that underpins the agent's decision-making process is a deep neural network, optimized to handle high-dimensional sensory input and execute complex decisions. The network comprises an input layer that receives raw data from various sensors, including spatial positions and velocities, as well as spectral analyses. This layer is specifically designed to manage the diverse inputs needed for comprehensive space navigation. Following this, multiple hidden layers equipped with a high number of neurons work to extract features and patterns from the input data, typically employing rectified linear activation functions (ReLU) to enhance computational efficiency and maintain effective gradient flow. The output layer of the network produces a vector of possible actions for the agent, ranging from trajectory adjustments to analytical tool deployments, with softmax functions used for selecting discrete actions and linear activations for continuous actions.

This sophisticated arrangement ensures that the agent's neural network not only processes the complex inputs effectively but also generates strategic outputs that guide the spacecraft through the simulated cosmos. The continuous evaluation of network performance through a feedback loop involving rewards from the agent's actions allows for real-time adjustments to the network's parameters. This optimization process, guided by the principles of PPO, ensures the maximization of cumulative rewards while minimizing potentially disruptive shifts in policy, thus equipping the agent to adapt to new challenges and refine its strategies within the dynamic environment of the simulated solar system.

As the spaceship agent continues to evolve within the Unity environment, its ability to adapt and respond to increasingly complex scenarios becomes critical. This adaptability is significantly enhanced by further refining the integration of the agent's sensor technology with its decision-making systems. The depth and variety of data collected through the agent’s sensors—spanning spatial, spectral, and collision detection data—form a comprehensive dataset that informs all navigational and operational decisions. Integrating this data effectively requires advanced data fusion techniques that allow the agent to process and synthesize information from multiple sources quickly and accurately.

Data fusion in this context involves combining the different modalities of sensory information to form a unified overview of the agent’s environment. This is crucial when navigating through space, where information from a single source might be incomplete or insufficient for safe navigation. For example, spatial sensors providing distance measurements need to be corroborated with spectral data to ensure that the objects being navigated are identified correctly, not just in terms of location but also composition. Similarly, collision detection sensors that alert the agent to potential hazards must work seamlessly with the spacecraft’s navigational tools to execute immediate and effective evasive maneuvers.

Further enhancing the agent's capabilities, simulation tools within Unity can be employed to create predictive models that forecast potential future scenarios based on current and historical sensor data. This predictive capability allows the agent to prepare for and react to potential conditions before they occur, such as anticipating the trajectory of a nearby asteroid or predicting the emergence of a solar flare. These models can be dynamically updated in real-time, providing the agent with ongoing assessments of potential future states, thereby allowing for proactive rather than merely reactive decisions.

In addition to predictive modeling, machine learning techniques, especially those involving unsupervised learning, can be instrumental in identifying patterns or anomalies in the data that may not be immediately apparent to human operators or traditional programming approaches.

For instance, clustering algorithms can reveal hidden structures in the data or unexpected relationships between different celestial phenomena. These insights can lead to more informed decision-making and enhanced scientific discovery, enabling the agent to identify areas of interest that may require closer examination or direct intervention.

Moreover, the agent's ability to interact and communicate within a fleet of similar agents or other autonomous systems introduces an additional layer of complexity and capability. Multi-agent systems require not just individual learning and decision-making but also coordination and shared strategies that can significantly enhance the efficiency and effectiveness of space missions. For example, agents can share sensor data and decision logs to create a more comprehensive map of their environment or coordinate their efforts in scientific exploration or hazard avoidance.

Ultimately, the continuous development and refinement of the spaceship agent’s capabilities in Unity showcase a progressive move towards creating not only autonomous agents capable of navigating and performing tasks within a simulated solar system but also forming part of a collaborative network of agents. This networked approach leverages collective intelligence and data sharing to enhance mission outcomes and represents a forward-thinking perspective on managing complex, dynamic environments such as space. Through these advancements, the agent design not only supports current space exploration objectives but also paves the way for future missions that might involve more extensive interplanetary travel or the autonomous construction and management of space-based infrastructure.

These technological advancements not only improve the operational capabilities of space missions but also push the boundaries of what can be achieved in space exploration. By harnessing the power of immersive technologies and securing communications, the Unity-based simulation and control system prepares future space missions for more ambitious endeavors beyond the current frontiers, opening up new possibilities for exploration and interaction in the vast expanse of space.

3.4 Real Time Data Generation and Interaction

In the dynamic and meticulously designed environment of a simulated solar system using Unity, real-time data collection is the cornerstone that empowers the spaceship agent to learn and adapt with high efficiency. This process of continuous data generation encompasses capturing information in real-time as the agent interacts with various elements within the simulation. Such an approach guarantees that the collected data is not only up-to-date but also highly relevant to the immediate challenges and environmental contexts that the agent faces.

Data collection in this simulated environment is multi-faceted, primarily focusing on:

1. **Sensor Output:** The spaceship's suite of sensors plays a pivotal role, providing a steady stream of detailed environmental data. These include spatial, spectral, and collision detection sensors that gather critical information about distances to nearby objects, their compositions, and potential collision threats—data that is essential for navigation and safety.

2. **Agent Actions and Responses:** Every maneuver made by the agent, from trajectory adjustments to speed changes, and the outcomes of these actions, such as the successful avoidance of obstacles or less optimal results, are meticulously recorded. This record forms a rich dataset that feeds into the agent's learning algorithm, enabling it to refine its strategies and responses continuously.

3. **Environmental Changes:** The simulation captures dynamic environmental changes—such as the movement of celestial bodies or the sudden appearance of new obstacles like passing comets—that can affect navigation and strategy. Understanding these dynamics is crucial for the agent as it allows for better preparedness and adaptability to the ever-changing space environment.

The types of data collected are diverse, each adding a layer of depth to the agent's understanding and strategy development:

1. **Spatial Data:** This includes the absolute and relative positions of objects and their trajectories, which helps in mapping out navigational routes and predicting future positions of these objects.

2. **Velocity Data:** Details the speeds and directional changes of various objects, crucial for assessing dynamic conditions that could impact the agent's navigation and safety.

3. **Encounter Outcomes:** Provides direct feedback on the agent's interactions with the environment, indicating the success of navigational strategies and contributing to the learning process.

4. **Spectral Data:** Offers insights into the chemical and physical properties of celestial bodies, enriching scientific exploration and mission planning.

The creation of a dynamic dataset from this continuous data collection is invaluable for the ongoing training and enhancement of the spaceship agent's capabilities. This evolving dataset not only supports the agent's ability to continuously learn and adapt to new scenarios and changes but also facilitates the refinement of its decision-making policies. With each new piece of data, the agent's neural network, powered by Proximal Policy Optimization (PPO), updates its strategy, increasingly refining its actions to better align with the mission goals and environmental realities.

This system of real-time data interaction and generation within the Unity-based simulated solar system is pivotal for developing a highly responsive and adaptable agent capable of handling the complexities and unpredictabilities of space exploration. It ensures that the spaceship agent is not only reacting to current conditions but is also prepared for future challenges, thereby enhancing its effectiveness and reliability as a tool for both exploration and research.

As the real-time data generation process continues to evolve within the Unity-based simulation, the significance of managing and analyzing this voluminous and complex dataset becomes increasingly apparent. Efficient data handling and analysis are crucial for transforming raw data into actionable insights, which directly influence the agent's learning and decision-making processes. This conversion is facilitated through advanced data processing techniques that categorize, filter, and interpret the data to ensure it is optimally utilized for training the spaceship agent.

Data processing in this simulated environment involves several key stages:

1. **Data Filtering:** Given the vast amount of data generated, not all collected data is equally relevant or necessary for effective learning. Data filtering helps in prioritizing information that contributes significantly to decision-making processes, such as identifying critical spatial or spectral data that indicates navigational hazards or scientifically valuable targets.

2. **Data Integration:** This stage involves synthesizing data from multiple sensors to create a cohesive understanding of the environment. For instance, integrating spatial data with velocity data can provide a more comprehensive picture of the motion dynamics of celestial bodies, crucial for advanced navigational algorithms.

3. **Pattern Recognition:** Utilizing machine learning algorithms, particularly unsupervised learning techniques, to detect patterns and anomalies in the data which might not be obvious through traditional analysis. This capability is vital for recognizing new or changing conditions that the agent must adapt to, such as unexpected asteroid fields or unusual planetary activity.

The insights derived from these data processing activities are instrumental in continuously updating and refining the spaceship agent's policies. By employing Proximal Policy Optimization (PPO), the agent leverages these insights to make informed decisions that are both timely and contextually appropriate. The PPO algorithm, with its capability for handling large and complex datasets, ensures that each policy update is based on a thorough analysis of the most relevant and current data, maintaining the agent's performance and adaptability in a dynamic environment.

The integration of real-time data within the simulation also extends to enhancing the interactive capabilities of the spaceship agent, allowing for more sophisticated training and operation scenarios. The real-time interactivity facilitated by Unity's powerful simulation environment is crucial in mimicking the volatile nature of space, where conditions can change rapidly and unpredictably. This setup allows the agent to practice and refine its responses under a variety of scenarios that closely replicate potential real-world conditions. For example, the agent can be tested on its ability to navigate through an unexpectedly dense asteroid belt or respond to a sudden shift in a planet's magnetic field, experiences that are vital for developing robust navigational strategies.

Moreover, the real-time data stream supports a form of interactive learning known as reinforcement learning, where the agent continually adjusts its actions based on the immediate consequences of its decisions. This form of learning is essential in environments as dynamic and unpredictable as space, where pre-programmed responses may prove insufficient and the ability to adapt quickly is critical. The spaceship agent, by receiving instant feedback on its actions through the simulation's reward system, learns to optimize its decisions to achieve better outcomes, whether this involves conserving fuel, minimizing travel time, or avoiding potential hazards.

The continuous loop of action, feedback, and adjustment not only accelerates the learning curve of the agent but also ensures that the learned behaviors are deeply ingrained and readily accessible when needed. This capability is especially important when considering the deployment of such technologies in actual space missions, where autonomous decisions need to be both quick and highly reliable. The agent's ability to interact with the simulation in real time also facilitates the testing of various "what-if" scenarios, enabling researchers and engineers to evaluate the agent's responses to a wide range of conditions and situations that could occur during actual space missions.

The capability to visualize and interact with the simulation in real time transforms the training process from a static, linear exercise to a dynamic, multi-dimensional learning experience. This approach not only enhances the agent's performance and reliability in navigating through a simulated solar system but also significantly contributes to our understanding of complex space environments.

3.5 Observations and Actions

In the sophisticated simulation of a solar system developed using Unity, the spaceship agent is endowed with a variety of sensors tailored to capture detailed environmental data crucial for its operations. These sensors are central to the agent's ability to perceive and understand the vastness of space around it, serving as the primary input for all navigational and scientific tasks. Spatial sensors, for instance, deliver precise information about the position, orientation, and distance of various objects within the agent's vicinity, which is indispensable for safe navigation and effective collision avoidance. Similarly, velocity sensors provide insights into the speed and directional movement of both the agent and other celestial objects, enabling accurate trajectory plotting and speed adjustments necessary for mission success.

Moreover, the agent is equipped with spectral sensors that play a critical role in its scientific exploration duties. These sensors analyze the chemical composition of celestial bodies, aiding in the identification of planets and asteroids that may hold scientific interest or valuable resources. Collision detection sensors complement this suite by ensuring the safety of the spaceship, alerting it to potential collisions with space debris, asteroids, or other celestial bodies, thus preventing mission-compromising accidents.

The raw data collected by these diverse sensors undergoes sophisticated processing involving a series of filters and algorithms designed to extract and refine the information needed for decision-making. Techniques such as noise reduction, data smoothing, and transformation into formats suitable for neural network processing ensure that the inputs are reliable and actionable. This processed data feeds into the agent's decision-making system, powered by the Proximal Policy Optimization (PPO) algorithm, which effectively maps observations to actions.

The decision-making process within the spaceship agent utilizes a neural network architecture to interpret the refined sensory data, assessing the dynamics of the environment, including distances to objects, their trajectories, and potential threats or points of interest. The neural network then outputs a probability distribution of potential actions, guided by the current policy maintained by PPO. This policy is constantly updated based on feedback from the environment, which comes in the form of rewards or penalties following the agent's actions.

For instance, successfully navigating through a challenging asteroid field would reinforce the actions taken, encouraging the agent to repeat similar maneuvers under comparable circumstances.

PPO plays a critical role in ensuring that the policy updates are balanced, promoting exploration of new strategies while also capitalizing on known successful behaviors. The clipping mechanism within PPO is particularly vital as it prevents overly drastic policy changes that could destabilize the learning process, thus maintaining a stable progression towards optimal decision-making.

The actions available to the spaceship agent are meticulously defined to cover a comprehensive range of navigational and exploratory activities within the solar system. These include accelerating to reach a destination more quickly or to escape a gravitational pull, decelerating to approach objects of interest safely, changing direction to navigate around obstacles or toward targets, collecting scientific data, and sending communications. Each action is parameterized and selected based on the neural network's output, which evaluates the potential rewards of each action within the context of the current environmental state and the established policy.

This intricate system of sensors, data processing, and decision-making through PPO enables the spaceship agent not only to perform effectively in the immediate tasks at hand but also to continuously learn from its experiences. By adapting its strategies over time, the agent enhances its autonomy and capability in handling the complexities of space navigation and exploration. This ongoing learning and adaptation process is crucial for the success of missions within such a dynamic and unpredictable environment as space.

Continuing from the initial understanding and operational setup of the spaceship agent in the Unity-simulated solar system, the agent's ability to process information and make decisions in real-time becomes increasingly refined through its interaction with the environment. This ongoing process is essential for the agent to effectively manage the complexities of space navigation and scientific exploration, adapting its strategies in response to new data and scenarios.

1. In-depth Exploration of Real-time Decision-Making:

- **Integration of Sensory Data:** The comprehensive suite of sensors provides the agent with a detailed perception of its surroundings. This data is crucial for immediate decisions and is integrated in real-time, allowing the agent to react to dynamic changes in the environment such as the sudden appearance of an asteroid or an unexpected alteration in a planet's orbit.

- **Predictive Analytics:** Leveraging advanced machine learning techniques, the agent employs predictive models to forecast potential future scenarios based on current and past sensor inputs. This ability enhances the agent's predictive capacity, preparing it for potential future states and enabling proactive rather than reactive decisions.

2. Dynamic Adaptation through Continuous Learning:

- **Feedback Loop Efficiency:** The feedback from each action taken by the agent—whether successful navigation through an asteroid field or an unsuccessful attempt to analyze a celestial body—is immediately processed. This feedback is essential for adjusting the learning algorithms in real-time, ensuring that the agent continuously improves its accuracy and decision-making efficacy.

- **Policy Optimization and Refinement:** Utilizing the PPO algorithm, the spaceship agent regularly updates its decision-making policies based on real-time interactions and feedback.
  This process involves:
  1) Evaluating the effectiveness of current strategies and making incremental improvements.
  2) Ensuring that updates to the policy are balanced, avoiding drastic changes that could destabilize the agent’s performance.

3. Strategic Action Selection:

- **Action Execution:** Based on the processed data and the current policy, the agent executes actions that are calculated to yield the best possible outcomes. These actions range from adjusting the spacecraft's speed and direction to initiating scientific data collection or communicating findings back to mission control.

- **Complex Decision-Making:** The decision-making process is inherently complex, involving a multitude of factors such as the agent’s current state, immediate environmental conditions, and long-term mission goals. Actions are selected not only for their immediate benefits but also for their potential long-term advantages, optimizing both safety and mission success.

4. Enhanced Operational Capabilities through Simulation:

- **Scenario Testing:** The Unity environment allows for the simulation of various complex scenarios that the agent might face in actual space missions. By testing different strategies in simulated high-risk environments, the agent can learn to navigate through real-world space conditions more effectively.

- **Adaptive Learning:** As the simulation progresses, the spaceship agent adapiles its strategies based on accumulated experiences, which enhances its capability to handle unexpected situations. This adaptive learning process is crucial for missions where flexibility and quick thinking are required for success.

In summary, the observations and actions of the spaceship agent within the Unity-based simulated solar system are governed by a sophisticated framework that combines real-time data processing, adaptive learning, and strategic decision-making. This setup not only enables the agent to respond effectively to immediate environmental conditions but also to learn from these interactions and refine its strategies over time, thereby enhancing its autonomy and effectiveness in complex navigation and exploration tasks. This ongoing evolution in capabilities showcases the potential of simulation technologies to prepare autonomous systems for the demanding conditions of space exploration.

3.6 Reward System and Policy Improvement

In the Unity-simulated environment for space exploration, the reward system serves as a foundational element that drives the learning process of the spaceship agent. This system is intricately configured to align with the complex goals of navigating and exploring a virtual solar system, guiding the agent's behavior through feedback that reflects the success or failure of its actions in relation to mission objectives. Central to this system is the application of the Proximal Policy Optimization (PPO) algorithm, which uses the rewards and penalties to refine and enhance the agent's decision-making capabilities.

The reward mechanism is designed with several critical criteria in mind, each tailored to encourage behaviors that contribute to the overall success of the mission. For example, the agent receives positive reinforcement for efficiently navigating between points of interest, such as planets or asteroids, without incurring collisions. This not only promotes the skill of precise maneuvering but also underscores the importance of safe navigation. Additionally, the agent is rewarded for successful scientific data collection, which involves utilizing its array of sensors to gather valuable information when near celestial bodies, furthering the scientific goals of the mission.

Equally important are the rewards for effective collision avoidance, where the agent learns to identify and evade potential hazards like space debris, an essential skill for ensuring long-term operational safety. Fuel efficiency also plays a significant role in the reward system; by optimizing routes and speeds to conserve fuel, the agent adheres to realistic space mission constraints, adding an additional layer of complexity to its learning environment. Conversely, penalties are assigned for actions that negatively impact the mission, such as colliding with objects, deviating from the planned mission path, or failing to gather important scientific data. These penalties serve as crucial negative feedback, discouraging the agent from engaging in risky or non-productive behaviors.

The optimization of rewards is a dynamic and ongoing process within the PPO framework. The algorithm continuously evaluates the agent's policy by simulating actions within the environment and observing the resultant rewards.

To ensure that updates to the agent's policy are beneficial and do not destabilize the learning process, PPO employs a clipped surrogate objective function. This function modifies the optimization criterion to prevent excessive deviations from the existing policy, which helps maintain stability in the agent's learning trajectory. The clipping mechanism specifically limits the ratio of new policy probabilities to old ones, ensuring that updates remain within a reasonable range to foster gradual and consistent improvement.

Furthermore, the policy updating process incorporates methods like stochastic gradient descent to methodically adjust the policy parameters. This method iteratively refines the policy towards maximizing the rewards, with PPO running multiple epochs over the same batch of data before discarding it, allowing for an in-depth learning from each set of experiences. This approach not only improves the efficiency of the learning process but also enhances the robustness of the agent's decision-making capabilities.

Through these mechanisms, the reward system and policy improvement strategy in the simulated solar system environment are meticulously crafted to support the agent's development. By providing clear, mission-aligned incentives and employing advanced optimization techniques, the agent is continually guided towards behaviors that enhance its effectiveness and efficiency in space exploration tasks. This structured and strategic approach to learning ensures that the agent not only performs its current tasks competently but also adapts to new challenges and complexities as the mission evolves.

The refinement of the agent’s policy through the Proximal Policy Optimization (PPO) algorithm is crucial for adapting its behavior to the demanding requirements of space exploration within the Unity simulation. This refinement is rooted in the continuous and meticulous analysis of the agent's interactions and the outcomes of its actions, facilitated by the reward mechanism. Each successful maneuver, whether it involves navigating through an asteroid belt without incident or effectively gathering data from a distant planet, not only provides immediate feedback but also serves as a learning point that informs the policy update process.

This process of policy refinement is intricately designed to be both adaptive and conservative, ensuring that the agent's learning progression is stable yet responsive to new information. The PPO algorithm achieves this balance through its distinctive approach to policy updates, where it considers both the short-term benefits of specific actions and their long-term implications. By calculating the advantage of each action—the expected payoff over what was predicted—PPO can assess the relative value of different strategies under varying conditions. This is critical in a dynamic environment like space, where unexpected challenges can arise, requiring quick yet informed decision-making.

Moreover, the clipping mechanism within PPO plays a pivotal role in maintaining the stability of the learning process. By setting thresholds for how much the policy is allowed to change with each update, it prevents drastic shifts that could lead to performance degradation or erratic behavior. This mechanism is particularly beneficial in complex simulations where the agent must continually adapt to new scenarios without losing the gains made from past learning experiences. The stability provided by the clipping ensures that the agent does not deviate too far from proven strategies, but still has the flexibility to incorporate new lessons learned from recent actions.

The iterative nature of the policy update process also contributes significantly to the agent's development. PPO does not rely on a single instance of feedback but instead uses multiple epochs of data to refine the agent's strategy. This repeated analysis of actions allows the algorithm to build a more accurate and robust understanding of effective behaviors. Each iteration helps to fine-tune the agent’s decisions, ensuring that each action taken is more aligned with the overarching mission objectives, such as maximizing scientific output while minimizing risks and resources.

The feedback loop, a core component of this learning process, is meticulously crafted to ensure sensitivity to both the immediate results of actions and their longer-term outcomes. This dual focus is essential for space missions where short-term decisions can have long-lasting impacts. The feedback mechanism evaluates every action based not only on its immediate success or failure but also on its contribution to the agent’s future capabilities and mission goals. For instance, choosing a fuel-efficient trajectory might delay arrival at a target but saves resources for later phases of the mission, a trade-off that the agent learns to navigate effectively.

Through these sophisticated policy improvement and reward optimization strategies, the agent is equipped to handle the complexities of navigating a simulated solar system. This continuous cycle of action, evaluation, feedback, and adjustment forms the backbone of the agent's ability to learn and adapt effectively. As the agent evolves, so too does its capability to undertake more complex and demanding tasks, reflecting the progressive nature of learning in high-fidelity simulations like those created in Unity. This ongoing advancement is crucial for preparing autonomous systems for actual space exploration, where adaptability and decision-making precision are paramount.

The continuous improvement of the agent’s decision-making capabilities through the reward system and policy updates in the Unity-simulated solar system is intricately linked to the overall simulation fidelity and the complexity of the scenarios presented. As the agent encounters diverse and challenging environments, the sophistication of the reward structure and the policy refinement mechanisms are tested and honed. This evolutionary process is vital for developing a robust autonomous system capable of sophisticated space missions.

The complexity of the simulation plays a critical role in the agent’s development. As the simulated environment becomes more intricate, with a variety of celestial bodies exhibiting unique physical and orbital characteristics, the agent is required to make increasingly complex decisions. Each decision must consider multiple factors such as distance, velocity, potential hazards, scientific value, and mission priorities. The reward system is designed to encourage the agent to navigate these complexities successfully, promoting strategies that optimize scientific discovery while ensuring safety and resource conservation.

To handle the increased complexity, the reward mechanism and the PPO algorithm work in tandem to provide nuanced feedback that is both specific and contextual. The agent receives detailed feedback for actions based on their outcomes, which directly influences subsequent policy updates. For example, if the agent successfully navigates a particularly challenging asteroid field using minimal fuel, it receives a higher reward, reinforcing the strategies employed during the maneuver. Conversely, penalties for risky maneuvers that result in near-misses or minor collisions provide a learning opportunity, guiding the agent to adjust its approach.

The iterative learning process enabled by PPO is essential for the agent to refine its strategy over time. Each cycle of action and feedback not only adjusts the immediate responses of the agent but also contributes to a deeper understanding of the environment. This ongoing process allows the agent to develop a layered strategy that incorporates both the learnings from past experiences and adaptations to new challenges. The use of multiple epochs in policy updates ensures that the agent's strategy is robust, accounting for various potential scenarios and outcomes, rather than optimizing for a single, possibly isolated event.

Moreover, the agent’s ability to learn and adapt is enhanced by the simulation’s capacity to introduce random or rare events that can test the limits of the agent’s current policy. This includes unexpected phenomena like solar flares or sudden shifts in planetary alignments, which require the agent to quickly reassess its strategy and adapt. These events are crucial for testing the flexibility and resilience of the agent’s decision-making framework, ensuring that the system is not only effective under normal conditions but also capable of handling crises or anomalies.

Additionally, the agent's performance and the efficacy of the learning process are continually monitored through comprehensive analytics that track various metrics such as decision speed, accuracy, fuel efficiency, and scientific data collection rates. These metrics provide critical insights into the agent's proficiency and areas needing improvement, guiding further refinements in the training process and reward system. By analyzing these metrics, researchers and developers can identify patterns or shortcomings in the agent’s strategy, enabling targeted adjustments to the simulation and learning algorithm.

Through this detailed and dynamic approach to training and development, the spaceship agent progressively enhances its capability to manage the demands of space exploration. The sophisticated interplay between the reward system, policy updates, and the simulated environment ensures that the agent not only learns to perform its current tasks more effectively but also builds the foundation for handling future, more complex missions. This continuous adaptation and learning are key to preparing the agent for real-world applications, where the unpredictability of space can pose unique challenges and opportunities.

3.7 Flowchart Outlining operational logic of ML Agent

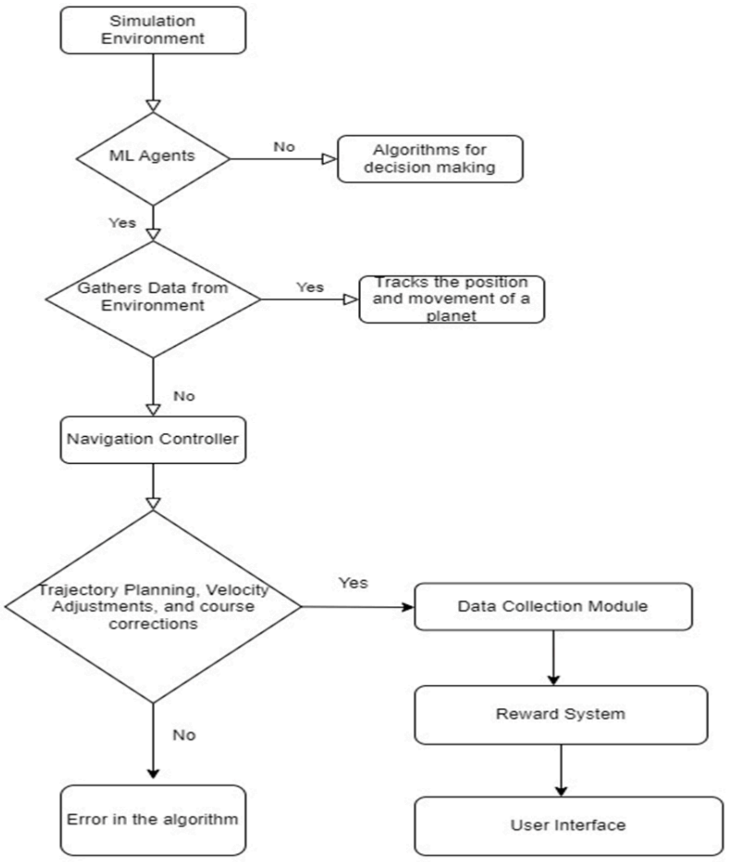

Simulation Environment
ML Agents
No
Algorithms for decision making
Yes
Gathers Data from Environment
Yes
Tracks the position and movement of a planet
No
Navigation Controller
Trajectory Planning, Velocity Adjustments, and course corrections
Yes
Data Collection Module
Reward System
User Interface
No
Error in the algorithm

Figure 3.7.1: Flow Diagram of the Entire Project

In the elaborately designed architecture of the deep reinforcement learning environment simulated in Unity, the simulation environment stands as the expansive canvas upon which the entire system operates. This virtual cosmos, an intricate representation of a solar system, is not merely a backdrop but an interactive ecosystem buzzing with celestial activity. It is rich with planets in their orbits, asteroids on their lonely paths, comets streaking with ice and dust, and perhaps even human-made spacecraft silently gliding through the void. The environment is governed by the laws of physics that dictate the motion and interaction of these bodies, creating a dynamic and ever-changing landscape for the machine learning agents to navigate.

Central to this architecture are the ML Agents, the avatars of artificial intelligence that learn to thrive in this digital expanse.

These agents are embedded with the capability to perform a variety of tasks through deep reinforcement learning—each agent learning from its interactions with the environment, adjusting its behaviors based on the consequences of its actions, and continuously striving to achieve its defined objectives.

The agents operate by gathering data, a fundamental task that forms the core of their learning process. They are equipped with virtual sensors mirroring the capacity of real-world instruments, measuring distances, velocities, and compositions of objects within their vicinity. They scan and scrutinize their surroundings, gathering streams of data that feed into their neural networks. This information is the raw material from which knowledge is extracted, the grist for the mill of decision-making algorithms that lie at the heart of the agents.

When not actively engaging with the environmental data, the agents turn to their internal decision-making processes. Here, the PPO algorithm comes into play, processing the data previously gathered to determine the next best action. The decision-making is not haphazard but a calculated affair, deeply rooted in the current policy that the agent follows—a policy that is, in turn, shaped by the rewards and penalties that the algorithm has learned to associate with various actions.

A specialized function of the agents is to track the position and movement of a planet within the simulation. This is not a mere tracking task but a complex dance of prediction and response, where the agents anticipate the celestial body's path and adapt their own trajectory in real-time. It's an intricate task, representative of the agents' broader mission to explore, analyze, and understand the workings of the simulated solar system.

When data gathering is on pause, the navigation controller assumes command. This subsystem is imbued with pre-established rules and heuristics that dictate the agent's navigation when autonomy takes a backseat. The navigation controller is the agent's autopilot, guiding it along predetermined paths or taking control when the situation falls outside the bounds of the agent's learning.

From the decisions made by the PPO algorithm, the agents embark on trajectory planning, making velocity adjustments and course corrections as needed. This step is where the theoretical transforms into the practical, where calculations and predictions become actual movements within the simulated space. Each adjustment is a decision brought to life, a virtual thrust or turn that propels the agent toward its goal.

The success of these navigational endeavors is captured and consolidated by the data collection module. Here, the agent's actions and the outcomes are logged, building a repository of experiences. This module is not merely a record keeper but a lynchpin in the learning process, collecting the fruits of the agent's labor for future analysis.

At the core of the reinforcement learning paradigm is the reward system, the mechanism that defines the success or failure of the agent's actions. It is the agent's guiding star, illuminating the path to mission success by assigning value to each action based on criteria such as efficiency, safety, data quality, and adherence to mission parameters. This system encourages the agent to repeat beneficial behaviors and deters it from taking actions that could jeopardize its objectives.

The culmination of the agent's activities is presented through the user interface, a portal through which the human overseers can monitor and interact with the agent's operations. This interface serves as a bridge between the complex data and decision-making processes of the agent and the intuitive comprehension of the human user. It provides visualizations, analyses, and real-time updates, ensuring that the human participants remain informed and in control.

Errors in the algorithm are not mere setbacks but opportunities for growth and improvement. When the agent's actions do not yield the expected results, the system flags an error, triggering a phase of assessment and adjustment. Here, the architects of the simulation—the programmers and engineers—step in to debug and enhance the algorithm, propelling the agent's learning forward.

This sophisticated architecture, with its interweaving of data collection, decision-making, action, and feedback, creates a rich tapestry of learning and adaptation. It represents a microcosm of autonomous intelligence, a testament to the power of simulation in mirroring the challenges of real-world space exploration. The first page of this exploration has set the stage for a deeper dive into the agent's journey through its digital universe.

As the simulation progresses, the intricate mechanisms that enable the spaceship agent to learn and adapt become increasingly apparent. The agent's capacity to process and interpret the vast quantities of data collected from its sensors is not a static capability but a dynamic one that evolves over time. The more the agent interacts with the virtual environment, the more refined its data processing becomes. This evolution is key to the agent's ability to navigate through the complex, simulated solar system with greater precision and autonomy.

Within the simulation, the agent relies heavily on the data it collects to inform its decision-making. Each piece of data represents a puzzle piece of the larger picture of the solar system's environment. Spatial data enables the agent to orient itself, velocity data informs its movement and speed, and spectral data unveils the composition of celestial bodies. The agent’s neural network synthesizes this information, painting a comprehensive picture of the surrounding space that becomes the basis for each subsequent decision.

When the agent processes this data and no immediate environmental changes necessitate a reaction, the decision-making algorithms begin their work. These algorithms, rooted in the Proximal Policy Optimization (PPO) technique, weigh the potential outcomes of various actions against the backdrop of the agent's current objectives. This is where the true strength of the agent lies — not just in its ability to act but to act with intention and foresight. The algorithms consider not only the immediate impact of an action but also its downstream effects. For example, a course correction might avoid a nearby obstacle, but it also needs to be weighed against fuel consumption and the agent's long-term mission goals.

Tracking the movement and positions of celestial bodies like planets is one of the agent's specialized functions. This task requires an acute understanding of gravitational forces, orbital dynamics, and the ability to forecast future positions based on current trajectories. It's a delicate task that combines raw computational power with sophisticated modeling to anticipate the dance of the cosmos.

The navigation controller serves as a critical subsystem when data gathering is not in progress. It ensures the agent maintains its course or executes maneuvers according to preprogrammed directives.

This might include executing a slingshot maneuver around a planet to conserve fuel or orienting the spacecraft to ensure solar panels are optimally positioned to gather energy. The navigation controller's directives provide a safety net of sorts, an underlying layer of programming that the agent can fall back on when outside the bounds of learned behaviors.

As the agent acts within the simulation, adjusting its trajectory and speed, the data collection module quietly operates in the background. It captures the results of each action, feeding this information into the reward system that evaluates the agent’s performance. This module doesn't just collect data; it contextualizes it, providing a narrative of the agent's journey through space, chronicling its successes and learning opportunities.

The reward system is the crucible within which the agent's policy is tempered and strengthened. This system assigns value to each of the agent's actions, reinforcing those that contribute positively to mission goals and discouraging those that do not. The sophistication of the reward system lies in its ability to discern not just the success of an action but its efficiency and relevance to the mission's objectives. The agent learns through this system to not only perform its tasks but to perform them well — to not just reach a destination but to do so with optimal use of resources.

Finally, the user interface is where the abstract world of algorithms and data transforms into something tangible and understandable. Through the interface, users can interact with the agent, offering directives, setting new objectives, or simply observing the fruits of the agent’s labor. This interface is more than a tool; it’s a lens through which the complexity of the agent's operations becomes accessible and the marvel of its learning process can be appreciated. It is here, in the harmonious interplay of agent actions and user inputs, that the simulation reveals its full potential as a tool not just for observing but for engaging with the frontiers of artificial intelligence in space exploration.

3.8 Controller logic and Execution

In the simulated environment of Unity, where the digital void is speckled with the virtual constructs of planets and stars, the spaceship agent operates as an embodiment of both machine learning theory and practical execution. Its actions within this realm are directed by a controller logic that serves as a translator, converting the complex decisions derived from neural networks into actual movements that propel the agent through space. This translation is not straightforward but requires an intricate mapping of neural network outputs, such as action probabilities or vector values, into the physical mechanics governed by Unity's physics engine.

The outputs of the neural network are contingent upon the extensive array of data the spaceship agent collects as it navigates the simulated cosmos. This data includes not just spatial coordinates or velocities but a rich tapestry of sensory inputs that paint a detailed picture of the environment. The controller logic must then interpret these outputs appropriately, recognizing, for instance, that a certain output value corresponds to the firing of thrusters for acceleration or the initiation of a sequence to collect stellar data. It must then articulate these actions within the constraints of Unity's physics-based system, where actions are manifest as forces and torques applied to the rigid body of the spaceship.

Real-time adjustments are a fundamental aspect of the spaceship's journey. As it courses through the simulated solar system, the agent must respond to an array of dynamic factors. It might need to alter its trajectory to harness the gravitational pull of a planet or adjust its speed to navigate an asteroid field with agility. These adjustments are made in real-time, reflecting a continuous loop of processing, decision-making, action, and feedback. Sensor data processed by the neural network informs decisions, which in turn are translated by the controller into adjustments of the spaceship's trajectory and speed.

The controller logic follows a sequence that begins with the input of sensor data, proceeds to the neural network's decision-making, translates these decisions into specific movements, and culminates in the execution of these movements within Unity.

This cycle is constant, with the results of each action being evaluated against the mission's objectives. For example, if an action results in a deviation from the planned path, the controller logic must ascertain whether this deviation is acceptable within the mission's parameters or whether corrective action is required. The outcomes of actions are then fed back as updated data to the neural network, which refines its policy based on the new information—information that now includes the results of previous decisions.

This feedback loop—where actions lead to outcomes, which inform future actions—is central to the reinforcement learning process. It allows the spaceship agent to not only react to the immediate environment but also to learn from its experiences, developing a more sophisticated understanding of how to navigate the complex and ever-changing space environment.

As the spaceship agent continues to interact with its surroundings, the controller logic ensures that the agent's movements are not only responses to immediate stimuli but also part of a strategic approach to fulfilling its mission objectives. The logic enables the spaceship to apply thrust when needed, to decelerate with precision, and to make navigational adjustments with finesse. It is within this procedural framework that the spaceship agent moves and learns, with each loop through the cycle sharpening its ability to navigate and operate within the vast digital expanse of the Unity simulation.

The controller logic's capacity to enact the neural network's intricate directives within the Unity environment hinges on a deep understanding of the physics that govern simulated space. Each decision output from the neural network encapsulates a potential action—be it a subtle shift in course to align with a distant moon or a burst of speed to escape the pull of a planet's gravity. Translating these outputs into Unity movements demands a precision akin to that of an orchestral conductor, where each movement and cue leads to the harmonious flow of the ensemble, here represented by the agent's trajectory through space.

To achieve this, the controller logic deciphers the neural network outputs, translating abstract values into definitive commands. When the network dictates an acceleration, the controller responds by calculating the exact force needed to be applied to the spaceship's thrusters, factoring in current speed, mass, and the counterforce of any gravitational bodies.

For turning maneuvers, it determines the torque required to rotate the spaceship along its axis, balancing the need for speed with the spacecraft's inertia and angular momentum. All these calculations are anchored in Unity's physics engine, ensuring the simulated movements are as realistic and physically accurate as possible.

The real-time nature of these adjustments is critical. Space is not a static environment; it is a symphony of moving parts, each influencing the other. The controller logic thus acts not only as a translator but also as a dynamic navigator, continuously recalibrating and readjusting the spaceship's course. This navigation is not a mere reaction to immediate stimuli but rather a complex interplay of current conditions, predictive modeling, and the agent's learned experiences. If a new asteroid field is detected, the controller must weigh the risk of immediate course alteration against the longer-term mission plan, always aiming to maintain the delicate balance between exploration and safety.

Furthermore, the controller logic functions within a feedback loop that epitomizes the essence of machine learning. It is within this loop that the agent's actions are not merely executed but also evaluated against their effectiveness in achieving mission objectives. Positive outcomes reinforce the neural network's current policy, while less successful actions prompt a reevaluation and adjustment of strategies. For example, if an intended acceleration consumes more fuel than anticipated, resulting in an efficiency penalty, the neural network takes this feedback, adjusts its policy, and the controller logic, in turn, will execute a more fuel-efficient acceleration on the next occasion.

This continuous cycle of execution and evaluation is what enables the agent to refine its behaviors over time. With each pass through the loop, the agent becomes more adept at interpreting its environment and selecting the optimal course of action. The feedback is essential; it is the soil in which the seeds of experience are sown and from which better, more informed actions sprout.

Within this second page of our exploration, we begin to see how the neural network's complex decisions are brought to life within the Unity simulation. The controller logic's role as the intermediary is paramount, bridging the gap between the theoretical world of machine learning and the practical realm of simulated space navigation.

It is here, in the meticulous execution of movements and the vigilant monitoring of their outcomes, that the agent hones its ability to maneuver through the celestial challenges of the simulation, becoming ever more skilled in the art of virtual spaceflight.

As our exploration of the simulated Unity environment concludes, we arrive at a crucial understanding: the agent's journey is an ongoing narrative of adaptation and learning. The controller logic is the linchpin in this narrative, converting the theoretical wisdom of the neural network into the practical actions that navigate the star-studded tapestry of the simulated cosmos. It orchestrates a ballet of forces and motions, where the agent, ever-responsive and increasingly adept, weaves through the intricacies of its environment with growing finesse.

This environment, a crucible of virtual space, poses an array of challenges, from gravitational puzzles to navigational conundrums, each demanding precision and acuity. The controller logic, drawing from the deep well of data and experience, directs the agent with decisions that are the product of continuous learning and meticulous calibration. The actions taken, whether a thruster burst to clear the shadow of a planet or a gentle glide along the periphery of an asteroid belt, are etched into the fabric of the simulation, each one feeding back into the neural network, informing the next cycle of decision-making.

The symphony of this complex system reaches its crescendo in the harmony of action and adaptation. The spaceship agent, once a mere occupant of the virtual universe, emerges as a seasoned explorer, capable of charting its course through the celestial dance with increasing autonomy and precision. In the vastness of this digital space, the agent stands as a testament to the power of simulation and machine learning, a beacon of the potential for artificial intelligence to extend beyond the confines of theory into the realm of tangible, directed action. With each pass through the feedback loop, the agent evolves, and with it, our understanding of what it means to navigate the final frontier, whether in simulation or, eventually, the reality beyond our atmosphere.

In summary, the spaceship agent's journey within the Unity simulation encapsulates the transformative power of machine learning, bridging the gap between abstract decision-making and concrete action.

CHAPTER 4

IMPLEMENTATION AND TESTING

4.1 Implementation

In the vast canvas of a Unity simulation, the CelestialBody script is the cornerstone for sculpting the cosmos. Here, celestial bodies are not static props, but entities governed by the invisible forces of gravity, each with their own orbits and trajectories. This script captures the essence of the physical laws that govern real-world celestial mechanics, translating them into the virtual world with a finesse that challenges the boundary between simulation and reality.

At the heart of the CelestialBody script is the careful definition of each body's physical properties: the radius that determines its scale in the virtual universe, and the surface gravity, a measure that dictates the gravitational pull it will exert on surrounding objects. These properties are more than mere numbers; they embody the celestial body's presence in the simulation, determining how other entities will navigate and interact with it. The initial velocity is a defining trait, setting each body into motion along a path that will evolve organically as the simulation unfolds.

The script goes on to intricately calculate the mass of the celestial body, using the surface gravity and radius in a formula that anchors the virtual entity in the established laws of physics. This mass is not static but is pivotal to the gravitational dance that ensues. It is the script's mass property that other celestial bodies in the simulation will sense and react to, each pull and push forming the choreography of a cosmic ballet.

This CelestialBody script also harnesses the power of Unity's Rigidbody component, a bridge between the scripted properties and Unity's physics engine. By tying the celestial body's mass to the Rigidbody, the script ensures that the virtual representation behaves as it would within the cosmos, subject to forces, collisions, and the relentless pull of gravity. The initial velocity becomes the starting push that sends the body hurtling through space, whether it's tracing the elegant ellipse of an orbit or hurtling through the void on a solitary trajectory.

But the script’s role doesn’t end at inception; it is the custodian of each body’s journey through the simulated expanse. Through methods like UpdateVelocity and UpdatePosition, the script actively adjusts each body's course, accounting for the myriad of gravitational influences exerted by other bodies. This results in a dynamic system where celestial bodies do not simply follow preordained paths but respond to the ever-shifting tableau of the simulation’s universe.

Adjusting the velocity with each frame, the script calculates the pull from neighboring celestial objects, updating the course with a responsiveness that ensures accuracy and realism. The UpdatePosition method takes these calculated changes in velocity and seamlessly translates them into movement, shifting the celestial body’s position within the Unity world, maintaining the celestial waltz’s fluidity and grace.

The CelestialBody script is a masterclass in simulation, a piece of code that captures the essence of orbital mechanics and celestial dynamics. As the foundation for a system of planets, moons, and stars, it ensures that each body moves according to the same physical principles that guide our own universe. It’s a blend of physics, mathematics, and computer science that creates a living, breathing cosmos within the digital confines of Unity. This script does not merely create a backdrop for gameplay or visualization; it weaves the very fabric of a universe in motion, setting the stage for explorations, adventures, and discoveries that mirror the grandeur of the cosmos we inhabit.

Shifting focus from the CelestialBody script, we delve into the crucial role of the Controller Logic script within the Unity environment. This script is indispensable as it bridges the gap between the abstract decision-making of neural networks and the concrete, physical actions within the simulation. It effectively translates commands from high-level AI decisions into actionable, physics-based movements that Unity's engine can process and execute.

The Controller Logic script functions as a translator, converting outputs from the neural network—ranging from simple commands like "move forward" to complex navigational maneuvers—into precise actions. It maps these decisions onto the game entities' physical attributes, applying the necessary forces, torques, and adjustments required to enact these decisions in the game world.

Central to its operation is a robust feedback loop that integrates real-time data from the environment. This data informs the neural networks, aiding them in refining their strategic outputs. Once these decisions are made, the Controller Logic script executes these actions physically, adjusting continuously to the dynamic conditions and feedback from the environment. This loop ensures that the entities react and adapt effectively to their surroundings, maintaining realism and strategic behavior.

Furthermore, the script is designed to manage and mitigate potential risks or errors in decision-making. It includes mechanisms to prevent outcomes that could disrupt gameplay, such as collisions or illogical actions, by enforcing checks and constraints that align actions with the game's physical laws.

Continuing our exploration of essential scripts within the Unity simulation environment, let's turn our attention to the Ship script, a dynamic component crucial for managing spacecraft within game settings. This script not only controls the basic movements of a ship but also integrates complex functionalities such as entering and exiting the ship, managing its operational state, and handling its interactions with the game environment.

The Ship script serves as the central management system for spacecraft within Unity games. It enables the ship to navigate through the virtual cosmos with a series of propulsion and steering mechanisms. These mechanisms are controlled through player inputs, which dictate commands such as acceleration, deceleration, and directional changes. The script translates these inputs into physical responses, adjusting the ship’s Rigidbody component to reflect changes in speed and orientation according to the laws of physics defined within the Unity environment.

An integral part of this script is its ability to handle interactions with other game elements, such as celestial bodies, space stations, or enemy crafts. It includes collision detection systems that not only prevent the ship from undergoing uncontrolled damage when encountering other objects but also trigger specific game events like battles, docking procedures, or resource gathering. These interactions are managed through a series of triggers and events within the script, ensuring that each encounter is handled according to the game’s rules and storyline.

Additionally, the Ship script manages the entry and exit mechanisms for the spacecraft, providing players with the ability to leave the ship to explore planets, space stations, or engage in external repairs. This functionality is crucial for games that feature expansive exploration or survival elements. The script coordinates the transition between the ship’s interior and the external environment, maintaining continuity in player experience and game narrative. It ensures that all environmental conditions, such as gravity and atmosphere, are adjusted to provide a seamless transition for the player.

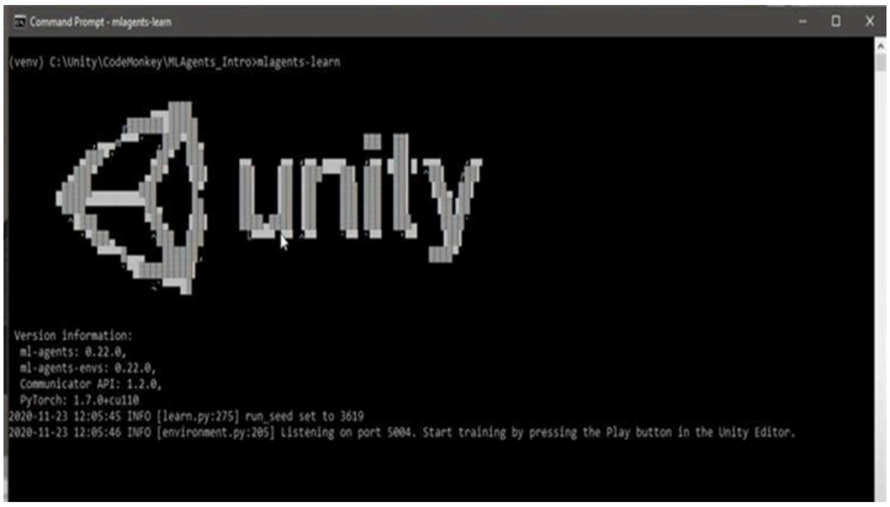
Furthermore, the Ship script is designed to be highly customizable, allowing developers to adjust various parameters such as speed limits, fuel consumption, and handling characteristics to fit different types of gameplay. It can be modified to accommodate different levels of player skill, with settings that can simplify controls for casual players or provide detailed and complex control systems for more experienced users.

Delving further into Unity's scripting capabilities that enhance player interaction and environmental engagement, we examine the HatchButton and Interactable scripts. These scripts are pivotal in creating immersive and interactive gameplay, especially within games that feature complex environments and vehicles like spacecraft.

The HatchButton script exemplifies specialized interaction within the game environment. It is a derivative of the broader Interactable class, tailored specifically to manage the operation of a spacecraft's hatch. This script enables dynamic interaction, allowing players to open or close the hatch based on the game's context and their input. The functionality of the HatchButton is directly tied to the state of the ship's hatch; it checks whether the hatch is open or closed and updates the display message accordingly for the player. This could be a simple prompt like "Press F to open hatch," reflecting the immediate action the player can take. This script not only adds a layer of realism by simulating physical components of the spacecraft but also enhances player engagement by integrating them into the minutiae of spacecraft operation.

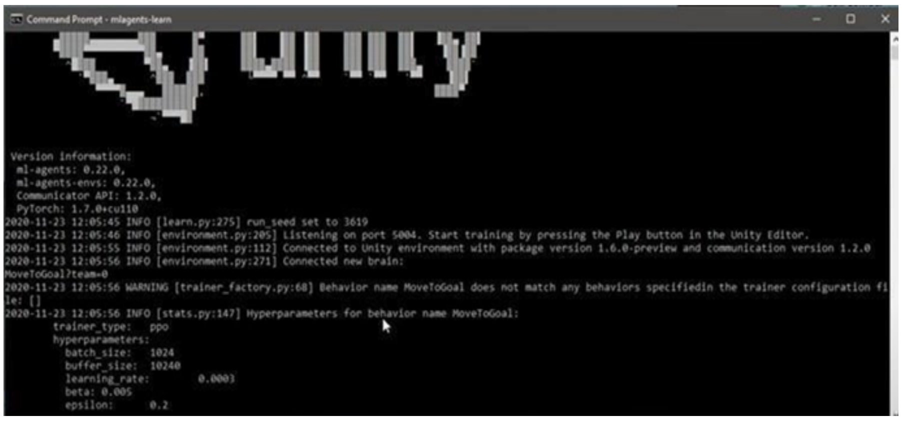
On the broader spectrum, the Interactable script serves as a foundation for creating interactive objects within the Unity environment. It's designed to be highly versatile, accommodating a wide range of interactive possibilities. This script manages how objects can interact with players when they enter a specific zone, typically defined by a collider.

4.2 Training and Testing Analysis



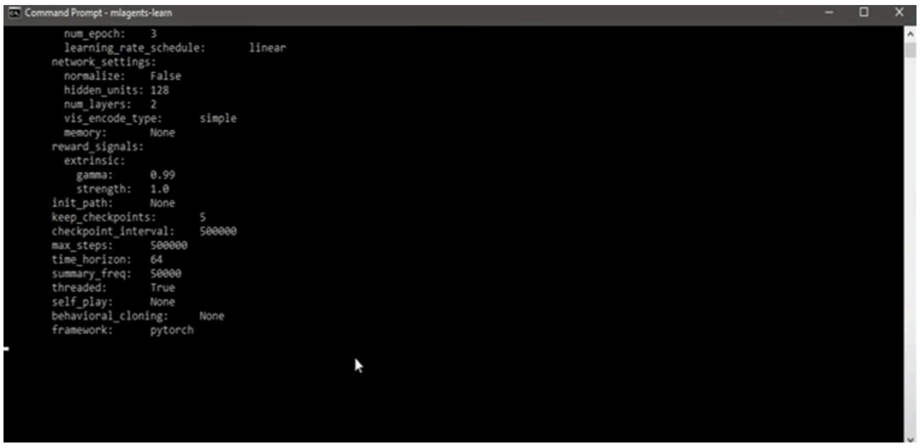
```
(venv) C:\Unity\CodeMonkey\MLAgents_Intro>mlagents-learn

 Version information:
  ml-agents: 0.22.0,
  ml-agents-envs: 0.22.0,
  Communicator API: 1.2.0,
  PyTorch: 1.7.0+cu110
2020-11-23 12:05:45 INFO [learn.py:275] run_seed set to 3619
2020-11-23 12:05:46 INFO [environment.py:205] Listening on port 5004. Start training by pressing the Play button in the Unity Editor.
```

Figure 4.2.1 Setting up Training Configuration



```
Version information:
 ml-agents: 0.22.0,
 ml-agents-envs: 0.22.0,
 Communicator API: 1.2.0,
 PyTorch: 1.7.0+cu110
2020-11-23 12:05:45 INFO [learn.py:275] run_seed set to 3619
2020-11-23 12:05:46 INFO [environment.py:205] Listening on port 5004. Start training by pressing the Play button in the Unity Editor.
2020-11-23 12:05:55 INFO [environment.py:112] Connected to Unity environment with package version 1.6.0-preview and communication version 1.2.0
2020-11-23 12:05:56 INFO [environment.py:271] Connected new brain:
MoveToGoal?team=0
2020-11-23 12:05:56 WARNING [trainer_factory.py:68] Behavior name MoveToGoal does not match any behaviors specifiedin the trainer configuration fi
le: []
2020-11-23 12:05:56 INFO [stats.py:147] Hyperparameters for behavior name MoveToGoal:
        trainer_type:   ppo
        hyperparameters:
          batch_size:   1024
          buffer_size:  10240
          learning_rate:        0.0003
          beta: 0.005
          epsilon:      0.2
```

Figure 4.2.2 Training Parameters



```
          num_epoch:    3
          learning_rate_schedule:       linear
        network_settings:
          normalize:    False
          hidden_units: 128
          num_layers:   2
          vis_encode_type:      simple
          memory:       None
        reward_signals:
          extrinsic:
            gamma:      0.99
            strength:   1.0
        init_path:      None
        keep_checkpoints:       5
        checkpoint_interval:    500000
        max_steps:      500000
        time_horizon:   64
        summary_freq:   50000
        threaded:       True
        self_play:      None
        behavioral_cloning:     None
        framework:      pytorch
```

Figure 4.2.3 Training and Testing Configuration

Unity's ML-Agents provides a dynamic training environment, where each new episode can be a fresh start with randomized initial conditions. This injects a level of unpredictability into the training, forcing the agent to generalize its policy rather than memorizing specific paths to success. Actions are delivered to the agent through an array, which could represent forces applied to move the agent or discrete choices in a strategy game. This array is central to the learning process as the neural network learns which actions lead to higher rewards and are hence more desirable to repeat in future steps.

In the sphere of Unity's ML-Agents, the convergence of data collection, neural network adjustment, and the reward system forms a triad that underpins the evolution of intelligent agents. These agents, akin to sentient beings in a digital ecosystem, engage with their environment in a relentless quest for optimization, continually refining their approach to the tasks at hand.

At the heart of this process is the reward system—a carefully calibrated guide that shapes the learning trajectory of each agent. Like the invisible hand in economics, it invisibly steers the agents, rewarding them for desirable outcomes and imposing penalties for errors or inefficiencies. This system of incentives and disincentives is pivotal in training the agents to align their objectives with those set by their digital universe.

Complementing the reward system is the neural network, the bedrock of the agent’s decision-making capabilities. Each experience, each interaction of the agent with its environment, serves as a data point that feeds into this network. Through repeated episodes—each a fresh canvas—the neural network processes the data, learns patterns, and adapts its parameters. The outcome is an ever-evolving policy, a set of internal directives that become increasingly adept at navigating the complexities of the environment.

A critical aspect of this learning is the introduction of variance and uncertainty. Randomizing starting conditions, altering environmental factors, and shifting objectives prevent the agents from falling into the trap of rote memorization. Instead, they are compelled to understand, to infer, and to strategize. This generalization is crucial for developing robust behaviors that are effective across a spectrum of scenarios, rather than being narrowly tailored to a single set of conditions.

As the agents operate within their digital worlds, they are not solitary entities. Their actions are often influenced by the presence and movements of other dynamic elements within the environment, such as targets or obstacles. Interactions with these elements are critical learning moments—each collision, avoidance, or successful goal attainment provides invaluable data that feeds back into the learning loop, further refining the agent’s behavior.

This continuous cycle of action and reaction is orchestrated by the Unity engine, where the physics of movement, the rendering of visuals, and the capture of data are synchronized to create a holistic learning environment. Agents must navigate, strategize, and adapt in real-time, their every move underpinned by the complex interplay of physics-based simulation and the neural network's ever-evolving logic.

In this environment, the training process transcends simple task execution. It becomes an intricate dance of strategy, where agents equipped with the wisdom of both their human trainers and their own trial-and-error experiences navigate towards their objectives. They learn to chart a course not just to reach a target but to do so with efficiency, precision, and a nuanced understanding of the multifaceted world they inhabit.

As we consider the culmination of an agent’s training within the Unity ML-Agents framework, the creation of a neural model—or the agent's 'brain'—emerges as a monumental task. This model, a digital edifice constructed from layers of computational neurons, becomes the cornerstone of the agent’s ability to navigate through space and make decisions. For a spaceship agent tasked with the exploration of planets, the brain model is the culmination of countless interactions, decisions, and adjustments, each informed by the agent's encounters with the cosmos.

The spaceship's trajectory towards a planet is a calculated ballet of physics and decision-making, orchestrated by its neural brain. Through the lens of reinforcement learning, each successful orbit, smooth navigation, or precise landing is the result of positive reinforcement. As the spaceship avoids asteroids and deftly maneuvers through space, it gathers positive rewards, encouraging the neural network to strengthen the associations between certain actions and outcomes.

Conversely, when the spaceship falters—when an asteroid collision occurs or a misstep leads to a crash—the negative feedback loops back into the system. The spaceship's neural brain registers these as actions to avoid, tuning the parameters within its layers to decrease the likelihood of repeating such mistakes. Negative rewards serve a crucial purpose, fine-tuning the agent's policy by penalizing actions that lead away from the objective.

The spaceship's journey towards a planet is a test of its accumulated knowledge and the robustness of its neural model. As it approaches its destination, the intricate dance between positive and negative rewards becomes a testament to the efficacy of its learning. The spaceship, leveraging its reinforced behaviors, must execute a series of precise movements to enter the planet's orbit and prepare for descent. It's a challenging task, requiring the agent to synthesize everything it has learned to successfully navigate and land.

Once the spaceship touches down on the planetary surface, a new phase of exploration begins. The brain model, now tried and tested in the crucible of space travel, must adapt to the terrestrial challenges. Positive rewards continue to guide the spaceship, reinforcing actions like smooth navigation over the terrain and effective data collection. The exploration phase is as much a part of the spaceship's learning journey as the interstellar travel that preceded it.

In every action and decision, the spaceship's brain model seeks to maximize the cumulative reward, a numerical embodiment of its success. The agent's policy, a strategic guide formed through rigorous training, leads the spaceship towards the most favorable actions. It's a dynamic process, with the spaceship's brain continually evolving, learning not just to react but to predict and strategize, weaving through the vast tapestry of space to the rhythm of positive reinforcement and the caution of its penalties.

The journey of the spaceship agent, from launch to landing and exploration, is a mirror to the potential of Unity ML-Agents. It demonstrates the power of a well-trained neural model to navigate complex environments and achieve goals that are defined by the intricate balance of reward-driven learning—a testament to the transformative potential of machine learning in the virtual expanse.

CHAPTER 5

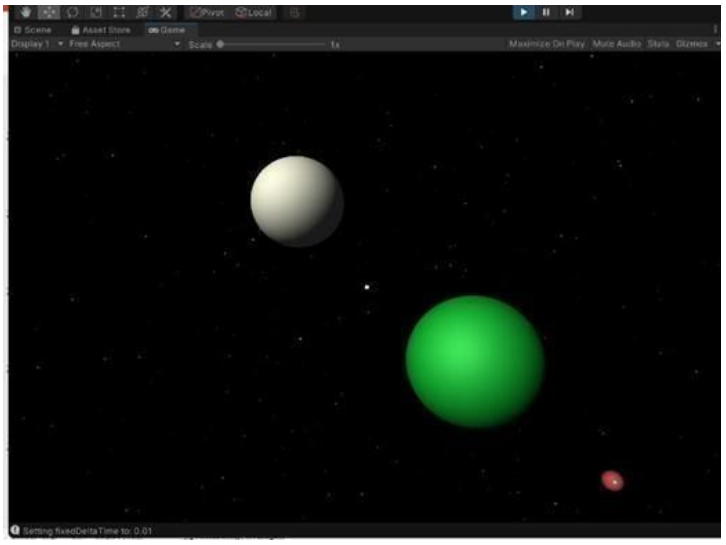
RESULTS AND DISCUSSIONS

5.1 Results

The results of our Solar System Adventure project reveal a significant breakthrough in AI-driven space exploration. Utilizing deep reinforcement learning, the ML-agent spaceship adeptly maneuvered through the cosmos, making strategic decisions to avoid obstacles and reach targets. Incorrect choices triggered negative rewards, effectively steering the agent toward successful outcomes. The agent’s ability to adapt to complex environments was showcased with high precision, reflecting the potential for AI in autonomous navigation. Our findings highlight deep reinforcement learning's role in advancing cognitive computing and its applications in simulating intricate space scenarios.

System Interaction Scene:

The dynamic environment in which the spacecraft agent interacts with the simulated solar system is captured in this image. It draws attention to how carefully crafted celestial bodies are, symbolizing the complex environments the agent must work through. The scenario lays the groundwork for showcasing the agent's capacity to take in and adjust to the intricacies of space.



Setting fixedDeltaTime to: 0.01

Figure 5.1.1: Working Environment

Planetary Tracking and Navigation:

Here, we observe the spacecraft tracking and locating planets in the virtual solar system using deep reinforcement learning. The spaceship's route is visually represented by the dashed line and velocity indications as it determines the best course to take in order to reach its destination, demonstrating the usefulness of deep learning in autonomous space navigation.

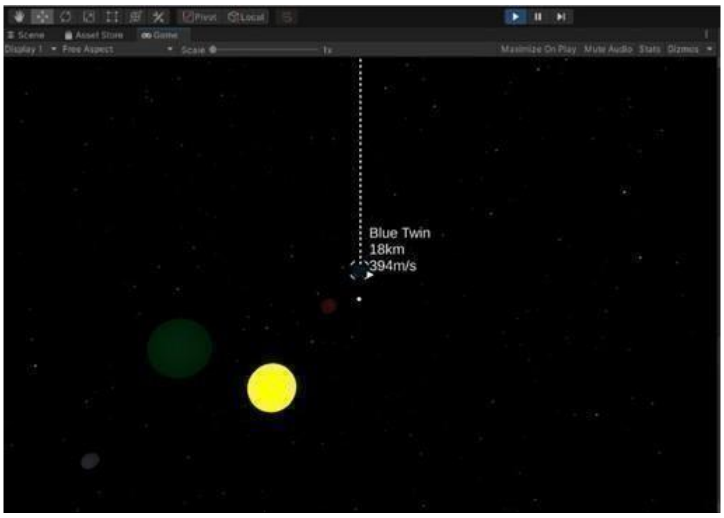


Figure 5.1.2: Planetary Tracking

Hatch Operation:

The hatch control system, one of the spaceship's primary interactive features, is captured in the picture. A feature of the user interface that allows physical interaction with the spaceship and serves as a failsafe as well as an extra layer of immersion for operators within the simulated environment is the ability to open or close the hatch by pressing the letter "F."

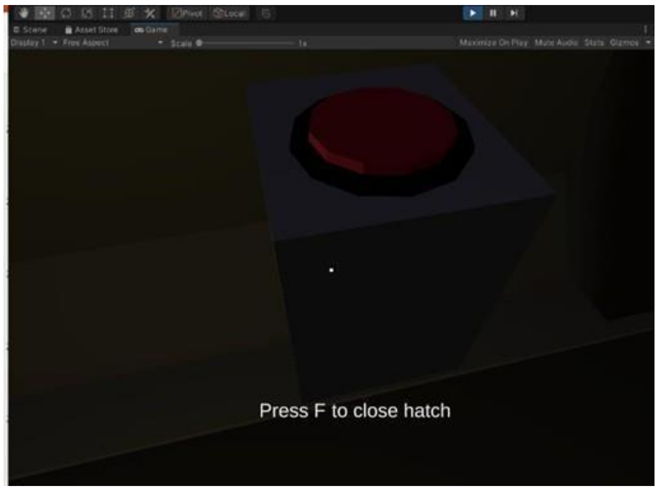


Figure 5.1.3: Hatch Opening and Closing

Space Exploration Commences:

The spaceship is prepared to begin its adventure as the door opens to reveal the immensity of space. At this critical point, the simulation moves from preparation to active exploration, with the DRL algorithm directing the agent's travel through the stars and its acquired behaviors serving as its driving forces.

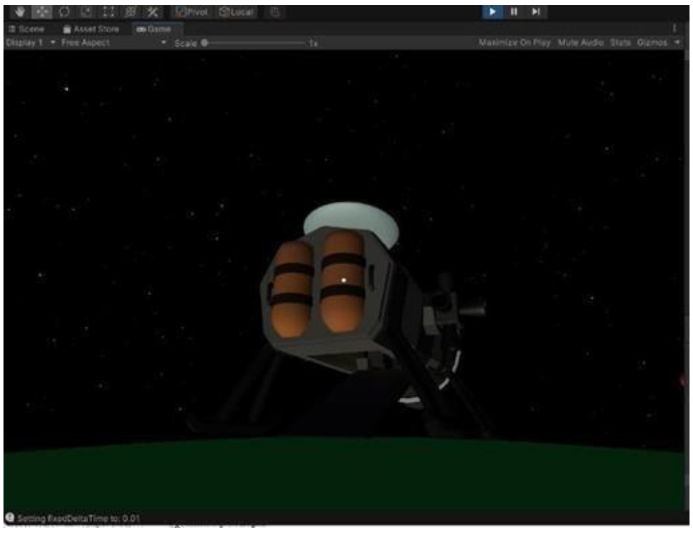


Figure 5.1.4: Space Exploration

Navigational Decision-Making:

This scene demonstrates the crucial situations in which the ML agent-controlled spaceship must make accurate navigational decisions. The spaceship is in danger of colliding with asteroids if this is not true, as evidenced by its closeness to the Purple Giant. This emphasizes how important it is for the agent to make the right choices in order to stay out of trouble and finish the operation.

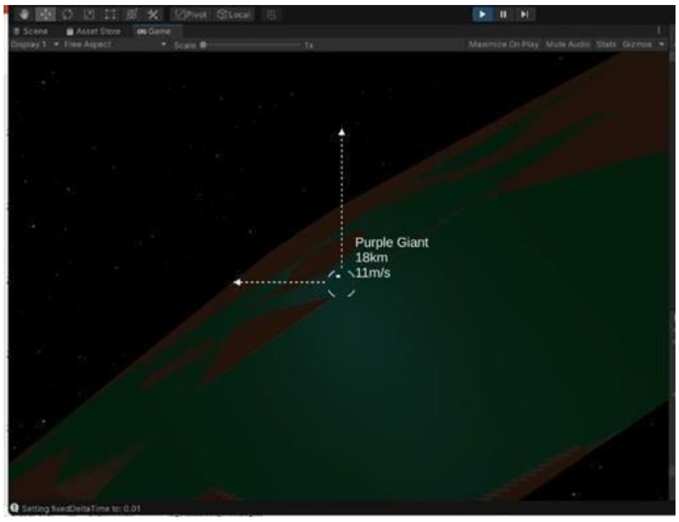
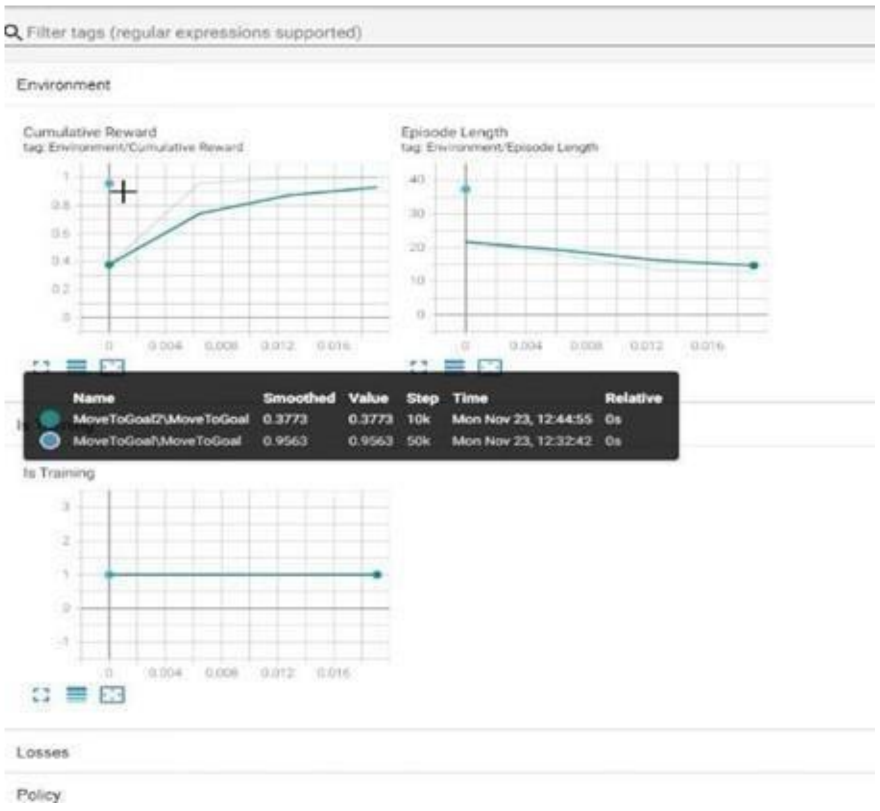


Figure 5.1.5: Collision with Asteroid

Visualization:

The data from TensorBoard shows that the agent is effectively learning over time. The agent appears to be continuously attaining more rewards as it gains knowledge from its surroundings, as seen by the increasing trend in the Cumulative Reward chart. The trending Episode Length chart indicates that the agent is achieving its goals more quickly as the episodes go by. The agent's performance is generally getting better, which is a sign that it is capable of learning.


Filter tags (regular expressions supported)
Environment
Cumulative Reward
tag: Environment/Cumulative Reward
Episode Length
tag: Environment/Episode Length
Is Training
Losses
Policy

	Name	Smoothed	Value	Step	Time	Relative
●	MoveToGoal2\MoveToGoal	0.3773	0.3773	10k	Mon Nov 23, 12:44:55	0s
●	MoveToGoal\MoveToGoal	0.9563	0.9563	50k	Mon Nov 23, 12:32:42	0s

Figure 5.1.6: Tensorboard Visualization

CHAPTER 6

CONCLUSION AND FUTURE ENHANCEMENTS

6.1 Future Scope in Project

Figure 6.1.1: Reimagining The Future

The completion of the Solar System Adventure project represents a significant advancement in the field of artificial intelligence (AI) and its relationship to space travel. The research has painstakingly illustrated the capability of an AI agent—a virtual spaceship—in negotiating the intricacies of a simulated solar system environment through the application of deep reinforcement learning (DRL). The agent's ability to make decisions, avoid obstacles, and acquire targets is not just a theoretical exercise; rather, it represents a potential model for future uses in both terrestrial and interplanetary environments.

Conclusion

The spacecraft agent can now learn and adapt on its own thanks to our successful use of machine learning capabilities in a virtual environment. The spacecraft agent has demonstrated during the project that it is capable of making calculated navigational judgements, planning the best possible route, and modifying its behaviour to maximise mission success. The carefully designed celestial bodies and dynamic interaction scene in the simulation's complex structure provide an ideal environment for the AI to show off its capabilities.

Future Enhancements:

The future implications of this research are numerous and extensive. Some of the potential improvements and additions that show promise are as follows:

1. **More Complicated:** It is possible to incorporate a wider range of astronomical events, including as comets, asteroids, and changeable star conditions, into the existing model. The agent's comprehension and navigational abilities might be enhanced by adding further complications like micrometeoroid strikes, solar radiation pressure, and gravitational slingshots.

2. **Integrating Real-World Data:** Enhancing the simulation with real-world data from satellite agencies could significantly increase the model's precision and applicability. This would make it easier to train the AI with situations that more closely resemble real-world space conditions.

3. **Intelligent Probes:** The project might be a first step towards creating fully autonomous probes that can travel deep into space. With their sophisticated artificial intelligence, the probes may be able to perform tests autonomously, evaluate celestial bodies, and make judgements about their navigation in real time.

4. **Mission Planning and Astronaut Training:** Astronauts might use the simulation as an advanced training tool to assist them become ready for a variety of space mission circumstances. It might also help with mission preparation by enabling researchers to test and simulate various exploration and navigational tactics.

5. **Interdisciplinary Collaboration:** Consulting with specialists in robotics, aerospace engineering, and astronomy may improve the adaptability and resilience of the AI model, paving the way for more complex simulations and ultimately practical space applications.

6. **Integration with Robotic Systems:** By putting the DRL model into practice in real robots intended for space exploration, on-ground testing may be made possible. This might result in iterative algorithmic improvements based on real performance data.

7. **Teaching Resources:** The simulation has the potential to grow into an instructional tool that teaches enthusiasts and students about the difficulties of space flight and how artificial intelligence can be used to overcome those difficulties.

8. **Multi-agent systems and scalability:** Extending the project to encompass numerous agents may replicate fleet operations, wherein collaboration and rivalry among diverse AI-managed spacecraft result in the emergence of novel behaviours and tactics.

9. **Repercussions in the Long Run and Ethics:** A wider conversation about the morality of AI in space travel is sparked by the initiative. It raises questions about the long-term viability of AI-guided space travel and creates opportunities for addressing issues like space debris control.

To sum up, the Solar System Adventure project is evidence of how machine learning can revolutionise how we approach space travel. With AI acting as our guide and comrade, the future improvements envisioned here will allow us to fully use this technology, pushing beyond the limits of our present capabilities and exploring the expanse of space. With every advancement, we go closer to a time when AI is an essential component of our quest for knowledge beyond our current understanding, not only a tool.

REFERENCES

[1] Lillicrap, Timothy P., Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. "Continuous control with deep reinforcement learning." arXiv preprint arXiv:1509.02971 (2015).

[2] Schulman, John, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. "Proximal policy optimization algorithms." arXiv preprint arXiv:1707.06347 (2017).

[3] Mnih, Volodymyr, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. "Playing atari with deep reinforcement learning." arXiv preprint arXiv:1312.5602 (2013).

[4] Mnih, Volodymyr, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. "Asynchronous methods for deep reinforcement learning." In International conference on machine learning, pp. 1928- 1937. PMLR, 2016.

[5] Mnih, Volodymyr, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves et al. "Human-level control through deep reinforcement learning." nature 518, no. 7540 (2015): 529-533

[6] Wang, Xu, Sen Wang, Xingxing Liang, Dawei Zhao, Jincai Huang, Xin Xu, Bin Dai, and Qiguang Miao. "Deep reinforcement learning: A survey." IEEE Transactions on Neural Networks and Learning Systems (2022).

[7] Schulman, John, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. "Trust region policy optimization." In International conference on machine learning, pp. 1889-1897. PMLR, 2015.

[8] Zhang, Qi, Tao Du, and Changzheng Tian. "Self-driving scale car trained by deep reinforcement learning." arXiv preprint arXiv:1909.03467 (2019).

[9] Fujimoto, Scott, Herke Hoof, and David Meger. "Addressing function approximation error in actor-critic methods." In International conference on machine learning, pp. 1587-1596. PMLR, 2018.

[10] Schulman, John, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. "High-dimensional continuous control using generalized advantage estimation." arXiv preprint arXiv:1506.02438 (2015).

[11] Arulkumaran, Kai, Marc Peter Deisenroth, Miles Brundage, and Anil Anthony Bharath. "A brief survey of deep reinforcement learning." arXiv preprint arXiv:1708.05866 (2017).

[12] Reuer, Kevin, Jonas Landgraf, Thomas Fösel, James O'Sullivan, Liberto Beltrán, Abdulkadir Akin, Graham J. Norris et al. "Realizing a deep reinforcement learning agent for real-time quantum feedback." Nature Communications 14, no. 1 (2023): 7138.
[13] Zolfagharian, Amirhossein, Manel Abdellatif, Lionel C. Briand, Mojtaba Bagherzadeh, and S. Ramesh. "A search-based testing approach for deep reinforcement learning agents." IEEE Transactions on Software Engineering (2023).
[14] Ana, Afina Anfa, Vitradisa Pratama, Agus Sukoco, Ary Setijadi Prihatmanto, Rin Rin Nurmalasari, and Agus Juhana. "ML-Agents in Kart Racing Game using Microgame." In 2023 17th International Conference on Telecommunication Systems, Services, and Applications (TSSA), pp. 1-6. IEEE, 2023.
[15] Balloni, Emanuele, Marco Mameli, Adriano Mancini, and Primo Zingaretti. "Deep Reinforced Navigation of Agents in 2D Platform Video Games." In Computer Graphics International Conference, pp. 288-308. Cham: Springer Nature Switzerland, 2023.
[16] Radwan, Mahmoud Osama, Ahmed Ahmed Hesham Sedky, and Khaled Mohammed Mahar. "Obstacles avoidance of self-driving vehicle using deep reinforcement learning." In 2021 31st International Conference on Computer Theory and Applications (ICCTA), pp. 215- 222. IEEE, 2021.
[17] Gil, Guilherme, Joao Rosas, and Luis Brito Palma. "Specification and development of dynamic systems and controllers based on game engines." In APCA International Conference on Automatic Control and Soft Computing, pp. 285-296. Cham: Springer International Publishing, 2022.
[18] Juliani, Arthur, Vincent-Pierre Berges, Ervin Teng, Andrew Cohen, Jonathan Harper, Chris Elion, Chris Goy et al. "Unity: A general platform for intelligent agents." arXiv preprint arXiv:1809.02627 (2018).
[19] Li, Hui. "Design and implement of soccer player AI training system using unity ML-agents." In CIBDA 2022; 3rd International Conference on Computer Information and Big Data Applications, pp. 1-4. VDE, 2022.
[20] Komrusch, Steve, and Henry Minsky. "Symbolic guidance for constructivist learning by neural model." In International Workshop on Self-Supervised Learning, pp. 63-76. PMLR, 2022.
[21] Pernas-Álvarez, Javier, and Diego Crespo-Pereira. "Open-source 3D discrete event simulator based on the game engine unity." Journal of Simulation (2024): 1-17.

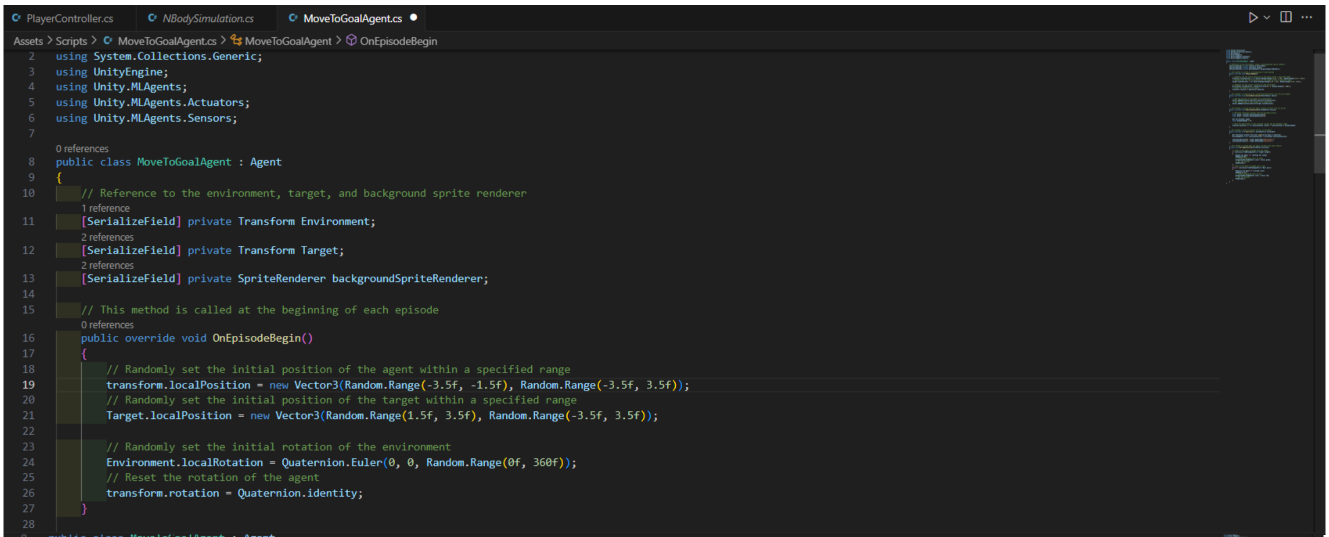
APPENDIX A
CODING AND SCREENSHOTS

SOURCE CODE:

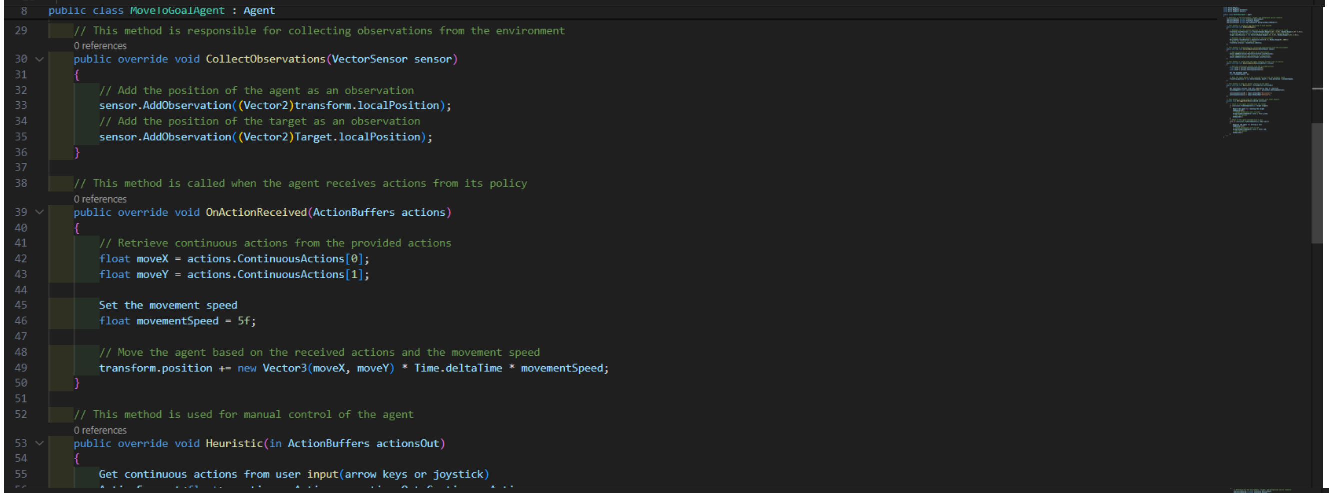
```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using Unity.MLAgents;
using Unity.MLAgents.Actuators;
using Unity.MLAgents.Sensors;

public class MoveToGoalAgent : Agent
{
   // Reference to the environment, target, and background sprite renderer
   [SerializeField] private Transform Environment;
   [SerializeField] private Transform Target;
   [SerializeField] private SpriteRenderer backgroundSpriteRenderer;

   // This method is called at the beginning of each episode
   public override void OnEpisodeBegin()
   {
      // Randomly set the initial position of the agent within a specified range
      transform.localPosition = new Vector3(Random.Range(-3.5f, -1.5f),
Random.Range(-3.5f, 3.5f));
      // Randomly set the initial position of the target within a specified range
      Target.localPosition = new Vector3(Random.Range(1.5f, 3.5f), Random.Range(-
3.5f, 3.5f));

      // Randomly set the initial rotation of the environment
      Environment.localRotation = Quaternion.Euler(0, 0, Random.Range(0f, 360f));
      // Reset the rotation of the agent
      transform.rotation = Quaternion.identity;
   }

   // This method is responsible for collecting observations from the environment
   public override void CollectObservations(VectorSensor sensor)
   {
      // Add the position of the agent as an observation
      sensor.AddObservation((Vector2)transform.localPosition);
      // Add the position of the target as an observation
      sensor.AddObservation((Vector2)Target.localPosition);
   }

   // This method is called when the agent receives actions from its policy
```

```
    public override void OnActionReceived(ActionBuffers actions)
    {
        // Retrieve continuous actions from the provided actions
        float moveX = actions.ContinuousActions[0];
        float moveY = actions.ContinuousActions[1];

        Set the movement speed
        float movementSpeed = 5f;

        // Move the agent based on the received actions and the movement speed
        transform.position += new Vector3(moveX, moveY) * Time.deltaTime *
movementSpeed;
    }



    // This method is used for manual control of the agent
    public override void Heuristic(in ActionBuffers actionsOut)
    {
        Get continuous actions from user input(arrow keys or joystick)
        ActionSegment<float> continuousActions = actionsOut.ContinuousActions;

        continuousActions[0] = Input.GetAxisRaw("Horizontal");
        continuousActions[1] = Input.GetAxisRaw("Vertical");
    }

    // This method is called when the agent collides with other objects
    private void OnTriggerEnter2D(Collider2D collision)
    {
        // Check if the agent collided with the target
        if (collision.TryGetComponent(out Target target))
        {
            Reward the agent for reaching the target
            AddReward(10f);
            // Change background color to green
            backgroundSpriteRenderer.color = Color.green;
            // End the episode
            EndEpisode();
        }
        // Check if the agent collided with a wall
        else if (collision.TryGetComponent(out Wall wall))
        {
            Penalize the agent for hitting a wall
            AddReward(-2f);
            // Change background color to red
            backgroundSpriteRenderer.color = Color.red;
            // End the episode
            EndEpisode();
        }
    }
}
```

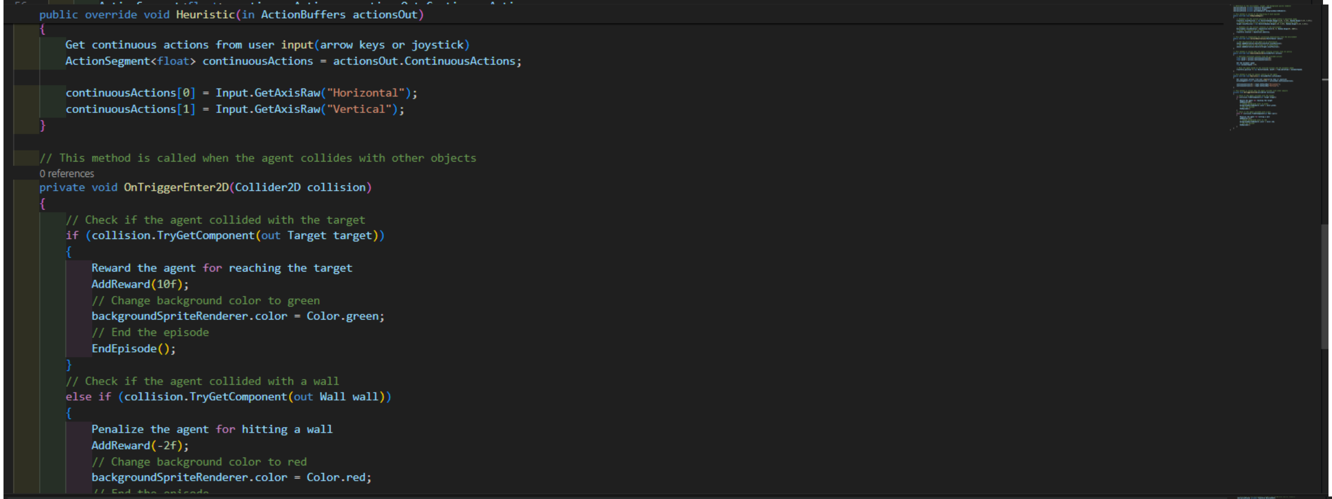
SCREENSHOTS:


PlayerController.cs
NBodySimulation.cs
MoveToGoalAgent.cs
Assets > Scripts > MoveToGoalAgent.cs > MoveToGoalAgent > OnEpisodeBegin

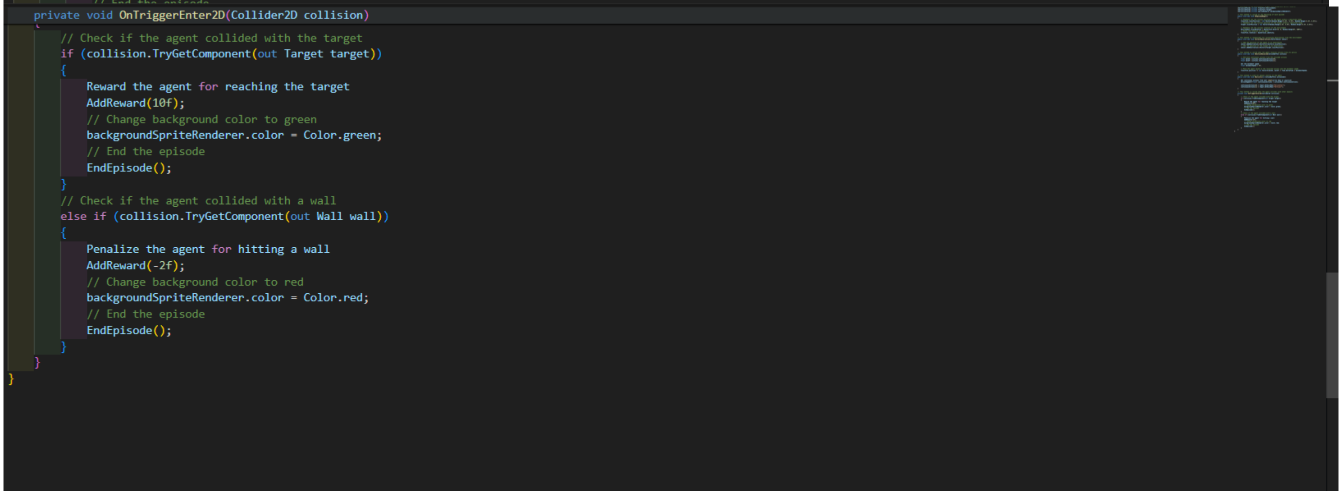
```
using System.Collections.Generic;
using UnityEngine;
using Unity.MLAgents;
using Unity.MLAgents.Actuators;
using Unity.MLAgents.Sensors;

0 references
public class MoveToGoalAgent : Agent
{
    // Reference to the environment, target, and background sprite renderer
    1 reference
    [SerializeField] private Transform Environment;
    2 references
    [SerializeField] private Transform Target;
    2 references
    [SerializeField] private SpriteRenderer backgroundSpriteRenderer;

    // This method is called at the beginning of each episode
    0 references
    public override void OnEpisodeBegin()
    {
        // Randomly set the initial position of the agent within a specified range
        transform.localPosition = new Vector3(Random.Range(-3.5f, -1.5f), Random.Range(-3.5f, 3.5f));
        // Randomly set the initial position of the target within a specified range
        Target.localPosition = new Vector3(Random.Range(1.5f, 3.5f), Random.Range(-3.5f, 3.5f));

        // Randomly set the initial rotation of the environment
        Environment.localRotation = Quaternion.Euler(0, 0, Random.Range(0f, 360f));
        // Reset the rotation of the agent
        transform.rotation = Quaternion.identity;
    }
```



```
public class MoveToGoalAgent : Agent
    // This method is responsible for collecting observations from the environment
    0 references
    public override void CollectObservations(VectorSensor sensor)
    {
        // Add the position of the agent as an observation
        sensor.AddObservation((Vector2)transform.localPosition);
        // Add the position of the target as an observation
        sensor.AddObservation((Vector2)Target.localPosition);
    }

    // This method is called when the agent receives actions from its policy
    0 references
    public override void OnActionReceived(ActionBuffers actions)
    {
        // Retrieve continuous actions from the provided actions
        float moveX = actions.ContinuousActions[0];
        float moveY = actions.ContinuousActions[1];

        Set the movement speed
        float movementSpeed = 5f;

        // Move the agent based on the received actions and the movement speed
        transform.position += new Vector3(moveX, moveY) * Time.deltaTime * movementSpeed;
    }

    // This method is used for manual control of the agent
    0 references
    public override void Heuristic(in ActionBuffers actionsOut)
    {
        Get continuous actions from user input(arrow keys or joystick)
```

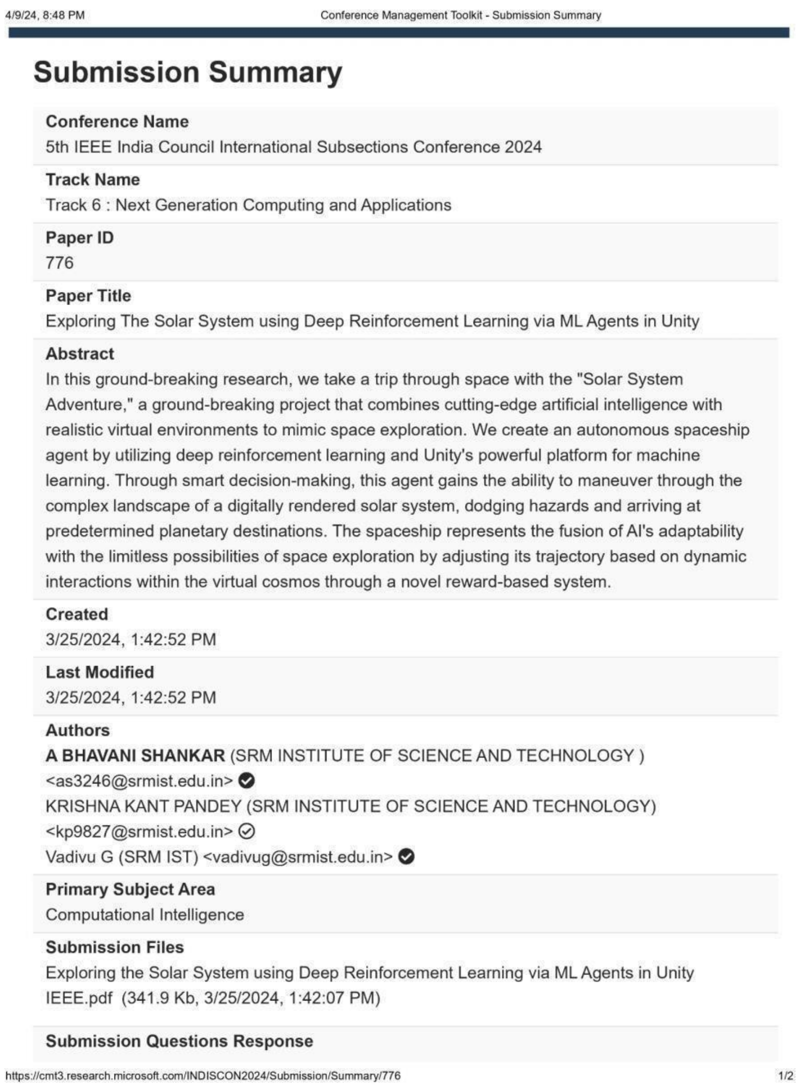


```
public override void Heuristic(in ActionBuffers actionsOut)
{
    Get continuous actions from user input(arrow keys or joystick)
    ActionSegment<float> continuousActions = actionsOut.ContinuousActions;

    continuousActions[0] = Input.GetAxisRaw("Horizontal");
    continuousActions[1] = Input.GetAxisRaw("Vertical");
}

// This method is called when the agent collides with other objects
0 references
private void OnTriggerEnter2D(Collider2D collision)
{
    // Check if the agent collided with the target
    if (collision.TryGetComponent(out Target target))
    {
        Reward the agent for reaching the target
        AddReward(10f);
        // Change background color to green
        backgroundSpriteRenderer.color = Color.green;
        // End the episode
        EndEpisode();
    }
    // Check if the agent collided with a wall
    else if (collision.TryGetComponent(out Wall wall))
    {
        Penalize the agent for hitting a wall
        AddReward(-2f);
        // Change background color to red
        backgroundSpriteRenderer.color = Color.red;
```



```
private void OnTriggerEnter2D(Collider2D collision)
    // Check if the agent collided with the target
    if (collision.TryGetComponent(out Target target))
    {
        Reward the agent for reaching the target
        AddReward(10f);
        // Change background color to green
        backgroundSpriteRenderer.color = Color.green;
        // End the episode
        EndEpisode();
    }
    // Check if the agent collided with a wall
    else if (collision.TryGetComponent(out Wall wall))
    {
        Penalize the agent for hitting a wall
        AddReward(-2f);
        // Change background color to red
        backgroundSpriteRenderer.color = Color.red;
        // End the episode
        EndEpisode();
    }
}
}
```

APPENDIX B
PAPER SUBMISSION



4/9/24, 8:48 PM Conference Management Toolkit - Submission Summary

Submission Summary

Conference Name
5th IEEE India Council International Subsections Conference 2024

Track Name
Track 6 : Next Generation Computing and Applications

Paper ID
776

Paper Title
Exploring The Solar System using Deep Reinforcement Learning via ML Agents in Unity

Abstract
In this ground-breaking research, we take a trip through space with the "Solar System Adventure," a ground-breaking project that combines cutting-edge artificial intelligence with realistic virtual environments to mimic space exploration. We create an autonomous spaceship agent by utilizing deep reinforcement learning and Unity's powerful platform for machine learning. Through smart decision-making, this agent gains the ability to maneuver through the complex landscape of a digitally rendered solar system, dodging hazards and arriving at predetermined planetary destinations. The spaceship represents the fusion of AI's adaptability with the limitless possibilities of space exploration by adjusting its trajectory based on dynamic interactions within the virtual cosmos through a novel reward-based system.

Created
3/25/2024, 1:42:52 PM

Last Modified
3/25/2024, 1:42:52 PM

Authors
A BHAVANI SHANKAR (SRM INSTITUTE OF SCIENCE AND TECHNOLOGY)
<as3246@srmist.edu.in> ✓
KRISHNA KANT PANDEY (SRM INSTITUTE OF SCIENCE AND TECHNOLOGY)
<kp9827@srmist.edu.in> ✓
Vadivu G (SRM IST) <vadivug@srmist.edu.in> ✓

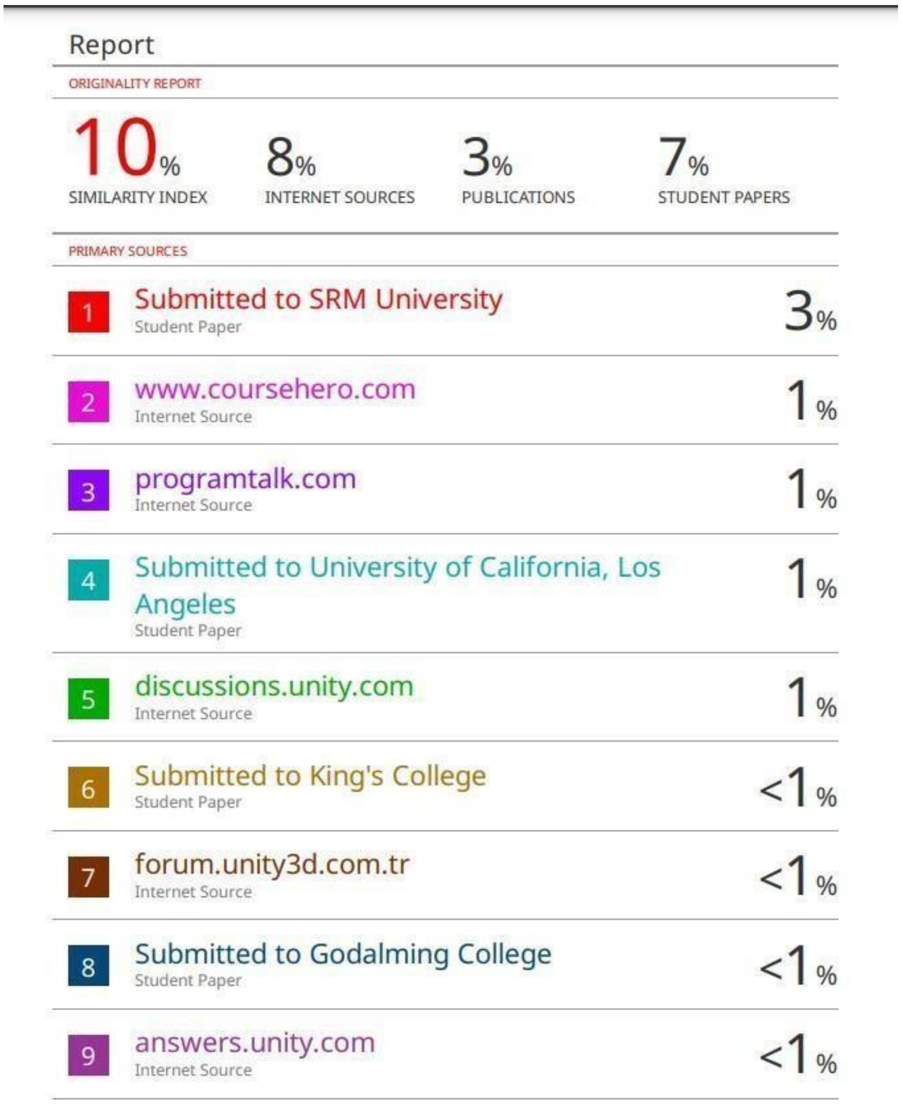
Primary Subject Area
Computational Intelligence

Submission Files
Exploring the Solar System using Deep Reinforcement Learning via ML Agents in Unity IEEE.pdf (341.9 Kb, 3/25/2024, 1:42:07 PM)

Submission Questions Response

https://cmt3.research.microsoft.com/INDISCON2024/Submission/Summary/776 1/2

APPENDIX C
PLAGIARISM REPORT



Report

ORIGINALITY REPORT

10%	8%	3%	7%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Submitted to SRM University Student Paper	3%
2	www.coursehero.com Internet Source	1%
3	programtalk.com Internet Source	1%
4	Submitted to University of California, Los Angeles Student Paper	1%
5	discussions.unity.com Internet Source	1%
6	Submitted to King's College Student Paper	<1%
7	forum.unity3d.com.tr Internet Source	<1%
8	Submitted to Godalming College Student Paper	<1%
9	answers.unity.com Internet Source	<1%