TRIBHUVAN UNIVERSITY

INSTITUTE OF ENGINEERING

ADVANCED COLLEGE OF ENGINEERING AND MANAGEMENT

DEPARTMENT OF ELECTRONICS AND COMPUTER ENGINEERING

KALANKI, KATHMANDU

A Major Project Final Defense Report On "AI Racing with Unity ML-Agents"

[CT 755]

SUBMITTED BY:

Mridul Sharma	[39676]
Mukesh Awasthi	[39677]
Niraj Niraula	[39678]
Prashant Shrestha	[39683]

SUPERVISED BY:

Er. Dhiraj Pyakurel

A project proposal report submitted to the department of Electronics and Computer Engineering in the partial fulfilment of the requirements for degree of Bachelor of Engineering in Computer Engineering

Kathmandu, Nepal

Baisakh, 2080

ADVANCED COLLEGE OF ENGINEERING AND MANAGEMENT

DEPARTMENT OF COMPUTER AND ELECTRONICS ENGINEERING

APPROVAL LETTER

The undersigned certify that they have read and recommended to the Institute of Engineering for acceptance, a project report entitled “AI Racing with Unity ML-Agents” submitted by:

Mridul Sharma	[39676]
Mukesh Awasthi	[39677]
Niraj Niraula	[39678]
Prashant Shrestha	[39683]

In partial fulfillment for the degree of Bachelor in Computer Engineering.

....

Er. Dhiraj Pyakurel

Project Supervisor

....

External Examiner

....

Er. Ajaya Shrestha

Head of Department

Department of Computer and Electronics Engineering

Baisakh, 2080

COPYRIGHT

ACKNOWLEDGEMENT

We take this opportunity to express our deepest and sincere gratitude to our Project Supervisor **Er. Dhiraj Pyakurel,** Department of Electronics and Computer Engineering for his insightful advice, motivating suggestions, invaluable guidance, help and support in successful completion of this project and also for his constant encouragement and advice throughout our Bachelor's programme.

We express our deep gratitude to **Er. Ajaya Shrestha**, Head of Department of Electronics and Computer Engineering, **Er. Bikash Acharya**, Deputy Head, Department of Electronics and Computer Engineering, **Er. Laxmi Prasad Bhatt,** Academic Project Coordinator, Department of Electronics and Computer Engineering for their regular support, co-operation, and coordination.

The in-time facilities provided by the department throughout the Bachelors program are also equally acknowledgeable.

We would like to convey our thanks to the teaching and non-teaching staff of the Department of Electronics & Communication and Computer Engineering, ACEM for their invaluable help and support throughout the period of Bachelor's Degree. We are also grateful to all our classmates for their help, encouragement and invaluable suggestions.

Finally, yet more importantly, we would like to express our deep appreciation to our grandparents, parents, siblings for their perpetual support and encouragement throughout the Bachelor's degree period.

Mridul Sharma	[39676]
Mukesh Awasthi	[39677]
Niraj Niraula	[39678]
Prashant Shrestha	[39683]

ABSTRACT

Computer games are an increasingly popular application for Artificial Intelligence (AI) research, and conversely AI is an increasingly popular selling point for commercial games. In video games, artificial intelligence (AI) is used to generate responsive, adaptive or intelligent behaviours primarily in non-player characters (NPCs) similar to human-like intelligence. Artificial intelligence has been an integral part of video games since their inception in the 1950s. Artificial intelligence in computer games covers the behaviour and decision-making process of game-playing opponents. AI in video games is a distinct subfield and differs from academic AI. It serves to improve the game-player experience rather than machine learning or decision making. Although games are typically associated with entertainment, there are many "serious" applications of gaming, including military, corporate, and advertising applications. There are also so-called "humane" gaming applications for medical training, educational games, and games that reflect social consciousness or advocate for a cause. Game AI is the effort of going beyond scripted interactions, however complex, into the arena of truly interactive systems that are responsive, adaptive, and intelligent. Such systems learn about the player(s) during game play, adapt their own behaviors beyond the pre-programmed set provided by the game author, and interactively develop and provide a richer experience to the player(s). Our project is also a gaming project where agents learn how to play by reinforcement learning where the agents are rewarded for making correct decisions. This will help agents in effective learning. Hence, making the game more tough and entertaining.

Keywords: *Responsive, adaptive, intelligent, reinforcement learning, Game AI, humane, agents.*

TABLE OF CONTENTS

APPROVAL LETTER i

COPYRIGHT ii

ACKNOWLEDGEMENT iii

ABSTRACT iv

LIST OF FIGURES vii

LIST OF ABBREVIATIONS/ACRONYMS viii

CHAPTER 1
INTRODUCTION 1

1.1 Background 1

1.2 Motivation 1

1.3 Statement of the problem 2

1.4 Project Objective 3

1.5 Significance of the study 4

CHAPTER 2
LITERATURE REVIEW 5

CHAPTER 3
REQUIREMENT ANALYSIS 8

3.1 Project Requirement 8

3.1.1 Software Tools 8

3.1.2 Hardware Tools 9

3.1.3 Functional Requirement 9

3.1.4 Non-functional Requirement 10

CHAPTER 4
SYSTEM DESIGN & ARCHITECTURE 11

4.1 System Design 11

4.2 Use case diagram 12

4.3 System Diagram 13

4.4 Sequence Diagram 14

CHAPTER 5

METHODOLOGY 15

5.1 Game Development Steps 15

5.2 Flowchart 17

5.3 Block Diagram 18

5.4 Training Algorithm 19

5.5 Software Development Model 20

CHAPTER 6

RESULT AND ANALYSIS 21

CHAPTER 7

CONCLUSION, LIMITATIONS AND FUTURE ENHANCEMENT 25

7.1 Conclusion 25

7.2 Limitations 25

7.3 Future Enhancements 25

REFERENCES 26

LIST OF FIGURES

Fig 4. 1: Architecture of the proposed game 11
Fig 4. 2: Use Case Diagram 12
Fig 4. 3: System Diagram 13
Fig 4. 4: Sequence Diagram 14
Fig 5. 1: Flowchart 17
Fig 5. 2: Game Development Block Diagram 18
Fig 5. 3: Agile Model 20
Fig 6. 1: Scene 21
Fig 6. 2: Game Menu 22
Fig 6. 3: Mode Option 22
Fig 6. 4: Difficulty Option 23
Fig 6. 5: Gameplay 23
Fig 6. 6: Gameplay(ii) 24

LIST OF ABBREVIATIONS/ACRONYMS

RL -	Reinforcement Learning
PPO -	Proximal Policy Optimization
AGI -	Artificial General Intelligence
AI -	Artificial Intelligence
NPC-	Nonplayer Character
3D -	Three-Dimensional
API -	Application Programming Interface
VR –	Virtual Reality
AR –	Augmented Reality
PC -	Personal Computer
RAM -	Random Access Memory
SSD -	Solid State Drive
PSU -	Power Supply Unit
ML-	Machine Learning
GAE -	Generalized Advantage Estimation

CHAPTER 1
INTRODUCTION

Reinforcement learning (RL) is an area of machine learning concerned with how intelligent agents ought to take actions in an environment in order to maximize the notion of cumulative reward[1]. Reinforcement learning is one of three basic machine learning paradigms, alongside supervised learning and unsupervised learning.

1.1 Background

Reinforcement learning is a machine learning training method based on rewarding desired behaviours and/or punishing undesired ones. In general, a reinforcement learning agent is able to perceive and interpret its environment, take actions and learn through trial and error[2]. In reinforcement learning, developers devise a method of rewarding desired behaviours and punishing negative behaviours. This method assigns positive values to the desired actions to encourage the agent and negative values to undesired behaviours. This programs the agent to seek long-term and maximum overall reward to achieve an optimal solution.

1.2 Motivation

Reinforcement learning helps you to find which situation needs action. Reinforcement learning helps you to discover which action yields the highest reward over the longer period. Reinforcement Learning also provides the learning agent with a reward function. It also allows it to figure out the best method for obtaining large rewards. With this project our intent is to create a full, playable aeroplane racing game in Unity with incredibly challenging AI opponents. The flight in the game will imitate real world flights through training with reinforcement learning, specifically PPO (Proximal Policy Optimization). With reinforcement learning, simulated aeroplane flight is expected to improve constantly over time.

1.3 Statement of the problem

As we are moving towards Artificial General Intelligence (AGI), designing a system which can solve multiple tasks (i.e., Classifying an image, playing a game ...) is really challenging. The current scope of machine learning techniques, be it supervised or unsupervised learning, are good at dealing with one task at any instant. This limits the scope of AI to achieve generality. To achieve AGI, the goal of RL is to make the agent perform many different types of tasks, rather than specialising in just one. This can be achieved by multi task learning and remembering the learning. In RL, an agent learns by interacting with its environment, observing the results of these interactions and receiving a reward (positive or negative) accordingly. This way of learning mimics the fundamental way in which we humans learn.

1.4 Project Objective

The objective of our project is:

- To create AI racing game with Unity ML-agents.

1.5 Significance of the study

Reinforcement learning delivers decisions. By creating a simulation of an entire business or system, it becomes possible for an intelligent system to test new actions or approaches, change course when failures happen (or negative reinforcement), while building on successes (or positive reinforcement). Much in the way human beings can develop a skill as they practice it, reinforcement learning only becomes more powerful when it’s executed at scale. The type of problems that reinforcement learning solves are simply beyond human capabilities, which is why it is AI that acts as the perfect assistant to human beings. They might involve making repetitive decisions that human workers don’t have the time to look at on a regular basis, decisions that are too complex with too many factors, that are highly mathematical and that occur at an enormous volume.

CHAPTER 2

LITERATURE REVIEW

A long-term, overarching goal of research into reinforcement learning (RL) is to design a single general purpose learning algorithm that can solve a wide array of problems. However, because the RL algorithm taxonomy is quite large, and designing new RL algorithms requires extensive tuning and validation, this goal is a daunting one. A possible solution would be to devise a meta-learning method that could design new RL algorithms that generalize to a wide variety of tasks automatically[3].

The growing awareness of reinforcement learning is happening at a time when the conversation needs to shift squarely towards real business, which can use it to simulate and discover the optimal sequence of decisions to make[4]. In the grocery industry, for example, acting on those decisions can grow revenue by > 5% and more than double net income.

L.P. Kaelbling et al. (1996) surveyed the field of reinforcement learning from a computer-science perspective. Reinforcement learning is the problem faced by an agent that learns behaviour through trial-and-error interactions with a dynamic environment[5]. The work described here has a resemblance to work in psychology, but differs considerably in the details and in the use of the word “reinforcement”.

Exploration is needed for an agent, in order to learn all available states, present in an environment and how to deal with it optimally, agent must know as many states as possible. In a RL problem, agent has limited access via its actions on the environment unlike other traditional supervised learning approaches[6]. As an outcome, there occurs a problem that is, for an agent to learn optimum policy it requires the right amount of experiences, but for obtaining those experiences requires a good policy.

According to Csaba Szepesvári, reinforcement learning is a learning paradigm concerned with learning to control a system so as to maximize a numerical performance measure that expresses a long-term objective. What distinguishes reinforcement learning from supervised learning is that only partial feedback is given to the learner about the learner's predictions[7]. Further, the predictions may have long term effects through influencing the future state of the controlled system. Thus, time plays a special role. The goal in reinforcement learning is to develop efficient learning algorithms, as well as to understand the algorithms' merits and limitations. Reinforcement learning is of great interest because of the large number of practical applications that it can be used to address, ranging from problems in artificial intelligence to operations research or control engineering.

The Unity Machine Learning Agents Toolkit (ML-Agents Toolkit) is an open-source project that enables games and simulations to serve as environments for training intelligent agents. Agents can be trained using reinforcement learning, imitation learning, neuroevolution, or other machine learning methods through a simple-to-use Python API. They also provide implementations (based on PyTorch) of state-of-the-art algorithms to enable game developers and hobbyists to easily train intelligent agents for 2D, 3D and VR/AR games. These trained agents can be used for multiple purposes, including controlling NPC behaviour (in a variety of settings such as multi-agent and adversarial), automated testing of game builds and evaluating different game design decisions pre-release[8]. The ML-Agents Toolkit is mutually beneficial for both game developers and AI researchers as it provides a central platform where advances in AI can be evaluated on Unity’s rich environments and then made accessible to the wider research and game developer communities.

Artificial intelligence (AI) in computer games covers the behaviour and decision-making process of game-playing opponents (also known as nonplayer character or NPC). Current generations of computer and video games offer an amazingly interesting testbed for AI research and new ideas. Such games combine rich and complex environments with expertly developed, stable, physics-based simulation. They are real-time and very dynamic, encouraging fast and intelligent decisions. Computer games are also often multiagent, making teamwork, competition, and NPC modelling key elements to success. In commercial games, such as action games, role-playing games, and strategy games, the behaviour of the NPC is usually implemented as a variation of simple rule-based systems. With a few exceptions, machine-learning techniques are hardly ever applied to state-of-the-art computer games. Machine-learning techniques may enable the NPCs with the capability to improve their performance by learning from mistakes and successes, to automatically adapt to the strengths and weaknesses of a player, or to learn from their opponents by imitating their tactics[9].

CHAPTER 3
REQUIREMENT ANALYSIS

3.1 Project Requirement

In this section, we are mentioning the software and hardware tools that are required to develop our project.

3.1.1 Software Tools

For the development of our project, we will use different software tools, and they are:

3.1.1.1 Unity

Initially released in 2005, Unity is a game development environment and engine for over 20 various platforms. It works on MAC OS X, Linux, and Windows. Unity quickly became the go-to for people warming up to game making with its simple and easy-to-understand interface. Frequent updates, good documentation and an even better community made it one of the best engines to work on.

3.1.1.2 C# (C-Sharp)

C# is a general-purpose, multi-paradigm programming language. C# encompasses static typing, strong typing, lexically scoped, imperative, declarative, functional, generic, object-oriented, and component-oriented programming disciplines. C# is clearer and more organized, and it's much faster at runtime. C# can be used to create a number of different programs and applications: mobile apps, desktop apps, cloud-based services, websites, enterprise software and games.

3.1.1.3 Blender

Blender is a free and open-source 3D computer graphics software toolset used for creating animated films, visual effects, art, 3D-printed models, motion graphics, interactive 3D applications, virtual reality, and, formerly, video games.

3.1.1.4 Python

Python is a high-level, interpreted, general-purpose programming language and is often used to build websites and software, automate tasks, and conduct data analysis. Its design philosophy emphasizes code readability with the use of significant indentation. Python is a general-purpose language, meaning it can be used to create a variety of different programs and isn't specialized for any specific problems.

3.1.1.5 GitHub

GitHub is a code hosting platform for version control and collaboration. It lets us and others work together on projects from anywhere. We can host our source code projects in different programming languages and keep track of the changes made by the team using Git. Unless GitHub is used for private development it is free to use for every user.

3.1.2 Hardware Tools

We will be using the following hardware components for the development of this project.

3.1.2.1 Personal Computer

A high-end PC with minimum of 16GB RAM, about 512GB SSD, Quad-Core processor, minimum of 4GB of Graphics Card and with about 650 to 850 watts of PSU.

3.1.2.2 Joystick

A joystick is an input device that can be used for controlling the movement of the cursor or a pointer in a computer device. This input device is mostly used for gaming applications and, sometimes, in graphics applications.

3.1.3 Functional Requirement

Some of the functional requirements of our software are:

- The game should not crash.
- The game must be fun to play.
- The AI opponents must be difficult to win for the player.

3.1.4 Non-functional Requirement

Here are some of the non-functional requirements of the system:

- Usability: The proposed game must be easy to play.
- Portability: The game must function properly on multiple devices with different specifications.
- Scalability: The capability of the proposed system to handle the growing users must be good.

CHAPTER 4

SYSTEM DESIGN & ARCHITECTURE

4.1 System Design

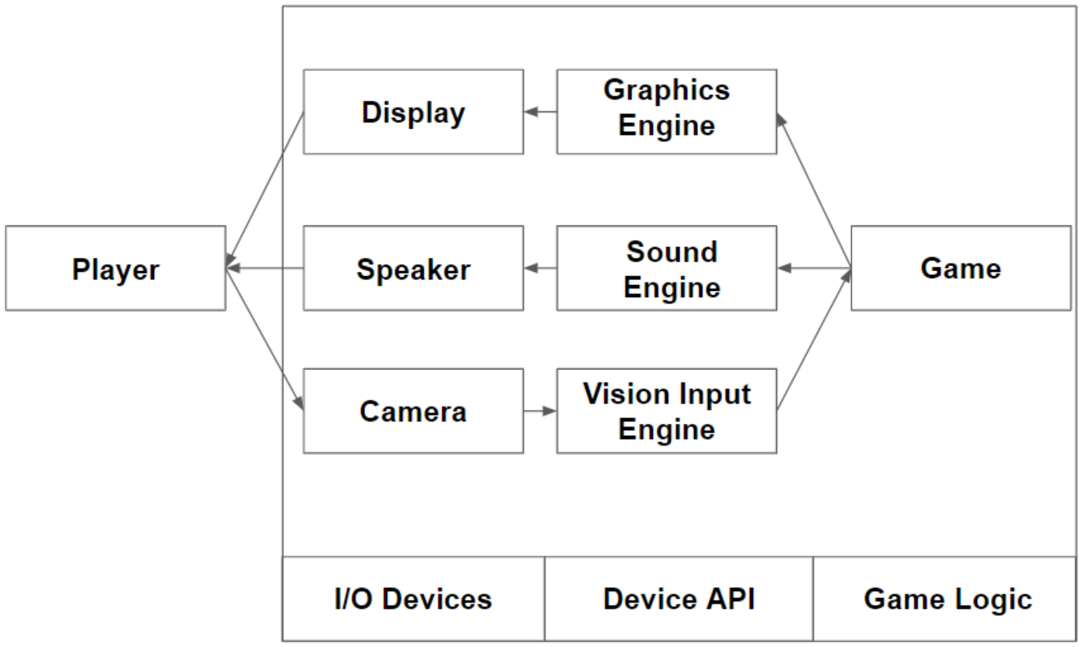

Fig 4. 1: Architecture of the proposed game

All of these above-mentioned components combine together to make a game playable to its full extent. Before discussing each of them, let's just make ourselves clear what an engine is. Engines in the computer field refers to a software which helps in performing some definite type of processing on programs. The major components of a game engine are input, graphics, physics, AI, sound, and networking.

A graphic engine is a software which in association with an application program helps to draw graphics on our computer's display. A sound engine is the component that consists of algorithms for dealing with sound and in-built programs are written into it to handle the sound effects in the game. And, the vision input engine produces 3D animated graphics.

For a game, there should have to be a strong interaction between the user and the game he/she is playing. So, for this the peripheral devices like mouse, keyboard, joysticks and monitors play a major role to make the game interactive. A device API can be defined as Application Programming Interface which lets the developer create any application which interacts with hardware devices plugged in within our system. A device API normally provides end users to employ their plugged or associated hardware for interacting with the system. Majority of the games need these device drivers and APIs to make the game work with all its components running successfully.

4.2 Use case diagram

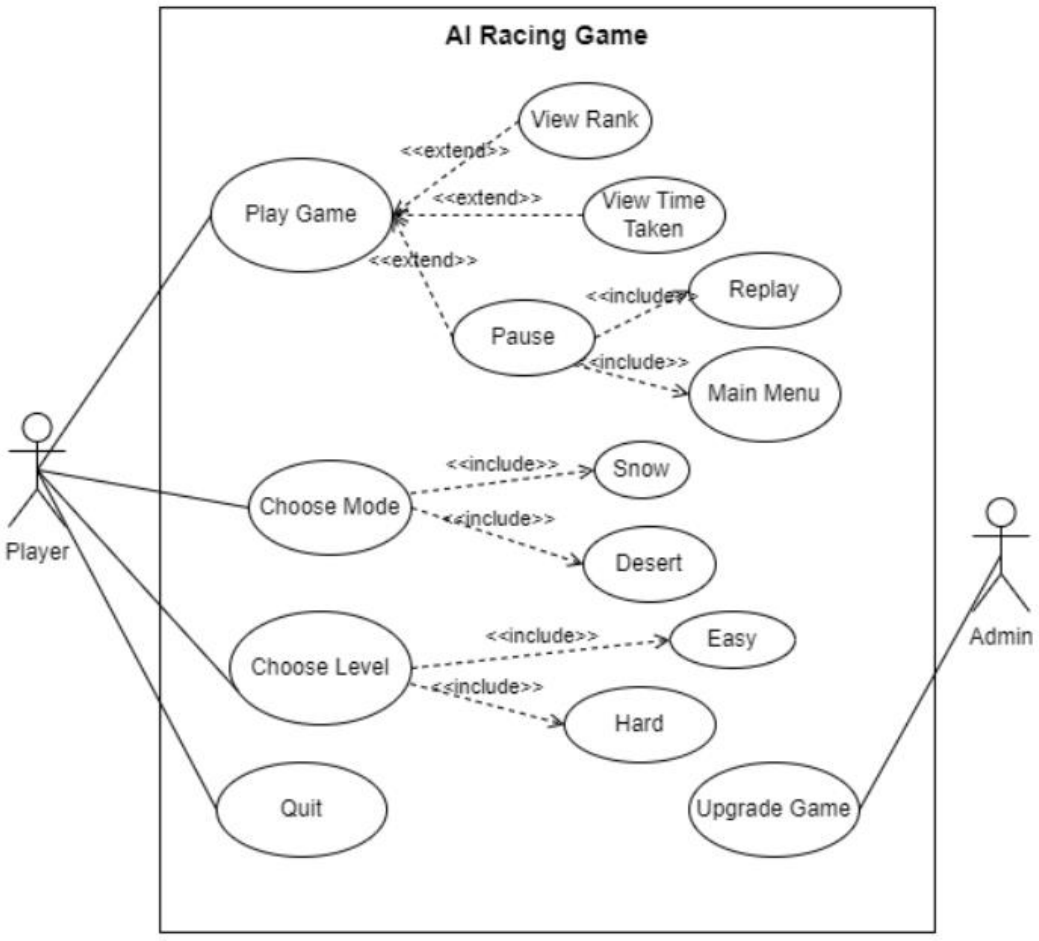

Fig 4. 2: Use Case Diagram

4.3 System Diagram

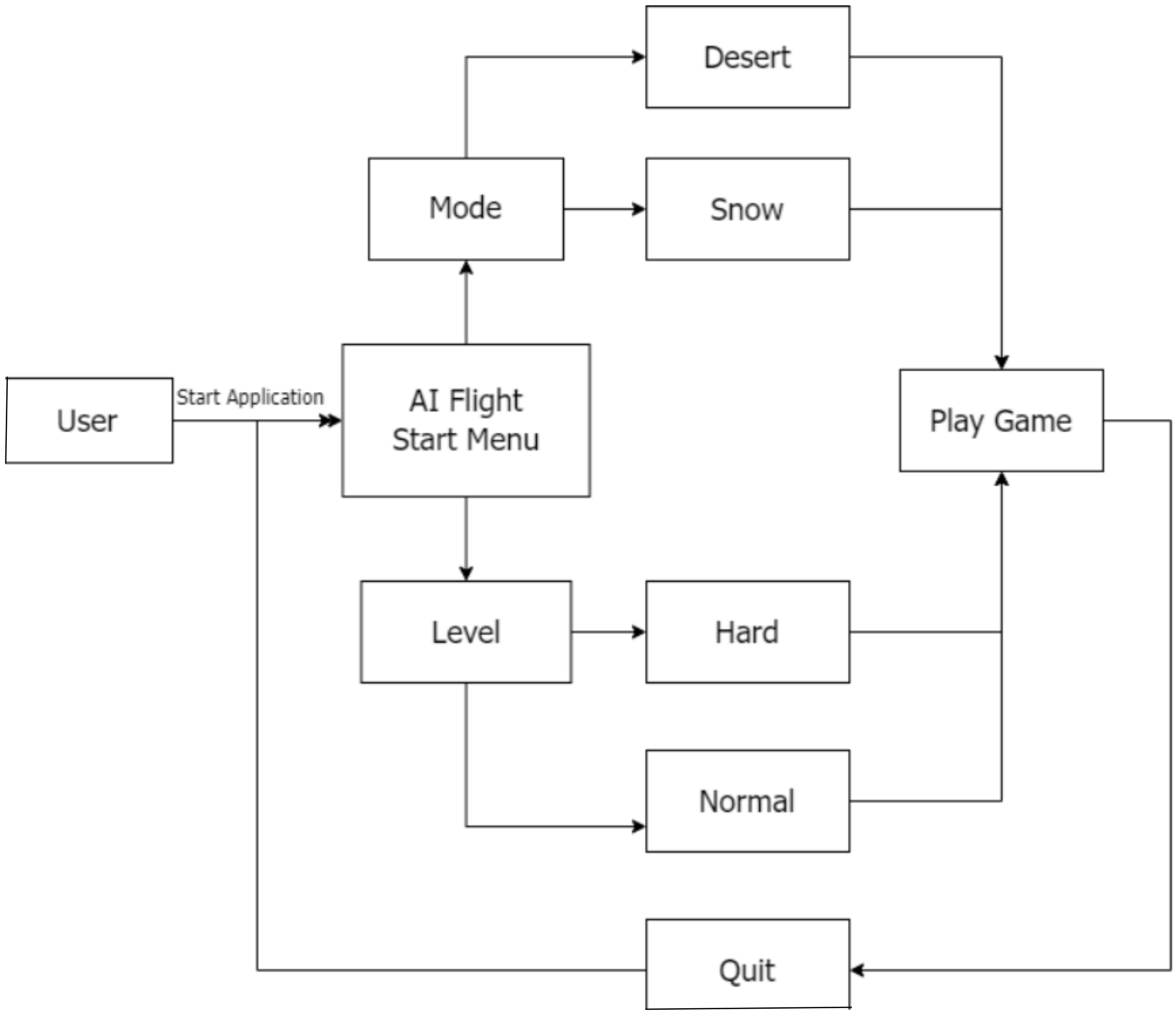

Fig 4. 3: System Diagram

4.4 Sequence Diagram

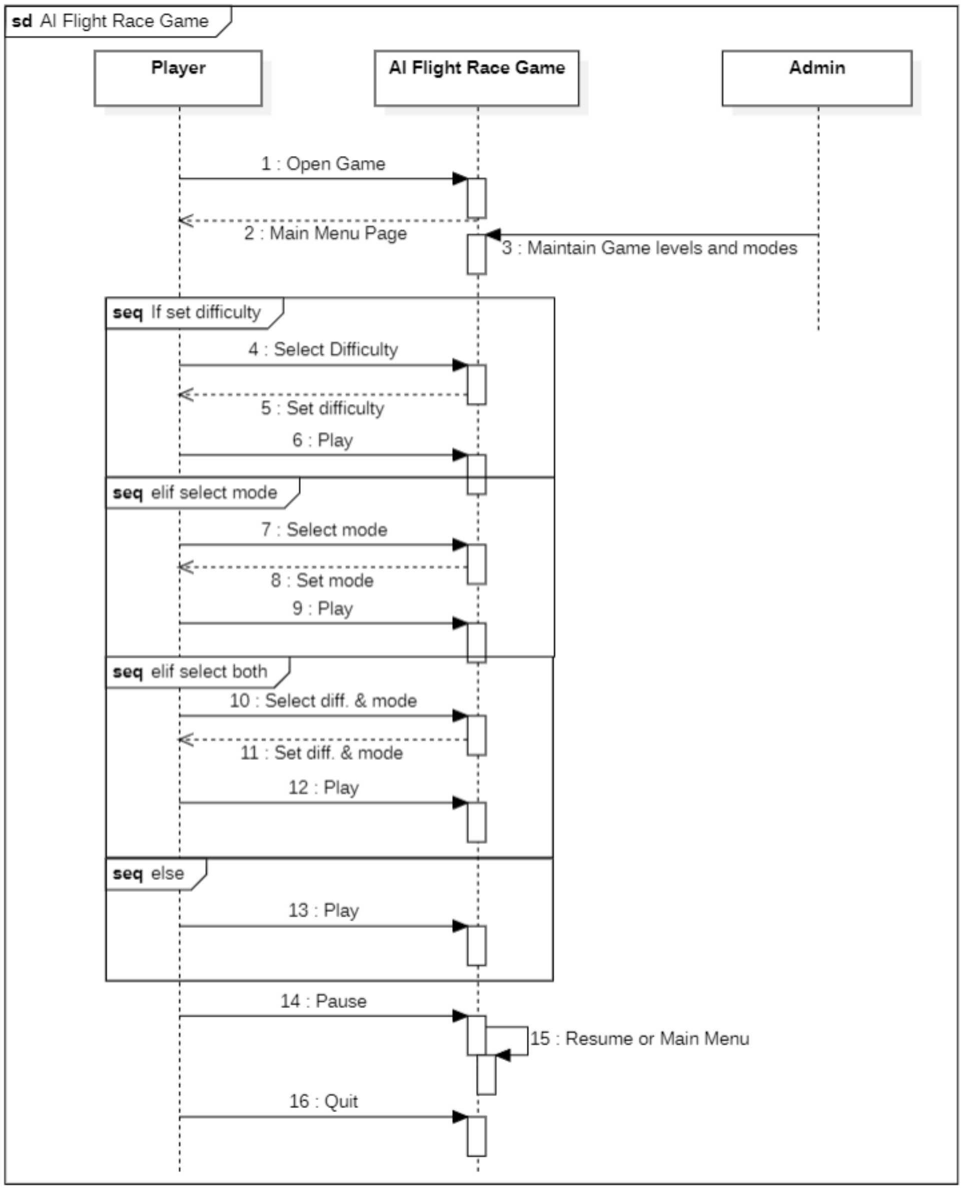

Figure 4. 4: Sequence Diagram

CHAPTER 5
METHODOLOGY

In this section, we briefly describe the process of our game development and Algorithm we are going to use.

5.1 Game Development Steps

We will be going through the following steps to complete our game.

5.1.1 Assets Creation

As we know gaming assets are the most basic things in any game. Initially, we will create the assets for our game which includes elements such as player agent, learning agents etc. We will achieve this by using Blender.

5.1.2 Importing Assets into Unity

Now, the assets and ML-Agents should be imported to Unity as we will be doing the rest of work using unity.

5.1.3 Environment Creation

A dimension which collaborates game rules, objectives, subject, and theoretical aspects together as a whole to provide an interactive flow of activity is called Game Environment. This step includes creation of the game environment with Unity.

5.1.4 Configuring Actions and Behaviours

In this step, we are basically going to configure the player agent. This can include things such as on which conditions, which actions should be taken, logics behind the game and many other aspects.

5.1.5 Creating learning environments and learning agents

We will have a player agent and the rest of agents will be learning agents. For training, we should create some learning environments and learning agents because we do not want to train the agents in the same environment as the final game environment in order to show RL.

5.1.6 Training the learning agents

In this step, we are going to train the learning agents so that they are able to react to entirely new gaming environments.

5.1.7 Pause the training, check the progress, tune hyperparameters etc.

In this step, we will pause the training in order to check the progress. If the progress is not as good, we tune in our hyperparameters, train again and choose the perfectly trained model which we require in our game.

5.1.8 Playable Game

The final step will be to create a playable game in unity where player gets to play and experience the game.

5.2 Flowchart

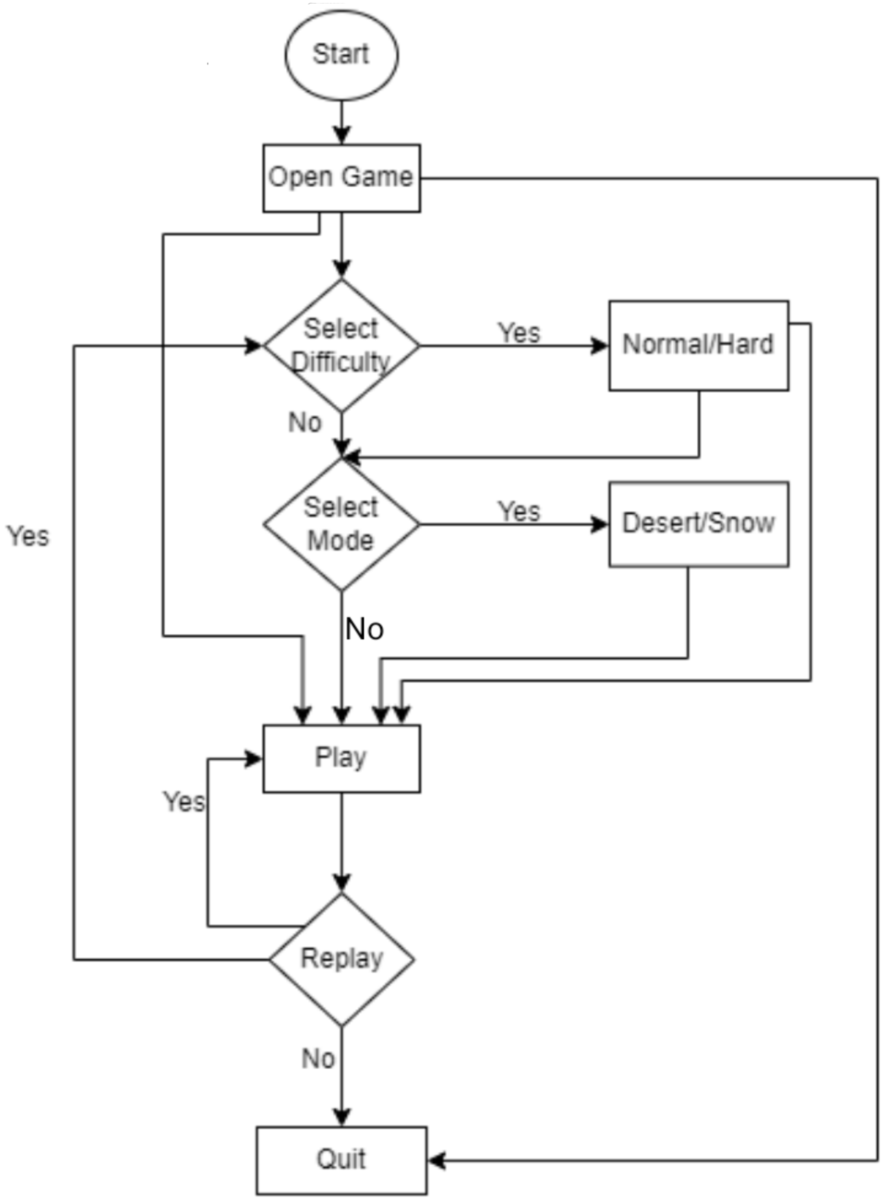

Fig 5. 1: Flowchart

5.3 Block Diagram

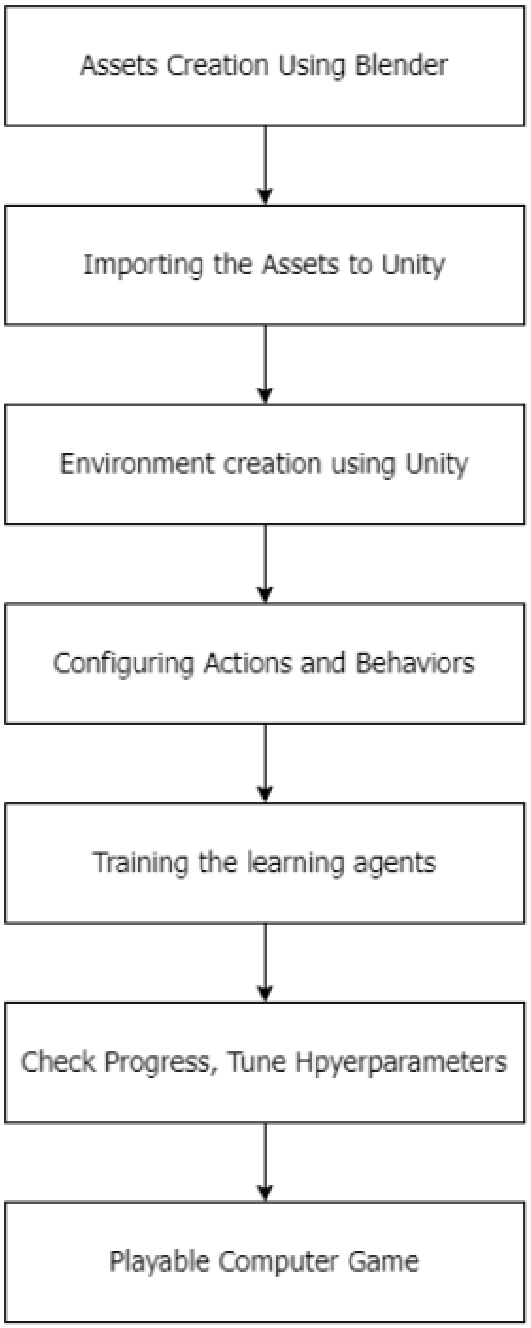
Assets Creation Using Blender
Importing the Assets to Unity
Environment creation using Unity
Configuring Actions and Behaviors
Training the learning agents
Check Progress, Tune Hpyerparameters
Playable Computer Game

Fig 5. 2: Game Development Block Diagram

5.4 Training Algorithm

With supervised learning, we can easily implement the cost function, run gradient descent on it, and be very confident that we'll get excellent results with relatively little hyperparameter tuning. The route to success in reinforcement learning isn't as obvious - the algorithms are hard to debug, and they require substantial effort in tuning in order to get good results. So, we are going to use an algorithm known as Proximal Policy Optimization (PPO). PPO strikes a balance between ease of implementation, sample complexity, and ease of tuning, trying to compute an update at each step that minimizes the cost function while ensuring the deviation from the previous policy is relatively small.

The PPO algorithm was introduced by the OpenAI team in 2017 and quickly became one of the most popular RL methods usurping the Deep-Q learning method. It involves collecting a small batch of experiences interacting with the environment and using that batch to update its decision-making policy. Once the policy is updated with this batch, the experiences are thrown away and a newer batch is collected with the newly updated policy. This is the reason why it is an "on-policy learning" approach where the experience samples collected are only useful for updating the current policy once.

The key contribution of PPO is ensuring that a new update of the policy does not change it too much from the previous policy. This leads to less variance in training at the cost of some bias, but ensures smoother training and also makes sure the agent does not go down an unrecoverable path of taking senseless actions.

Advantage can be defined as a way to measure how much better off we can be by taking a particular action when we are in a particular state. We want to use the rewards that we collected at each time step and calculate how much of an advantage we were able to obtain by taking the action that we took. In order to calculate this, we'll use an algorithm known as Generalized Advantage Estimation or GAE.

After Calculating the GAE, we now use it to calculate PPO loss and try to minimize it.

5.5 Software Development Model

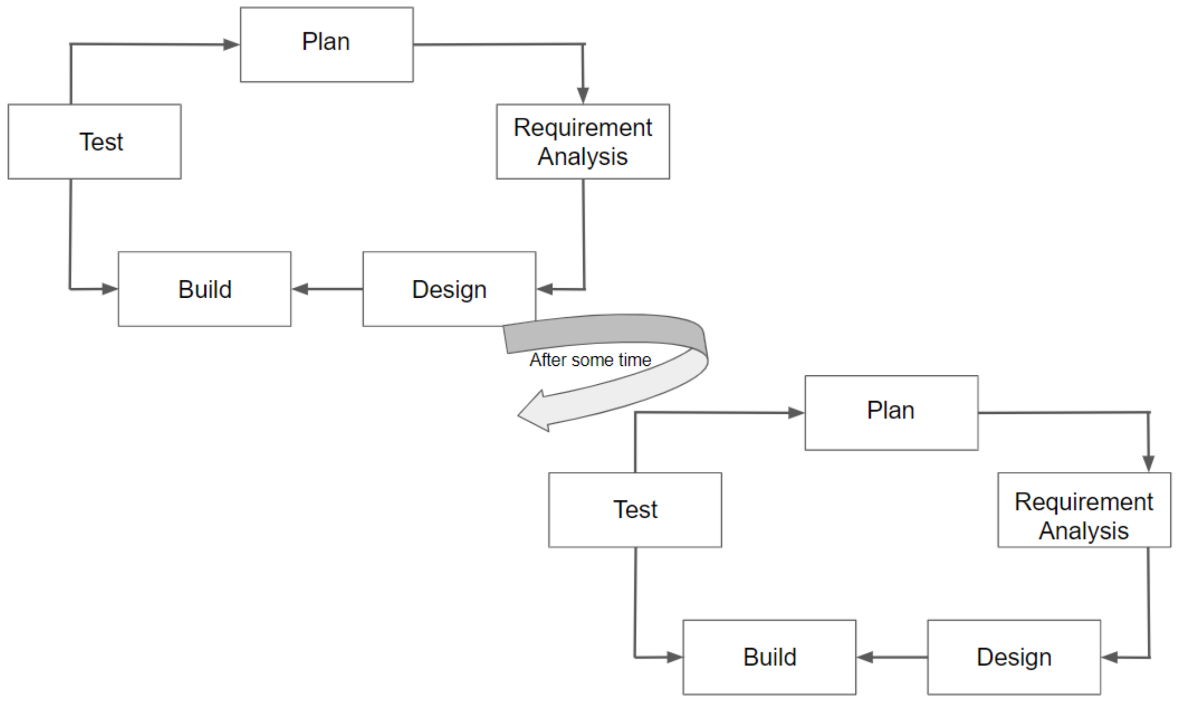

Fig 5. 3: Agile Model

The Agile model adopts Iterative development. Each incremental part is developed over an iteration. Each iteration is intended to be small and easily manageable and can be completed within a couple of weeks only. At a time one iteration is planned, developed and deployed to the customers. After some period of time the same iteration is performed and a more accurate product is developed by updating the old one. This model of software development is the most effective model in present software development society.

CHAPTER 6

RESULT AND ANALYSIS

From the creation of 3D models to training those models and placing them appropriately in the scene of the game was completed as per our work schedule. Lastly, the demo game works fine with little to no errors.

- Creation of 3D models.
- Environment and Agent creation.
- Development of Scene.
- Creation of Agent.
- Placement of Agent in the Scene.
- Scripts for the Scene and Agent.
- Scripts for game Hub.
- Training and Evaluating Model.

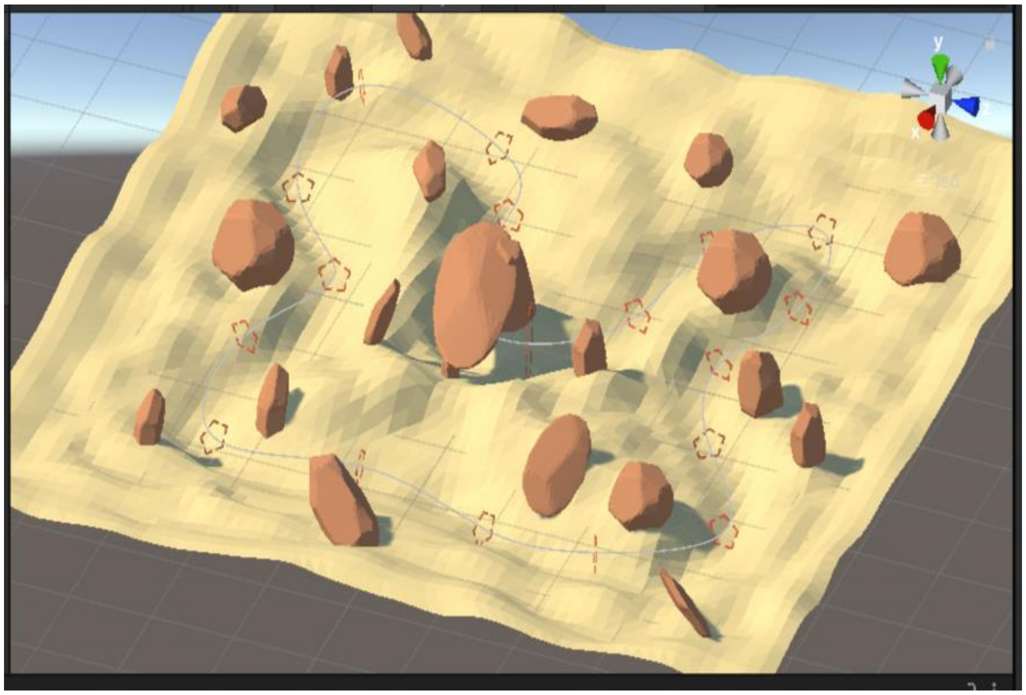

Fig 6. 1: Scene

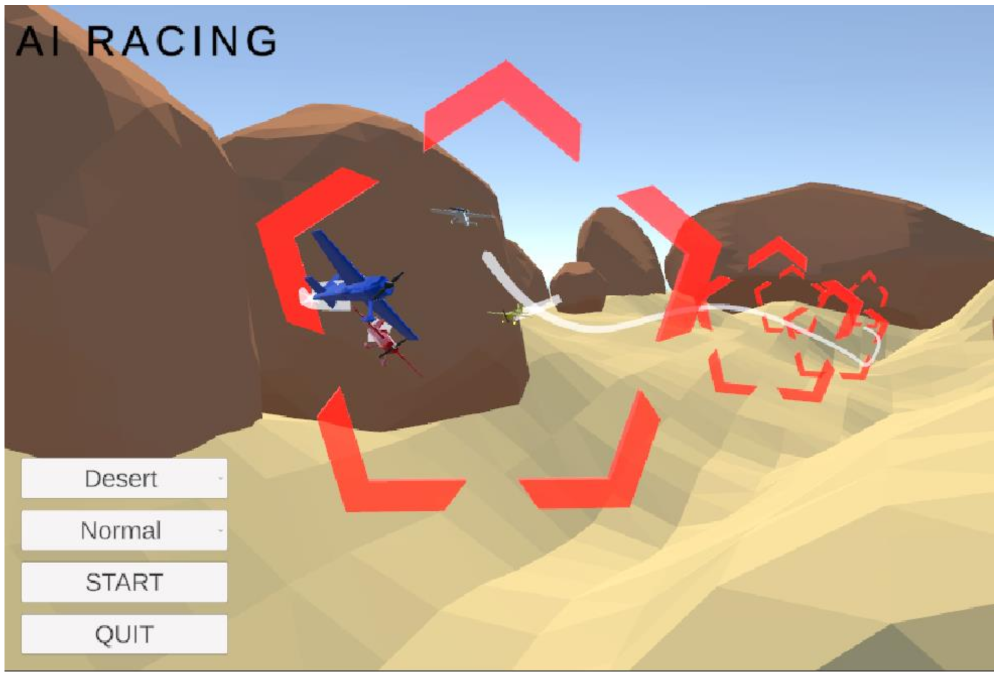

Fig 6. 2: Game Menu

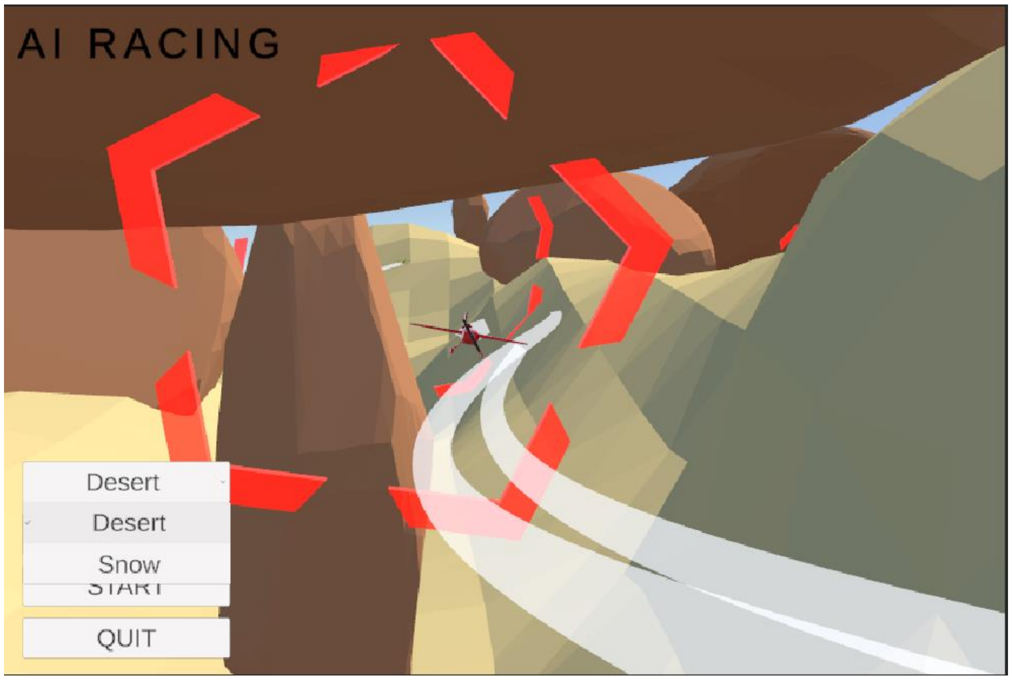

Fig 6. 3: Mode Option

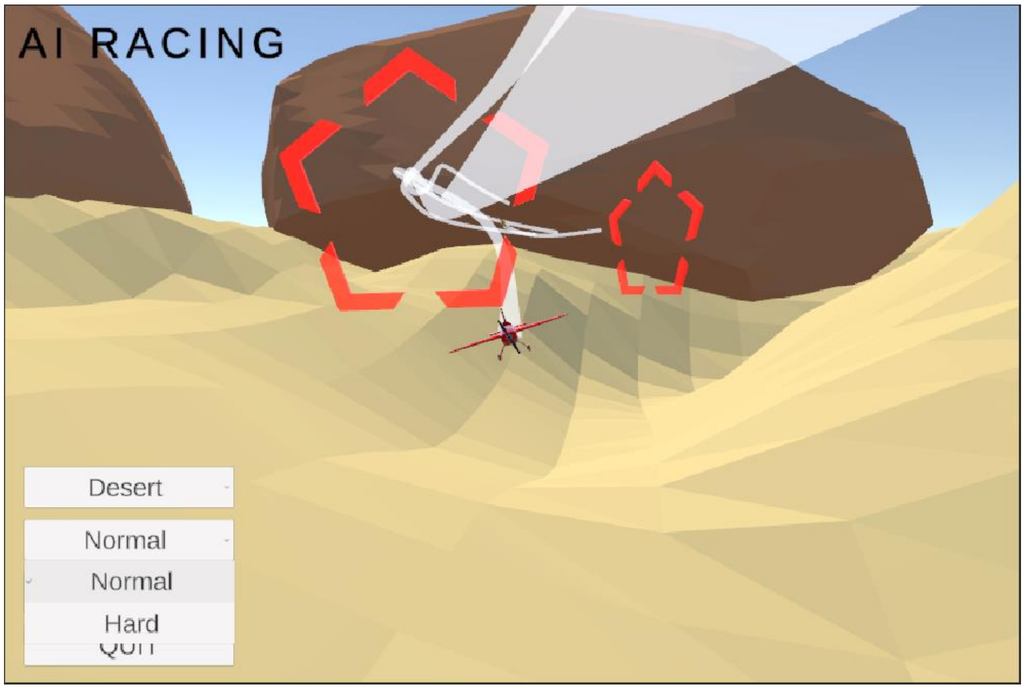

Fig 6. 4: Difficulty Option

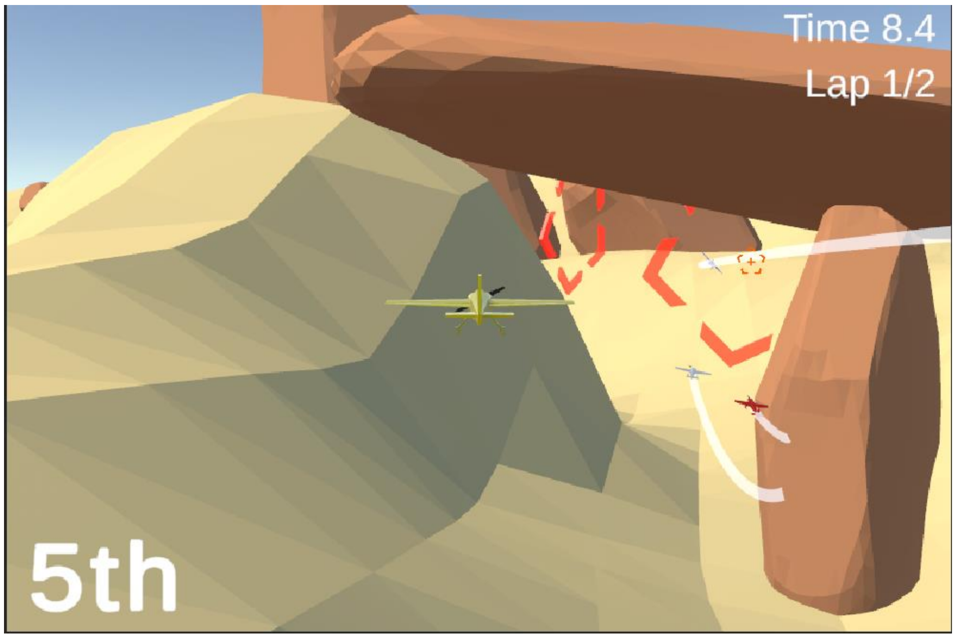

Fig 6. 5: Gameplay

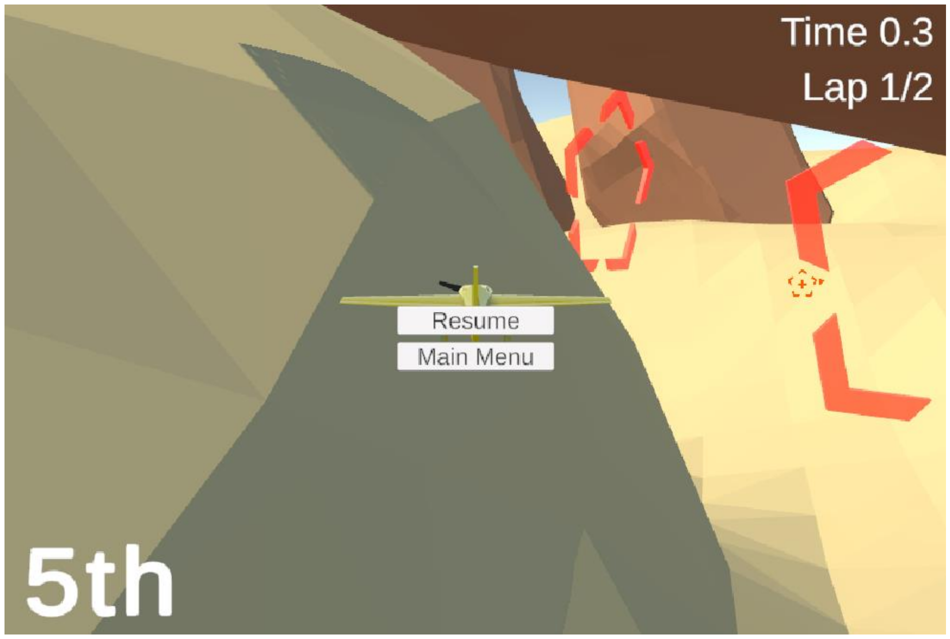

Fig 6. 6: Gameplay(ii)

CHAPTER 7
CONCLUSION, LIMITATIONS AND FUTURE ENHANCEMENT

7.1 Conclusion

As per our project's objective, we developed a fully playable flight racing game with unity ML-agents where the human player competes with almost flawless AI opponents that enhances the game playing experience for the users.

7.2 Limitations

- Only one player can play the game at a time.
- The game environment is not very sophisticated and is limited to a single scene at a time.

7.3 Future Enhancements

Almost everything has room for improvements, so does our project. Some of modifications that might improve the game overall in the future are:

- Make the game multiplayer and online.
- Develop application for the game.
- Develop an open world game with multiple added scenes and missions to make the game more enjoyable.

REFERENCES

[1] [Accessed on: https://en.wikipedia.org/wiki/Reinforcement_learning#:~:text=Reinforcement%20learning%20(RL)%20is%20an,supervised%20learning%20and%20unsupervised%20learning.]

[2] J.M. Carew “Reinforcement Learning”,2022.
[Accessed on: https://www.techtarget.com/searchenterpriseai/definition/reinforcement-learning]

[3] J. Co-Reyes and Y. Miao “Evolving Reinforcement Learning Algorithm”, April 22, 2021. [Accessed on: https://ai.googleblog.com/2021/04/evolving-reinforcement-learning.html]

[4] G. Saarenvirta “Why Reinforcement Learning Will Deliver What ‘AI’ Promised”, March 5,2019. [Accessed on: https://www.daisyintelligence.com/blog/reinforcement-learning-ai#:~:text=Reinforcement%20learning%20delivers%20decisions.,successes%20(or%20positive%20reinforcement)]

[5] L. P. Kaelbling, M. L. Littman and A. W. Moore, “Reinforcement learning: A survey”, *Journal of Artificial Intelligence Research*, vol. 4, pp. 237-285, 1996.

[6] M. Volodymyr *et al*., arXiv:1312.5602v1 “Playing Atari with Deep Reinforcement Learning”, December 19,2013.

[7] C. Szepesvári, Algorithms for Reinforcement Learning, Morgan and Claypool, 2010. [Accessed on:https://www.researchgate.net/publication/220696313_Algorithms_for_Reinforcement_Learning]

[8] [Accessed on: https://github.com/Unity-Technologies/ml-agents/blob/develop/docs/ML-Agents-Overview.md]

[9] K.W. Wong "Artificial Intelligence for Computer Games", January 2009. [Accessed on: https://www.researchgate.net/publication/220061084_Artificial_Intelligence_for_Computer_Games]