DRAFT



Computer Games Development SE607

Technical Design Document

Year IV

[Leo Chen]

[C00250593]

[24/04/2023]

[Declaration form to be attached]

**Contents**

[Acknowledgements 2](#_Toc54714373)

[Project Abstract 2](#_Toc54714374)

[Project Introduction 2](#_Toc54714375)

[Literature Review 3](#_Toc54714376)

[Evaluation and Discussion 3](#_Toc54714377)

[Conclusions 4](#_Toc54714378)

[References 4](#_Toc54714379)

[Appendices 4](#_Toc54714380)

# Acknowledgements

I would like to thank the following people who assisted in aiding this project.

Lei Shi, my supervisor for giving me advice and insights on the project as a whole and my colleagues for their ideas, feedback and input as well as believing in me and my project.

# Project Abstract

The Real Time Strategy Game Genre has provided a concrete and effective method to allow researchers to introduce and study complex algorithms that adapt to its evolving landscape. The popular RTS game StarCraft (SC) and its sequel StarCraft II (SC2) has been utilised as a basis for developing Machine Learning algorithms to further the research on Artificial Intelligence. Previous researchers have constructed a framework to build from that allows others to create and expand their own algorithms as they see fit with great success. The attempt for this project focuses on creating an entirely new, smaller scale RTS game that uses a machine learning algorithm as a core concept for an opponent. The idea is to create a basic environment that allows the algorithm to achieve simple goals that cumulate to the advancement of a much larger and complex objective through the mechanics of the game. In the end the machine learning algorithm would be able to adapt to a human opponent more readily.

# Project Introduction

With the advancement of artificial intelligence, most efforts have been placed on training neural networks to recognise images, understanding and reinterpreting data to provide an output. The purpose of this project was to create a simple but expandable environment with an entirely new game from the ground up. This will allow me to experiment with the algorithm it is present in and create new challenges for it to overcome.

Developed using the Unity engine for the foundation of this game, the platform will be the test bed for Unity’s Machine Learning Agent (MLA), other neural networks or non-machine learning AIs to work as the opponent. The environment will consist of two main objectives, Control points (CP) and Resource points (RP). One will generate score while the other generates funding for their respective player when captured. These areas will be the main points of contention the players will fight over.

The simplistic environment at first will allow for the MLA to get familiar with the controls and actions they will be utilising. Upon performing such actions to an acceptable level, the environment will expand to include more and more complex game mechanics for the MLA to familiarise and act on.

This project will also explore the use of other AIs as a potential opponent for the MLA. Its use as the MLA’s opponent will be used to gauge the effectiveness of what the MLA has learned and compare it to traditionally programmed AIs. Such traditionally made AI’s for comparison would be Finite State Machines, behaviour trees or decision trees.

The aim of this project is a means to demonstrate the potential use of Unity’s ML-Agent’s toolkit for creating adaptive AI opponents in RTS video games. By developing the MLA to evolve into an engaging and challenging opponent, the project could provide a good foundation for the expansion of utilising MLA for their own games in the industry.

# Literature Review

## Artificial Intelligence and Learning Algorithms in video games

The implementation of learning algorithms for expanding the research of artificial intelligence has woven its way into video games. With the cooperation between then Blizzard entertainment and Deepmind, they have developed the SC2LE (StarCraft 2 Learning Environment), a program that introduces a Reinforced Learning Algorithm into the complex and chaotic game. Though initially created in controlled “mini-game” environments, it has allowed other researchers to expand the scope of the algorithm to encompass the entire game. Different researchers have created AI Agents to compete against StarCraft 2’s built in cheat level built in AIs with great success.

Though most have mainly focused on SC2LE, a different project was developed to create a smaller scale RTS that featured a machine learning algorithm. Micro RTS, developed by Santiago Ontañón, is a small implementation of a real time strategy game which is used to observe and perform AI research. It involves two players controlling and managing small armies of units in which their goal is to destroy their opponent’s base. The implementation of the game is simple enough for algorithms to run out but the capacity for more advanced strategy with more complex unit types are not currently present.

StarCraft 2’s environment and gameplay revolves around much higher and intensive logical computation. The SC2LE was developed by Deepmind in partnership with (at the time) Blizzard Entertainment. The platform provides a complex and challenging arena for the AI agents to learn and adapt to its ever-shifting environment. The game itself involves several key aspects of that players need to manage. Players will need to manage resources, unit production, micro[[1]](#footnote-1) and macro[[2]](#footnote-2) gameplay, on top of a rapidly evolving environment. Both tactics and strategy are paramount to success in the game.

A study by Tencent AI Labs, University of Rochester and Northwest University developed the TStarBots which are two machine learning agents designed to defeat the built-in Cheating AI in StarCraft 2. The first AI known as TStarBot1 is a Macro Action Based Reinforcement Learning Agent developed to act as a single global controller with hard coded macro actions available to it. This multi-layer perception neural network architecture used for the study allows TStarBot1 to learn and act on situations present within an active game by utilising the macros available to it. The second AI known as TStarBot2 is a Hierarchical Macro-Micro Action Based Agent designed for both macro and micro actions. This version can, as the description implies, the ability to command both the Macro level of thinking as well as the Micro section of per unit commands. Both of which has seen success in completing a full match/level/game of StarCraft2 as well as being able to defeat high ranking player.

Micro-RTS-Py by the Farama-Foundation is another example for machine learning algorithms in the RTS Genre. It is a platform specifically designed for testing machine learning algorithms in an RTS inspired game environment which is the inspiration for the premises of this project. Unlike StarCraft 2’s more heavily complex game with several times the magnitude of computational need for training neural networks, Micro-RTS has strived for a highly efficient and affordable test bed for machine learning algorithms. Its environment can be used to train agents more quickly relative to the time it would take the likes of TStarBot or other machine learning algorithms in more complex environments to train. This test bed allows for researchers to test their machine learning algorithms to compete against other AIs in its field.

In the environment Micro-RTS provides, its 16x16 grid with two corners in which the players’ base is located. The unit and building roster consist of basic entities such as workers and different kinds of soldiers that make up the RTS environment as well as simple production and resource nodes. The simulation of the inputs for the algorithm consists of two main components that allow for the neural networks to act on. The observation and action space utilised features the observation space to generate the data seen on the grid while the action space as the name implies, provides the actions for the algorithm to utilise.

Overall, the studies found for machine learning algorithms in real time strategy games provide the potential for producing highly sophisticated game AIs. By utilising various methods of machine learning, an AI agent can create a challenging and engaging opponent for even the most skilled players of the genre. As artificial intelligence evolves over time, it is possible that machine learning algorithms as opponents in video games be a potential avenue for exploration.

## A new game environment

The reason I wanted a new environment for the AI to train in is so that the I can expand upon the potential actions and expand the unit roster. Starting off small with simple infantry units for the MLA get familiar with. The environment planned was, as detailed as before in the introduction, with 2 main areas of contention for the MLA to fight over. Control Points and Resource Points. One generating score while the other generated funding for both sides. Additionally for both players, there would be spawn points located on the opposite corners of each other where their respective units will be spawned.

There were three planned units for the players in the game to command, all with weaknesses and strengths to them. Normal infantry armed with firearms, grenadiers with grenades and lastly melee units with shields. Each unit would vary in utility on what the player/MLA could use. Formations are meant to be available as well, with the player/MLA being able to select which formation to use to gain a tactical advantage.

## Neural Networks and Unity’s Machine Learning Agent Toolkit

With the focus of artificial intelligence for the game, Unity’s ML-Agent toolkit was tested briefly on its effectiveness for use in the project. In addition to the toolkit, basic neural network architecture was also looked into for its viability in Unity’s scripting scene. For the toolkit and neural network, it would need to manage controls tailored from how a player would use them. Unit selection, camera movement and unit creation would need to be broken down into their base components to feed into the ML-Agent or neural network.

**Project Milestones**

Replace this text with Project Milestones.

Key project milestone dates and measurement on schedule, was project schedule adhered to, effectively planned for delivery on-time or ahead of schedule if appropriate.

**Project Review**

What went right? What went wrong? What (if anything) is still outstanding/missing (i.e., still left to do)? If starting again, how would you approach this project differently? What advice would you have for someone attempting a similar project in the future? Were your technology choices the right or wrong ones? If you chose the wrong technology, provide justifications for why you think this. What were the implications of your technology choices?

# Conclusions

Through the 7 months of working on this project, there were many mountains that were necessary to overcome before a solid threshold had been reached. That threshold was never reached as it lead to many problems both outside and during the development of the ML-RTS. The scope of the project was too vague in the beginning to create a concrete foundation for a suitable test bed for the Machine Learning environment. It wasn’t until after the Christmas break where a more cohesive idea was formed. Though at times, some goals were reached, I wouldn’t have considered them to be at a satisfactory level and to a level acceptable to my own standards. Much of the game is still left unfinished, with no finished gameplay loop or even the machine learning agent in the game. Though the Unity game engine was sufficient for this project, heavy optimisation was needed to increase performance and potential reduce training time.

**Future Work**

Indicate what might be some next steps to try (if a student next year was going to undertake a project in this area what might be an interesting thing for him/her to examine?).

# References

(n.d.).

Tencent AI Lab, China, University of Rochester, USA, cNorthwestern University, USA. (2018, September 19). *TStarBots: Defeating the Cheating Level Builtin AI in StarCraft II in the Full Game.* Retrieved from Papers With Code: https://arxiv.org/pdf/1809.07193v3.pdf

1. Micro: Controlling of units in skirmishing actions and individual battles. [↑](#footnote-ref-1)
2. Macro: The controlling of the strategic layer or “big picture” of the real time strategy game that involves managing resources, global troop movement/placement and general stratagem. [↑](#footnote-ref-2)