CREATING ARTIFICIALLY INTELLIGENT AGENTS IN MINECRAFT

Stuart Marples

Supervisor: Helen Hastie
Final Year Dissertation
Computer Science

SM861
H00156296

Declaration

I, Stuart Marples confirm that this work submitted for assessment is my own and is expressed in my own words. Any uses made within it of the works of other authors in any form (e.g., ideas, equations, figures, text, tables, programs) are properly acknowledged at any point of their use. A list of the references employed is included.

Signed: STUART MARPLES

Date: 24/11/2016

Abstract

Machine learning is a method for allowing a computer system to learn without being explicitly programmed which can be achieved through multiple methods such as reinforcement learning or supervised learning. Reinforcement learning is based off of psychological perspectives on human and animal behaviour to optimize an agent's behaviour in their environment. This is achieved by using past experience as a guideline in their environment allowing them to discover the worst and best actions in their surroundings. Supervised learning is the method of using training data and examples to find an optimal set of actions for a new task or environment. This is done by generalising the training data in a reasonable way to generate the best assumption for the agent to navigate through its environment.

Microsoft developed a new machine learning platform using the video game Minecraft as their virtual environment. This platform allows the playable character from Minecraft to be controlled by an AI that can be created by the user and can learn from its actions using various machine learning methods such as reinforcement learning or supervised learning. At completion of this project I aim to have used a machine learning method to complete a task and receive better results than the rule based systems that it will be compared to. The agent will then be judged objectively by comparing both the total reward and the time taken at the end of the task with the rule based systems results.

Throughout this report I will be discussing my experiment into great detail while discussing the possible benefits of using machine learning in games and in real life aspects. I will be discussing the benefits of using machine learning over rule based

systems and will look at the results of this experiment along with other similar papers to show the benefits.

Contents

1. Introduction

Throughout 2016 many Large companies such as Google, Microsoft, Amazon and Facebook announced their working with machine learning or backing companies that have machine learning as their main focus. Google recently made Tensor Flow open source which is a software library for numerical computations using data flow graphs which can be used for various projects such as generating music to discovering drugs. Microsoft created the Malmo platform which allows the use of machine learning in the virtual environment Minecraft and released this for use by the public in hopes of encouraging more people to work on developing artificial intelligence.

Working on the Project Malmo platform grants significant prospects for the world of gaming, education and with creating AI that can complete everyday tasks. For example, creating an AI robot to complete a task such as learning to walk up a hill would be expensive and impractical due to having to repair the robot anytime it failed by falling and breaking. Therefore, since Minecraft is a simulated environment it is the perfect platform for creating and testing this style of artificial agent because it does not cost anything each time you fail and allows for quick testing. It also allows for a more dangerous or risky set of circumstances to be tested due to the lack of consequences in the virtual world. This then saves money while also allowing for a more varied style of artificial agent to be created as various safety and monetary issues can be ignored when you are using Minecraft. Another advantage from using Minecraft is there are several situations that can occur in Minecraft that would not happen in life such as being attacked by zombies. This can increase the artificial agent’s knowledge base to cover situations beyond the normal scope of possibility making it able to handle more situations as they occur.

Gaming is a large industry in today’s world and is therefore always growing and improving. There are several examples of Artificial Intelligence already in games but there is always room for improvement as games are currently still very rule based. Project Malmo allows for the creation and testing of smarter and more independent artificial agents that could then be applied to various other games. A more data-driven approach in an artificial agent could improve a games ability to be replayed by supplying a more personalised version of the game to each player and their playstyle which allows for a more immersive experience.

Project Malmo also grants great opportunities for education because it’s a cheaper method of creating artificial agents than using anything physical that could break when a student is learning and allows anyone from a student to a professional to create an artificially intelligent agent. This means that students from around the world can start working with artificially intelligent agents at a younger age which allows for more people to be creating viable agents. With more people working in this field there is better chance of more household items becoming artificially intelligent for use in our daily lives. There could also be intelligent agents utilised in education for teaching and not just learning. If an agent is properly trained it could be used as an assistant in a classroom for various subjects like English or maths to help any students and ease the workload of the teacher. This could help in education as a teacher might struggle to help a class full of students but with an intelligent agent helping, there would be assistance for each student personally allowing the students to receive a stronger form of education through one to one tutoring.

My personal motivation for this project comes a natural interest in artificial intelligence and the gaming industry. Therefore, Project Malmo allows for a fantastic chance to work on and study these fields as I would get to work on intelligent agents while in the virtual world of Minecraft.

1.1. Aims:

The Aim is to create an artificially intelligent agent in Minecraft that will learn how to complete a task while attempting to achieve the highest score possible. The agent will be judged objectively against various rule based systems in terms of the number of apples collected, the final total rewards and time taken to complete the task.

1.2. Objectives:

1. To define tasks such as building objects like a wall or to find a way through an obstacle course to a desired location.

2. To define and create the arena for the task to take place in the Minecraft environment.

3. To find the best method of learning for the desired tasks after investigating various machine learning algorithms.

4. To compare the agent with rule based systems to view what methods work the most effectively and the differences between them.

1.3. Constraints

The agents available space is the 20x20 arena in which it is placed along with the enemies and the collectable apples. The agent will be given 60 seconds to complete the task before the task is ended and the appropriate details are stored about the run. The agent will complete the task when it collects all of the 20 available apples or dies to any of the enemies in the arena.

2. Background

2.1. Introduction

Creating artificially intelligent agents in games has been around for multiple years but making an artificial agent using a restrictive rule based system is still the widely-used method. If other methods like machine learning were utilised in games, then an artificial agent could learn how to play the game itself and adapt to any situations that occur. There have been many successful attempts at creating artificial agents in games using machine learning methods such as reinforcement learning or supervised learning, such as the use of reinforcement learning to play Super Mario Bros by Liao et al. (2012) discussed in section 2.2.1, but these methods are not widely used. Minecraft becomes one of the best environments for creating and training an agent due to its large worlds, adaptability and the creativity available from the game.

2.2. Machine Learning

Machine learning is a type of artificial intelligence that allows a computer to learn without being programmed directly. Machine learning will grant the ability for a computer program to teach itself and change when exposed to new data. Examples of machine learning techniques are reinforcement learning, supervised learning and deep reinforcement learning which will be discussed below.

2.2.1. Reinforcement Learning

Sutton & Barto (1998) discuss the basics of reinforcement learning and the issues while also comparing it to supervised learning which will be later discussed in the following section. Reinforcement learning focusses on a learning problem and not a learning method which means that any method that is suited to solving a problem, can be considered a reinforcement learning method. Reinforcement learning is based off behaviourist psychology where the agent will receive a cumulative reward each time it takes an action. This is shown in Figure 1 as each time the agent makes an action in the environment it receives a reward and move on to the next state.

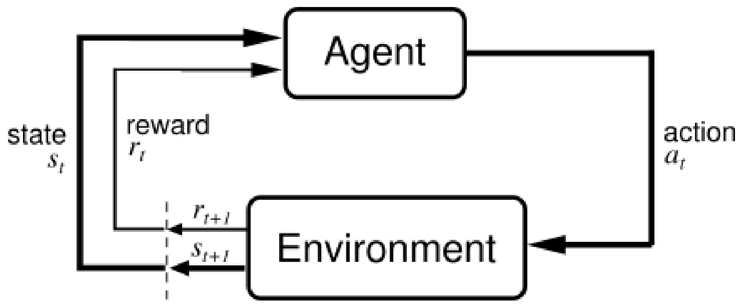

Figure 1: Reinforcement Learning Model

The reward is not always positive as the agent needs to know when it is doing something that it should not do to discourage the action from happening again. This makes reinforcement learning a good method for creating an artificially intelligent agent in Minecraft because the Project Malmo platform allows the agent to take in the information from its surroundings which makes it easier to implement. Sutton and Barto begin to briefly talk about various examples of reinforcement learning in games such as chess but there was more relevant information available in the Super Mario Bros. example discussed below (Sutton & Barto 1998, p.4 – 6).

Liao et al. (2012) programmed an agent using reinforcement learning that would play the game Super Mario Bros. They utilise ε-greedy Q-learning which is when the agent chooses the action that has the best probability of completing the long-term problem. To find the optimum method they use a Q value which is made up from the rule:

$$Q[s][a] = (1 - \alpha[s][a])Q([s][a] + \alpha[s][a](R + \gamma \max(Q(s',a'))) \qquad (1)$$

This is made up from four main parts: current state, chosen action, reward and future state where *R* is the reward, α is the learning rate and γ is the discount rate. The reward given to the agent is made up from progress forward, upward and from killing the enemies, the discount factor was chosen to be 0.6 and the learning factor decreased over each iteration. They trained the agent for 5000 iterations of the game which when completed created a win rate of ~90%. The reward function that was utilised in this project could be used for certain movement based tasks in Minecraft such as navigating around a maze. But due to Minecraft's complexity it would be difficult to apply this rule to other tasks because of the various actions that are available in the game (Liao et al. 2012).

2.2.2. Supervised Learning

Alpaydin (2014) discusses supervised learning into great depth as well as various other machine learning methods. He describes supervised learning as finding the positive and negative common features in a set of items and then being able to predict what else might be in that set of items by looking for the positive common features. This can be seen through the Supervised Learning model shown in Figure 2 where the training data can be used to create a classifier when analysed and applied to any input data to receive a prediction for the output. Or you could do knowledge extraction which is when you extract the information to discover what the common features are. I could apply this to this project by creating data that shows what some of the good and the bad actions are for the agent to take (Alpaydin, E. 2014 p.21 – 24.).

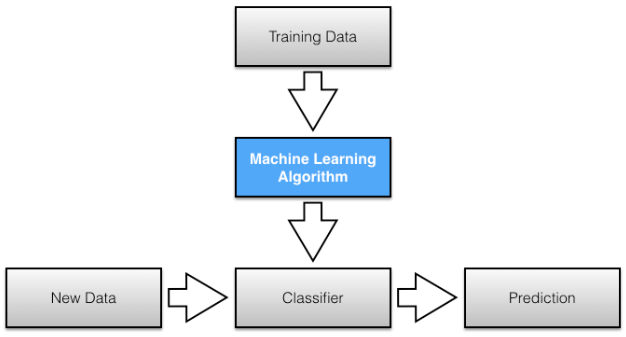

Figure 2: Supervised Learning Model

Kotsiantis (2007) discusses the various ways of implementing supervised learning methods including Decision Trees, a Neural Network(NN) and a Bayesian Network(BN).

In decision trees, each node of the tree is a decision on an attribute for example, if a coin is heads or tails, and each branch of the tree represents the outcome of the decision and the leaf nodes are the final decision after going through the entire tree. Decision trees can be translated into a set of rules where each rule is a single path from the root of the tree to a leaf. Decision trees could be used in this project but due to the trees size there would have to be a fast way to traverse the various branches in the tree.

Neural networks are a Perceptron based technique which means that the network will have a list of input values which will have a weight assigned to the link between the nodes and supplies an output which is designated based on the sum of the weighted inputs. In the creation of a NN it is difficult to correctly set the parameters for the parameters such as the size of a layer as this can negatively affect the performance of the network if a layer is too large or too small and requires proper testing to find the correct amount. Neural networks appear to be a good method of implementing supervised learning into the project but will require the appropriate time to test frequently to ensure that the running speed is decent while also finding the appropriate sizes for the various parameters in the NN. This means that due to the time restriction of this project that a Neural Network may not be a possible option despite the very beneficial features of utilising one.

Bayesian Networks are a graphical representation of the probability relationships among a set of variables. This assigns a conditional probability table to each node in the network which will take the values from the parent nodes and assign a conditional probability to the node. An issue with BN’s are that they are difficult to implement when there is a dataset with many features since this would increase the time and space required to construct the large network. This issue makes it an unlikely method for implementing supervised learning in this project due to the complexity that is involved in Minecraft (Kotsiantis, S. B. 2007).

Box (2014) discussed using supervised learning in a game to train a traffic control system. The method was to use a player as the expert who would use their strategies to win the game and then have the supervised learning algorithm learn from their strategies. The game would prompt for the player to change the lights in the game every 10 seconds and the player could choose to do nothing or make a change which becomes advantageous for generating training data but was held back due to the time limit as other control systems such as Microprocessor Optimised Vehicle Actuation (MOVA) do not have the time restriction. Using supervised learning in a game method like this could help for creating more advanced systems in the future as it allows for a system to learn from possibly multiple experts through a game format where an expert may get better results due to more interest in the system. This would create more sophisticated and intelligent systems capable of completing existing jobs more efficiently or being able to attempt harder or riskier jobs such as air traffic control instead of a regular traffic control system (Box, S. 2014).

Kotsiantis (2007) discusses various machine learning methods including supervised learning. He talks about some of the issues with supervised learning algorithms and why they are not so widely used anymore. To create a supervised learning algorithm, you need to have an expert to supply the required knowledge as data, otherwise it becomes a brute force method which takes everything available and then hopefully isolates the relevant features which is known as unsupervised learning. For this project, supplying the program with the information it needs should not be a problem due to Project Malmo already supplying most of the data ready for use and have previous experience with Minecraft for my own knowledge which would allow me to generate the required targets for the algorithm (Kotsiantis, S. B. 2007).

2.2.3. Deep Reinforcement Learning

Mnih et al. (2015) discuss how they created a reinforcement learning function that learned how to play 49 different Atari games. They managed this by utilising a Deep Q-Network which utilises reinforcement learning with deep neural networks which is when several layers of nodes are used to build more abstract representations of the data. This is visually shown through Figure 3 where the pixels of the screen and rewards are input into the network and the output is the action for the game. This means that this method works very similarly to a normal reinforcement learning method where they use the Markov Decision Process which consists of rewards, actions, current and future states but then feeds the input data into the neural network to acquire an optimal output which could be used as an action.

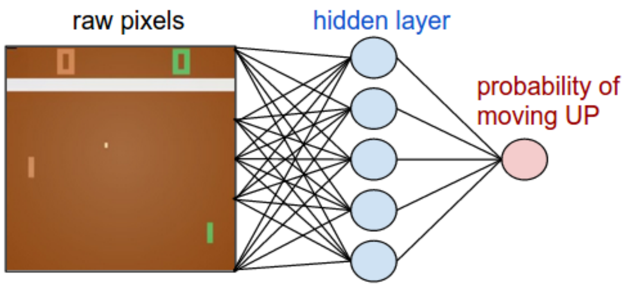

Figure 3: Two layer fully connected Network

A neural network would be useful for this project as it can be highly beneficial to learning but only if there were enough nodes to be added to the network as actions that the agent could take which I believe would not be the case with Minecraft. Although it has a lot of actions that the agent could choose from, I do not have the time to implement a neural network while handling the various issues that could cause instabilities in the project related to the learning. These instabilities are discussed in the paper and can cause incorrect behaviour in the agent unless they are correctly handled (Mnih et al. 2015).

2.2.4. Summary

From the discussion above we have found that using Deep Reinforcement Learning could be a very good way to create the agent but requires more time than is available and therefore cannot be utilised. This leaves Supervised learning and Reinforcement learning as the remaining viable options which can be both be used and can both achieve the required aims and objectives. Supervised learning can be utilised because Project Malmo already supplies all the required knowledge necessary to implement the method but would need the relevant training pairs to be added to create the supervised learning system. Reinforcement learning can also be utilised due to its reward based system making it perfect for the task based objectives. This information has led to the creation of the reinforcement learning method as the supervised learning method would take longer to extract the relevant information which allows less time for work on the agent and its learning.

2.3. Gaming in AI

Treanor et al. (2015) discuss a new model for designing games around Artificial Intelligence by using the AI as the main player experience instead of using AI as a supporting role. Some of the examples discussed are as old as chess where the AI plays as the opposition but other more recent games include The Sims where the AI are guided by the player but are largely independent or Alien Isolation where the AI is the alien that acts as the villain of the game. In Alien Isolation the AI would learn more about the player as the game goes on and learn about how they play the game, how they manoeuvre around the map and would use this information to create tension and build to a more unpredictable horror game for the player. This would then allow for an enriched gaming experience because in Alien Isolation the player cannot know what the AI is going to do at any point of playing the game creating a consistently tense environment to entice the player while a rule based system would eventually get predictable and lose any players emotional attachment to the game. My project aims to create an AI that could be used in similar ways to the examples shown where it learns how to interact with its environment and learns about what is around it which could be adapted into a similar situation to the Alien Isolation example for a better player experience (Treanor et al 20015).

Ram et al. (2007) discuss the various challenges and research opportunities in AI with the aim of creating a significant impact throughout the gaming industry. They also talk about how gaming does not need to be constricted to just entertainment and can be applied to various other applications such as humane gaming applications for medical training or educational games. Some of the challenges discussed through the paper are the replay ability and variability, unanticipated situations and knowledge engineering. These challenges cover having the artificial agent react to different player's playstyles and to react accordingly to the various situations that could occur. This work is relevant because in this project the artificial agent should learn how to play and react to the various elements in Minecraft to achieve a goal which could be applied to other games to help solve these challenges (Ram et al. 2007).

Rhalibi et al. (2009) discuss how video games offer an interesting testbed for research into Artificial Intelligence. They delve further into how games provide a stable, rich and complex environment in real time to encourage fast decisions but also are mainly created using simple rule-based systems. Machine learning techniques are rarely used in state of the art computer games even though they could enable non-playable characters with the ability to improve their performance by learning from mistakes, could adapt to the strengths and weaknesses of a player or learn from the way a player plays the game. There are multiple examples of people who have utilised games to enhance various computerized processes shown throughout the remainder of this journal. At completion of my project I hope to have created an agent in Minecraft that will learn from mistakes and adapt which could be adapted for another games usage (Rhalibi et al. 2009).

Eladhari et al. (2015) discuss the games “Contrabot” and “What Did You Do?” Which uses AI in inventive new ways for games. “Contrabot” is a smuggling game where the player is marking crates to communicate with a smuggler on the other side of a check point while the AI is acting as an inspector who occasionally checks the crates for any marks. The player receives a point for every crate they get past the inspector and attempts to get a high score while the inspector will remember every mark they have looked at to try and find the patterns from the player to correctly guess which crates are being smuggled. “What Did You Do?” Is a game that utilises an AI similar to a child that learns from the player who acts as the parent and their actions in the game. The game is a turn based game where the player tries to make their way through a grid based map collecting health and avoiding obstacles to reach the destination and as the AI attempts to mimic the player’s behaviour the player has to try and influence the AI into making the correct movements to achieve the same goal. These examples show a different style of using AI for games as the AI are playing a main part in the game but are being trained by the player and only get better through the training from the player for each individual game. This shows that creating AI in gaming could be more than just improving the current genres in gaming but could actually add a new genre of games completely (Eladhari et al. 2015).

2.4. Minecraft

Levin (2011) created a website to help various educators use Minecraft to educate their students. The site shares modifications that you can download and apply to Minecraft for education in Quantum Mechanics, Computer Programming, Space Science and Cross Curricular.

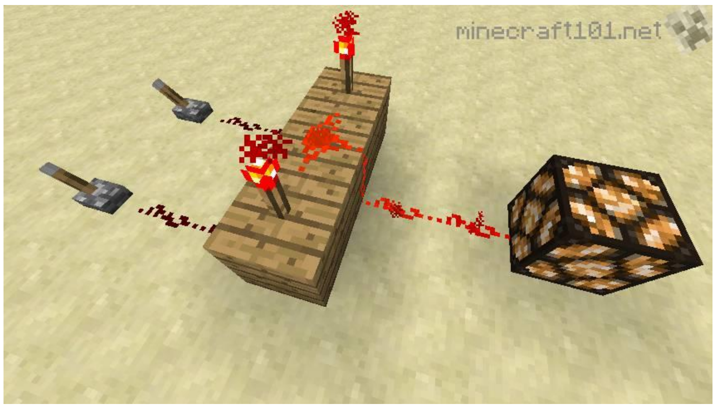

Figure 4: Logical AND Gate in Redstone

Some of the most complex parts of Minecraft involve using Redstone to be able to create many different types of circuits which can be used to help teach logic gates and can be used for other similar educational purposes. An example of using a Redstone AND gate to turn on a light is presented in Figure 4. There are also multiple real world examples shown on the website where teachers utilize these modifications and create a fun way of teaching these subjects. An example of a real-world situation is shown where there is a modification for teaching mathematics being used in Minecraft. This shows that Minecraft is a good environment for people to learn from because of its creativity and the number of available actions you can do while using it which also makes it a

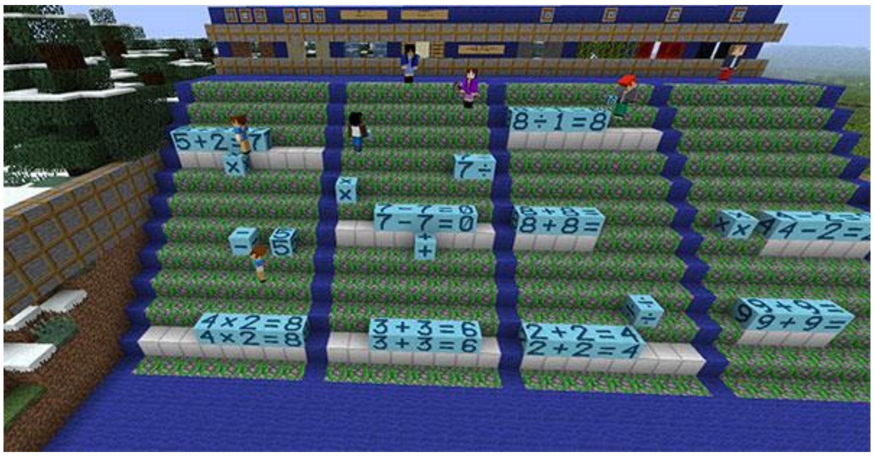

Figure 5: Mathematical Modification in Minecraft

great platform for creating a various number of tasks for teaching the agent. The disadvantage of using Minecraft is that students can get distracted and lose focus from what the class is trying to teach (Levin, 2011).

Due to Minecraft's freedom, environment and countless items the agent would be able to learn to deal with multiple situations after being trained in the environment as there are various in game features that can be used or modifications to the game such as the modifications discussed above. This could allow for an easy, cheap and fun method for developing artificially intelligent agents through Minecraft as the virtual environment over various other simulations that might be used only for one type of situation or task.

3. Requirements/Research Questions

I have determined a set of functional and non-functional requirements that have been created by thinking about the aims and objectives of the project. These requirements have been accompanied with a Completion category which will contain either "Completed", "Partially Completed" or "Not Completed" with an accompanying reason if they are only partially or not completed. These have then been displayed in separate tables with a priority and category assigned to each requirement and are ordered by displaying the Must Have items first.

3.1. Functional Requirements

Num-ber	Description	Priority	Category	Completion
F1.	Up to three rule based systems will be created to run the task for comparison with the Reinforcement Learning agent.	Must Have	Artificial Minecraft Agent	Completed
F2.	The artificial Minecraft agent will use Reinforcement learning to learn about its environment.	Must Have	Artificial Minecraft Agent	Completed
F3.	The artificial Minecraft agent should be able to run a specific task on command, such as finding its way through an obstacle course.	Must Have	Artificial Minecraft Agent	Completed
F4.	The artificial Minecraft agent should be able to judge from the environment how to handle a situation.	Must Have	Artificial Minecraft Agent	Completed
F5.	The correct environment must be generated on command.	Must Have	Environment	Completed
F6.	The artificial Minecraft agent will learn whenever it is being used and not just during training.	Could Have	Artificial Minecraft Agent	Partially Completed
F7.	The artificial Minecraft agent can be interacted with by the user to influence its learning.	Could Have	Artificial Minecraft Agent	Completed

Table 1: Functional Requirements

The Function Requirement F6 was only partially completed as it was possible to make the agent learn whenever it is in use, but would require the manual input of the most recent q-table into the program to continue learning from its most recent iterations.

3.2. Non-Functional Requirements

Number	Description	Priority	Category	Completion
NF1.	The artificial Minecraft agent will be created using the language Python.	Must Have	Performance	Completed
NF2.	The artificial Minecraft agent will be tested in the virtual environment Minecraft.	Must Have	Performance	Completed
NF3.	The artificial Minecraft agents training data will be stored separately from the code for later use.	Must Have	Performance	Completed
NF4.	Each task attempt will timeout after the artificial Minecraft agent has attempted the task for a certain amount of time.	Must Have	Performance	Completed
NF5.	Each task can only be run one task at a time.	Must Have	Performance	Completed
NF6.	The artificial Minecraft agent can be loaded to various locations and environments in Minecraft.	Must Have	Performance	Completed
NF7.	Minecraft will generate appropriate environments for the various tasks.	Must Have	Performance	Completed
NF8.	The agent will be robust and will handle any situations that can cause an error.	Must Have	Robustness	Completed

Table 2: Non-Functional Requirements

3.3. Research Questions

Since this project is part of new research into project Malmo and creating artificial agents in Minecraft, there have been multiple questions about this topic that should be answered before continuing this project. These questions are then displayed and answered below:

3.3.1. Is using a machine learning based system better than using a rule based one?

There are multiple reasons that using machine based learning can be better than rule based ones. For gaming, a rule based system is only as creative and flexible as the people creating it which means that it cannot possibly cover all the situations and playstyles that can occur in a game. A machine learning based system will be able to adapt and learn from new situations which can change the way the game reacts to the player's playstyle. This can improve a game substantially as it allows for spontaneity in what the artificial agents do while also responding appropriately to the situations which makes a game immersive and increases the replay value of the game.

For real world applications creating a rule based system can be easier than creating a machine learning system. Karthik Guruswamy (Guruswamy, 2015) discussed the differences between rule based and machine learning systems and said "Rules based system will work perfectly 'IF' you know all the situations under which decisions can be made." Which means that to use a rule based system you need to think of every and any situation that could occur. It also means you should keep the rule based system up to date by possibly adding more rules the longer the system is in use while a machine learning system can be trained and then has the various experiences that allow the system to react appropriately while also learning over time with new experiences. A disadvantage to using machine learning is that it can take a long time to initially train a system to guarantee appropriate reactions and safe use.

3.3.2. Is it possible to emulate human behaviour with machine learning techniques and is this the most effective way?

Emulating a human's behaviour can be very challenging but Moriarty et al. (2009) discussed learning human behaviour from observation from gaming applications and say that the best method of learning is through observation as that is how children as young as two years old learn about the world. They talk about how machine learning can be useful in the creation of non-playable characters but they can develop bad habits such as observing that something only happens 5% of the time and learning to never do it instead of learning to do it rarely. They also go on to talk about how machine learning can be used for this but can come into issues if the game is too complex due to the amount of processing that would need to be done. A way around this was to only process the information that was needed which made this a much more effective way of emulating human behaviour.

3.3.3. How effective is the Minecraft environment for this task?

Minecraft is a great testbed for the creation of artificial agents due its vast detailed landscapes and the creativity that is available in the game. Due to this creativity, I can create various tasks and situations that the agent will learn how to deal with. A benefit to using the Minecraft environment has been described by the director of artificial intelligence outreach at Microsoft research, Evelyne Viegas (Viegas 2016) who said “We’re looking for opportunities where we can really help accelerate the pace of artificial intelligence innovation in a way that is going to be very close to the real world, with real experiences and real data” which describes Minecraft as a good environment due its similarities to the real world. This makes it effective as using Minecraft is very close to the real world for data but doesn’t include the monetary cost or restart time that the real world could have for the failed attempts.

4. Methodology

For this project, a task has been defined and run after creating a reinforcement learning agent to test and compare with multiple rule based systems. The task, training and method used to train the agent will be discussed in the following sections along with what rule based systems have been created and some of the issues that were encountered while using the Malmo environment will be discussed.

4.1. Task

For this task, the agent would have to try and collect all of the 20 available apples in 60 seconds and would try to maximize its final score. The agent was placed in a 20x20 block arena where apples and enemies are also placed dependent on what parameters the task is utilising.

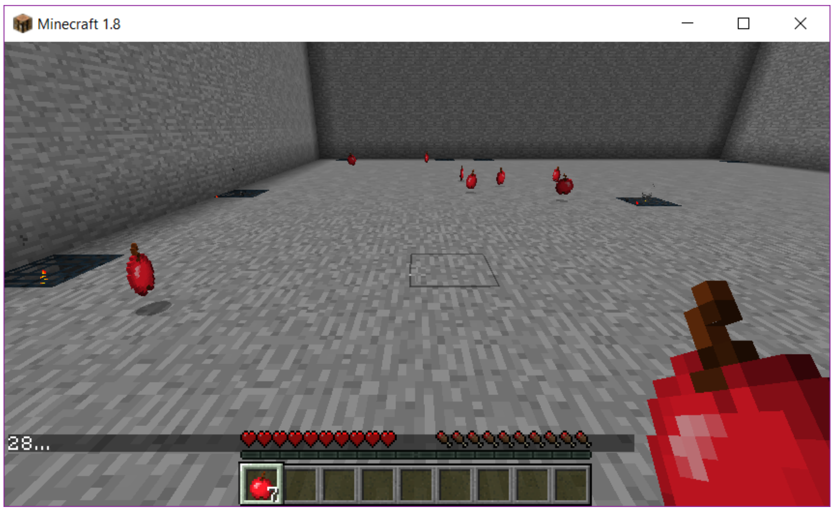

Figure 6: The arena while the agent attempts to collect the apples

Due to the high speeds of the task and the rapid turning actions I utilised a visual representation of the task to allow for easier viewing of the agent. This can be seen through figure 7 which displays the boundaries of the arena while showing the enemies as red dots, apples as blue dots and the player as a green

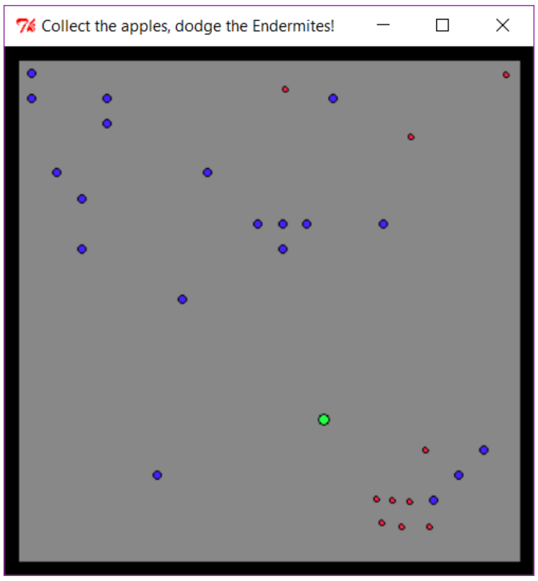

Figure 7: Visual representation of the overhead view of the arena

dot. This allowed me to witness the progression of the agent while also having a bird's eye view for the actions it is making to go towards the apples or away from the enemies which helped for testing and improving the way the agent learned.

The 2 different parameters that were defined for this task are:

1. If apples would be statically placed in the same position for each run or the apples would be randomly placed around the arena for each run.
2. If enemies would be able to spawn or would not be able to spawn.

At the end of the task the number of apples collected, the total score earned and the time taken for completion were recorded for comparison with rule based systems.

4.2. Rule Based Systems

The three rule based systems were created as baseline systems for comparison against the reinforcement learning agent. These 3 Rule based systems are:

1. "Random" which would make completely random actions with no regard to the environment around it.
2. "LawnMower" which would travel to a corner and attempt to travel through every block in the arena by travelling up or down and only moving left or right when it cannot move any further up or down.
3. "Mob_fun" is the most sophisticated system which would use heuristics similar to the reward system discussed later in Section 4.4 to judge every action possible and choose the action with the highest value. This system could only see in a two block radius around itself allowing it to turn away from enemies similarly to the agent.

4.2.1. Environmental Challenges

There were a few initial challenges with creating this task as the enemies cannot be individually placed in a certain position which led to the use of spawning blocks. A spawning block can be placed into a certain x, y and z position and can spawn a certain type of enemy but cannot control the number of enemies that spawn, how long they spawn enemies for or the radius in which the enemies spawn around this block. This led to inconsistencies through the training and experimentation as sometimes the agent could be swarmed by up to 15 enemies immediately and die or only 1 or 2 could initially spawn and create little issue for the agent.

The lack of control over the spawn blocks also created issues when attempting to use zombies as the enemies due to the various types of zombies that could spawn. Some of these zombies could move at faster speeds than the agent or could deal more or less damage than the other zombies, which caused issues with the movement and the survivability of the agent. This was fixed by spawning “Endermites” instead of zombies: an enemy that deals consistent damage and moves at a consistent speed. Although Endermites are slightly weaker which allowed for the simpler rule based systems to run through groups of them and not die which would not have occurred when using zombies.

The Malmo software allows the user to speed up the simulation from its normal speed up to 50 times as fast. Although due to the action base of the agent, the software could only be sped up by 2.5x the normal speed because the agent is constantly moving and only turns based on the action and unfortunately any faster caused the agent to spin in circles and make incorrect actions. The task could not be ended upon collection of the 20 apples, which led to a time limit being added to the task which would store the time and score of the agent once it had collected all of the

20 apples. This led to training times taking longer than originally expected due to the extra time spent waiting after completing the task.

Unfortunately, as a university student the required resources are not fully available for use through this project compared to similar projects done by Mnih et al (2015) where they utilise dedicated machines for training their agent. Due to using a personal computer the amount of time spent training the agent began to slow down the computer which would create some problems through starting the task. The agent began to have a delayed start where it would make an action a few seconds late while the enemies had already begun to move towards it which would create inconsistencies in the results and would usually result in an early death by the agent. This was mostly resolved by allowing some time for the computer to be shut down between the training runs but could be seen towards the end of some of the longer training runs.

4.3. Training

The tasks that did not involve enemies were trained over 2000 iterations while the tasks with enemies were trained over 10,000 iterations as the tasks with enemies needed more training for improvements to occur with the agent.

The agent was run at 2.5x the regular speed in efforts to accelerate the training process to a reasonable speed but would still take 13 hours to train the agent over 2000 runs. The rewards as described in Section 4.4 and the Q learning function were the same throughout all of these runs to allow for consistent output for comparison. Through the training, the best final total reward was stored along with the Q-table that resulted in this reward so that this Q-table could be run again later to create a small set of comparable results.

4.4. Q-Learning

By utilising Q-Learning, a Q-value is created for each state action pair in the Q-table based off the rewards earned through making an action at each state.

The set of available states (*S*) for the agent are the *X* and *Z* positions for each block within the 20x20 block arena. The current state(*s*) is stored as the whole number value of the agent's current *X* and *Z* position in the world. The next state (*s'*) was looked at to predict future rewards and to find the best future Q-value action to update the current Q-value.

The agent is moving at a constant running speed through the task and the action is chosen from a set of actions(*A*) which contains 10 different degrees of turning. Each action(*a*) is to turn every 36 degrees up to 360 degrees to allow for a full turning circle. A preferred action set would have been larger to allow for more accurate turns but due to the amount of time available to train the agent the larger action set was not viable as the training could not be sped up to a significant speed and a larger action set would have increased the training time tremendously. The smaller action set worked well enough as it allowed for full movement by the agent but did deny smaller turns which could have made the agent avoid enemies easier or found more optimal turning angles. The agent used a greedy algorithm where it would choose an action based off of the highest Q-value rewarded action from its current state. If there were multiple Q-values at the highest value available, the agent would choose an action that did not lead to an enemy at random.

The agent receives reward(*R*) based off of each action from the current state to the next state which can be seen in Table 3. The agent would be positively rewarded for collecting an apple or for collecting all of the apples. The agent would be negatively rewarded for running into the walls at the edge of the arena, taking damage, dying, stepping on the block used to spawn enemies and for each action the agent takes.

Rewarded Action	**Reward Values**
Collected one apple	+200
Collected all apples	+1000
Injury	-50
Death	-1000
Walking into wall	-5
Walking over an enemy spawn block	-50
Agent action	-1

Table 3: Reward Values

The reward system was tested in depth towards the beginning of the project in an attempt to find the best relationship for the agent. Previously the walking into walls reward was lower at -50 but it was quickly seen that the system would process this reward multiple times while the agent was at the same state which led to the change to -5 as a small value that would accumulate in this way to a larger negative value. Originally the agent would focus on survival over collecting apples which led to the change for the apples reward to increase from +100 to +200 while the pain reward was changed from -100 to -50 to attempt to encourage the agent to be more adventurous with collecting apples near enemies.

4.4.1. Q-Learning Function

Using the states, actions and rewards, I initially updated the Q-value in the table using Learning Function 1:

$$Q[s][a] = Q[s][a] + \alpha(R + \beta * \text{MAX}(Q[s']) - Q[s][a]) \qquad (2)$$

The learning factor is shown as α to represent the amount the agent will learn from its new Q-value compared to its old Q-value. This was set to 0.1 as a value closer to 1 is better for deterministic environments, which is only true for this task when the apples are static with no enemies. The discount factor is displayed as β and determines how much the agent focuses on future rewards. This was set to 0.95 as we wanted the agent to look for the best future rewards which would be the areas that contain apples.

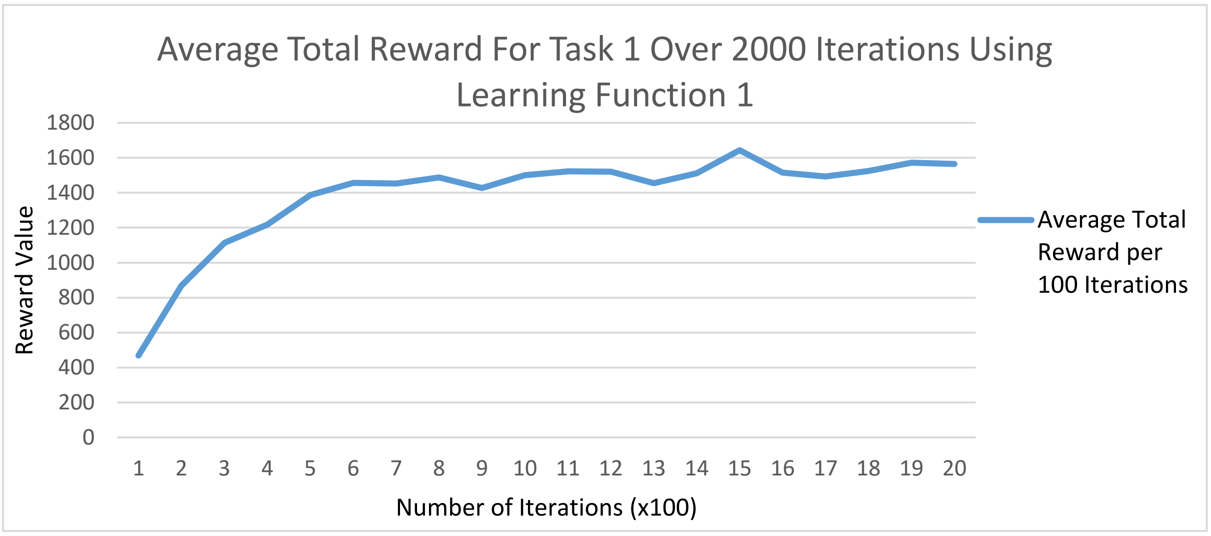

Figure 8: Training Task 1 Over 2000 Iterations Using Learning Function 1

This resulted in a learning curve over the 2000 training runs on static apples with no enemies shown in Figure 8. The training ended by collecting an average of 14 apples and only completed the task 6 times after reaching the peak of the curve at 600 iterations in. The average score was lower than expected and could be improved so I began to change Learning Function 1 which resulted in an average of 15 apples being collected and 17 completions from the same peak of 600 in Learning Function 2 shown below:

$$Q[s][a] = (1-\alpha) * Q[s][a] + \alpha(R + \beta * \mathrm{MAX}(Q[s'])) \qquad (3)$$

Where:

$$\alpha = \frac{1}{1 + \mathrm{Visits}[s][a]}$$

Visits keeps a count of how many times the agent makes the same action at the same state and updates the Q-value based off how many times they have done this. This was to try and encourage the agent to be more adventurous as the previous learning function did not make the agent venture out fast enough as it would circle areas where it collected apples previously.

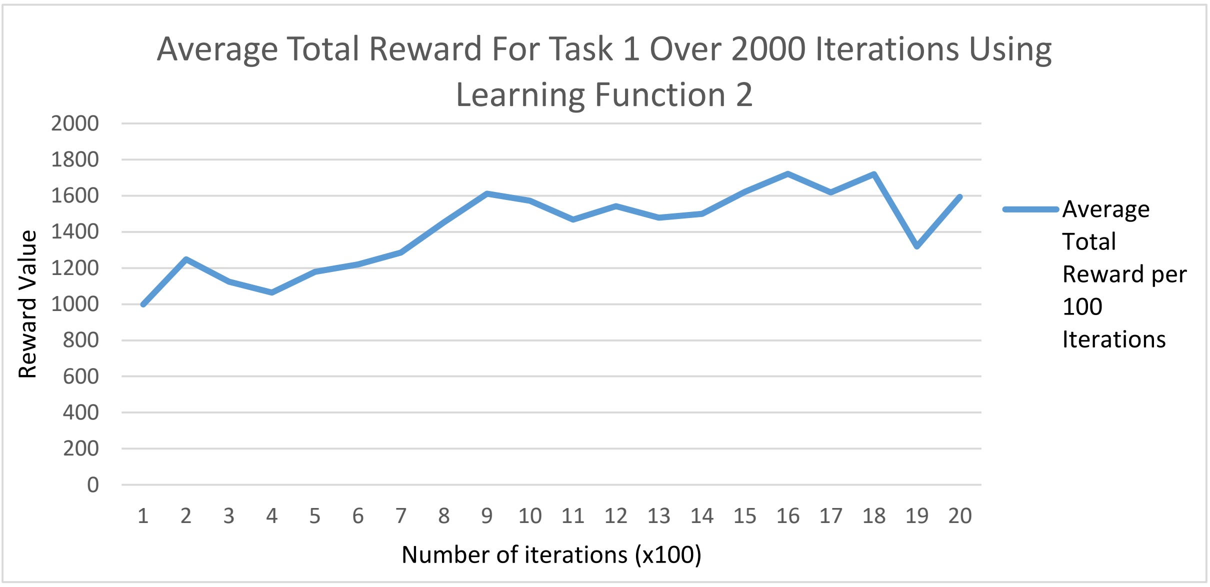

Figure 9: Training Task 1 Over 2000 Iterations Using Learning Function 2

Using Learning Function 2, I received the graph shown in Figure 9, which has less of a curve compared to Figure 8 as it started at a higher value and did not learn as rapidly but did receive slightly better results and eventually when tested on the task with

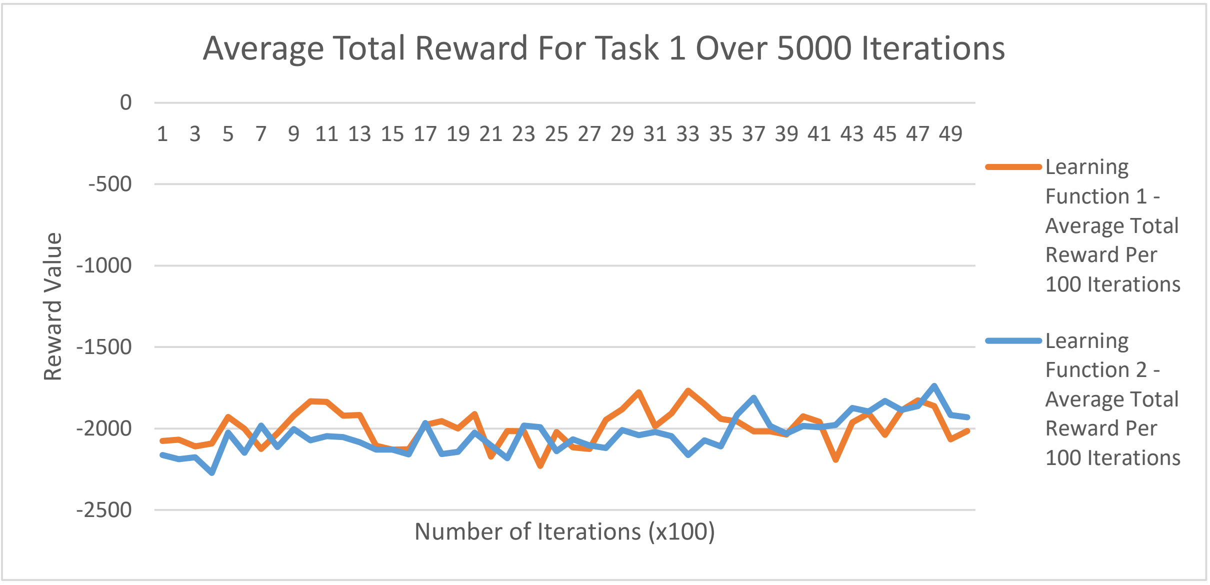

Figure 10: Training Task 3 Over 5000 Iterations Comparing Learning Function 1 and 2

enemies and statically placed apples the agent managed to improve the average reward by 200 and the average number of apples collected by 2 while Learning Function 1 did not improve at all and the average reward varied along the -2000 line only improving by 50 which can be seen in Figure 10. After this comparison Learning Function 2 was run a Further 5000 times and a more obvious but gradual improvement could be seen through Figure 11.

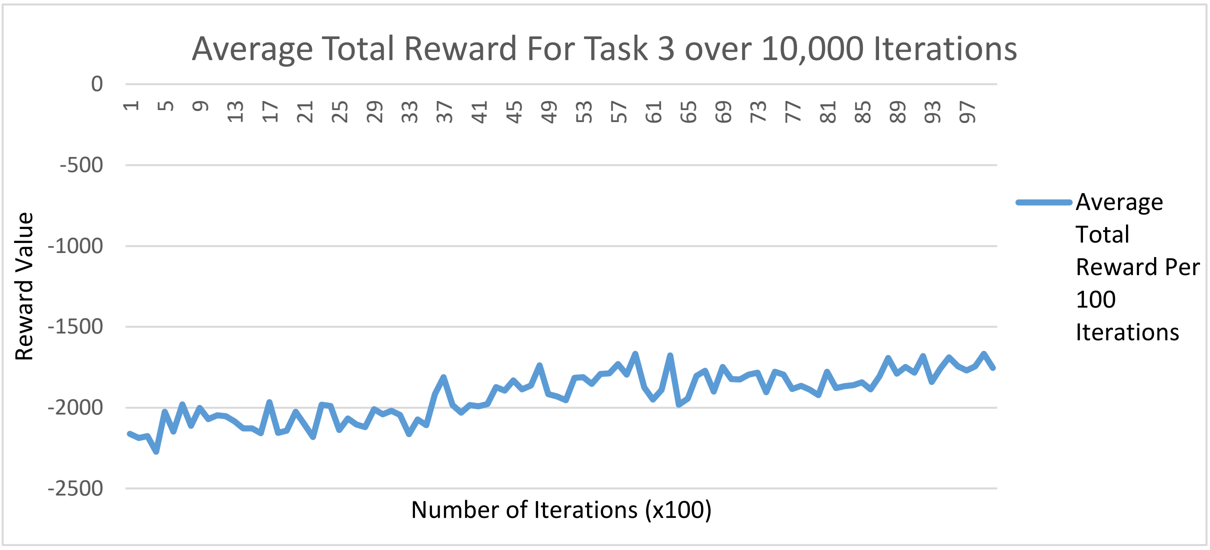

Figure 11: Training Task 3 Over 10,000 Iterations Using Learning Function 2

This meant that for training, testing and comparison I continued with Learning Function 2 as it resulted in a higher completion rate, score and number of apples collected. Unfortunately, I did not have the time required to run the tasks that have enemies longer than 10,000 runs despite the fact that they look like they are still improving as the training time would be too long.

4.4.2. Improvements to Speed Up Training

Due to the extensive training time I added some features to attempt to decrease the amount of time the agent took to train.

Originally the size of the arena was supposed to be larger as a 60x60 block arena to allow for more room to manoeuvre around the enemies and to create more challenge in finding the apples. Unfortunately, this was scaled down as the agent would need a more time than what was available to be trained properly. This led to the rule based system Random receiving higher results because there was less area for these system to search allowing it to occasionally stumble into large groups of apples instead of possibly just a single apple in a larger area.

To stop the agent making an initially terrible move that could lead to death I started it by using heuristics similar to the rewards received through the task. This allowed the agent to find one of the best actions immediately which would turn the agent away from enemies and towards the apples. This stopped the portion of the training where the agent would learn to not move in a certain direction from the start as it would lead to enemy spawn locations or just areas of no importance.

Through using Q learning the agent is supposed to learn through experience made through actions in states but I added an extra form of observation to attempt to improve the training time of the agent. I added a grid based observation system to influence its action based on the presence of enemies in the blocks around it. The agent would make a list of the highest Q-value actions from its current state so that it could randomly choose an action from this list as its final choice action. The agent would be able to see if an enemy was within two blocks away from it and would try to take an action from the list that would avoid coming into contact with these enemies unless this was the best option.

5. Evaluation Method

On completion of the training, I had utilised reinforcement learning to create an artificially intelligent agent which was trained in a specific task and could then be used for comparison with the rule based systems. From this point, I have trained each agent 2000 times for Task 1 and 2 and 10,000 times for Task 3 and 4 and have taken the objective measurements from each task. This contains the final total reward, the time taken to complete and the number of apples collected for each run of the task. This data is then displayed in a graph to visually represent the rate of improvement throughout the agents training and to find the best run to collect the best Q-table for use in further testing and comparison with the rule based agent.

Once the optimal attempt has been found, the agent will use the Q-table that resulted in this attempt and run the task a further 30 more times for comparison with each of the rule based systems which will also be run 30 times on the same task. The total reward, total time taken and number of apples collected for each of these runs will be used to compare the reinforcement learning agent and the rule based systems using tables to visually represent the data. This data was accompanied with a completion rate which is the percentage of the runs that the systems managed to collect all 20 available apples before the time limit ran out. A death rate was included for Task 3 and 4 to show the percentage of runs that the systems managed to survive through to the end of the time limit.

6. Results

After collecting the four separate best Q-tables discussed in Section 4.3, the agent was run another 30 times using these Q-tables and were compared with the various rule based systems which were also run 30 times.

The data collected from these runs will be displayed in a table as the averages and standard deviation of the time taken, number of apples collected and the finishing reward value while also displaying the completion rate and death rate over the 30 runs. The best values in the table will be highlighted to allow for easy viewing of the best system for each result.

The agent was allowed 60 seconds to complete the task but due to the 2.5x time increase the agents recorded finishing time was 24 seconds after waiting the full 60. This means all the time related data is between 0 and 24 to represent the full minute they spent in the task. Through task 1 and task 2, the lowest average finishing time would be the best because the systems cannot end unless they collect all of the apples, which means a smaller time means they have managed to complete the task faster. Through task 3 and task 4, the higher average finishing time would usually be better as the systems would not manage to live until the end of the 60 second time limit and would not be able collect all of the apples to finish the task early.

6.1. Task 1: No Enemies with Static Apples

Throughout the first task the agent was expected to receive good results due to the environment not changing allowing the agent to mark out and remember where the apples are and would be able to avoid the negative rewards. But due to the tasks simplicity the Mob_fun system was expected to do very well in comparison to the agent.

	Apples Collected		**Total Reward**		**Time Finished**		
	Mean	**Standard Deviation**	**Mean**	**Standard Deviation**	**Mean**	**Standard Deviation**	**Completion Rate**
Agent	**16.50**	2.09	1373.3	734.03	23.79	1.27	**7%**
Mob_fun	16.40	2.29	**2159.2**	662.18	**23.75**	1.26	**7%**
Random	10.53	3.06	-2033.6	1006.20	24.09	0.04	0%
Lawnmower	14.83	**1.21**	1162.1	**292.50**	24.09	**0.02**	0%

Table 4: Task 1 Results

From the results shown in Table 4, the agent collected the most apples on average while also having the highest completion rate along with the Mob_fun system similar to expectations but the average score of the agent was lower than expected. The agent otherwise finishes second to the Mob_fun system with average rewards and a close average time. The lawnmower system achieved the most consistent results as the environment did not change and the system moves along a set path allowing for similar times, apples and total reward at the end of the task. This can be seen with the Standard deviation for the apples collected was 1.21 where the second lowest was the agent with 2.09, the total rewards deviation was 292.5 which is under half of the Mob_fun system at 662.18 and time finished was 0.02 which was closely fol-lowed by Random with 0.04.

This was expected as the task was simple which allows for both the rule based sys-tems and the agent to work well and achieve decent results.

6.2. Task 2: No Enemies with Random Apples

The agent's results were expected to go down slightly for this task due to the apples now being randomised and would not know the best actions to make and should just try to explore the best areas which would be around the centre of the map away from the walls.

	Apples Collected		Total Reward		Time Finished		
	Mean	Standard Deviation	Mean	Standard Deviation	Mean	Standard Deviation	Completion Rate
Agent	12.70	**2.22**	518.8	942.40	24.10	**0.03**	0%
Mob_fun	**15.43**	2.30	**1930.7**	**552.16**	24.09	**0.03**	0%
Random	11.57	3.34	-2041.1	1054.94	24.09	**0.03**	0%
Lawnmower	14.67	2.87	1127.3	742.94	**24.07**	0.05	**3%**

Table 5: Task 2 Results

The agent did worse collecting an average of 4 apples less than it achieved in Task 1 but managed to consistently collect this many apples as it had the best standard deviation with 2.22. The rule based systems results varied insignificantly compared to the results of Task 1 with the exception of the Lawnmower system receiving more varied results creating a higher standard deviation. All of the systems failed to complete the task except for the Lawnmower system which managed to complete the task once giving the best average time and completion rate to this task.

The agent received a more varied set of total rewards due to finding apples along the edge of the walls which would negatively reward the agent in other runs if it searched the wall area for apples when there were none there which could explain why the standard deviation increased to 942.40 for this task.

6.3. Task 3: Enemies with Static Apples

Throughout this task, the expected result was for all of the systems results to go down due to the enemies inflicting damage and supplying a negative reward value and dying before the end of the task creating less time lived and less apples collected.

	Apples Collected		Total Reward		Time Finished			
	Mean	Standard Deviation	Mean	Standard Deviation	Mean	Standard Deviation	Completion Rate	Death Rate
Agent	7.23	**2.89**	-1572.8	**512.02**	6.90	**2.31**	0%	100%
Mob_fun	4.97	4.05	-2314.5	677.11	8.92	5.53	0%	**97%**
Random	2.70	2.97	-2368.3	532.27	5.34	2.40	0%	100%
Lawnmower	**8.87**	3.27	**-1256.6**	628.86	**9.64**	2.43	0%	100%

Table 6: Task 3 Results

As expected the results for each of the systems have lowered greatly with none of them completing the task or achieving a positive average reward. The mob_fun system was the only system that succeeded in living the full 60 seconds in the task creating a lower death rate at 97%. The agent received the most consistent results with the lowest standard deviation across the recorded data for this task while receiving the second best average apples collected and average score.

The lawnmower system achieved the best average data due to the movement of the system as the system would move up and down across the arena and would out run the enemies until it reached the wall. At the wall it would begin to turn and the enemies would catch up and deal some damage before the system begun to move away from the enemies and would be able to regain some health before the enemies would catch up again. This would usually move the system across the arena and away from the enemies faster allowing more apples to be collected until the agent reached the furthest wall where the enemies would be able to kill it because it has nowhere left to go. This system would have done worse and died faster at the mercy of stronger enemies as discussed in Section 4.2.1 while the Reinforcement learning agent and Mob_fun system would have been reasonably unchanged due to the way they attempt to avoid enemies.

6.4. Task 4: Enemies with Random Apples

The expected results for this task were similar to the results of Task 3 while also having the agent do slightly worse as the agent instead of just having to search for the apples has to search and avoid the enemies which caused the agent to not explore the middle of the map due to the enemies grouping up while trying to attack the agent.

	Apples Collected		Total Reward		Time Finished			
	Mean	Standard Deviation	Mean	Standard Deviation	Mean	Standard Deviation	Completion Rate	Death Rate
Agent	7.33	**3.00**	-1880.3	522.56	6.41	**1.99**	0%	100%
Mob_fun	6.83	3.97	-1924.3	528.75	7.89	4.83	0%	**93%**
Random	5.53	3.02	-2102.8	**502.54**	5.42	2.07	0%	100%
Lawnmower	**7.57**	3.99	**-1570.1**	721.10	**8.56**	3.36	0%	100%

Table 7: Task 4 Results

The results did not react as expected which can be seen through comparison of Table 7 and Table 6 as the results for the Agent and the Lawnmower were relatively unchanged except for a slight decrease in the average total reward. Mob_fun and Random improved with randomly placed apples by an average of 2 or 3 apples collected and a score of almost 400 and the average time spent alive among all systems was relatively unchanged.

The agent maintained the most consistent results except for the average total rewards which the Random system managed to improve enough to be the most consistent and Lawnmower maintained the best results with agent as a close second.

6.5. Results Discussion and Evaluation

Overall, the reinforcement learning agent had the most consistent results especially in the more complex environments while also getting the highest or second highest results among the systems except for the time survived as the agent would judge collecting apples as more important than taking damage which can be seen through comparison of Figure 12 and Figure 13. This could have been changed to insight better results from the agent with further amending of the reward system.

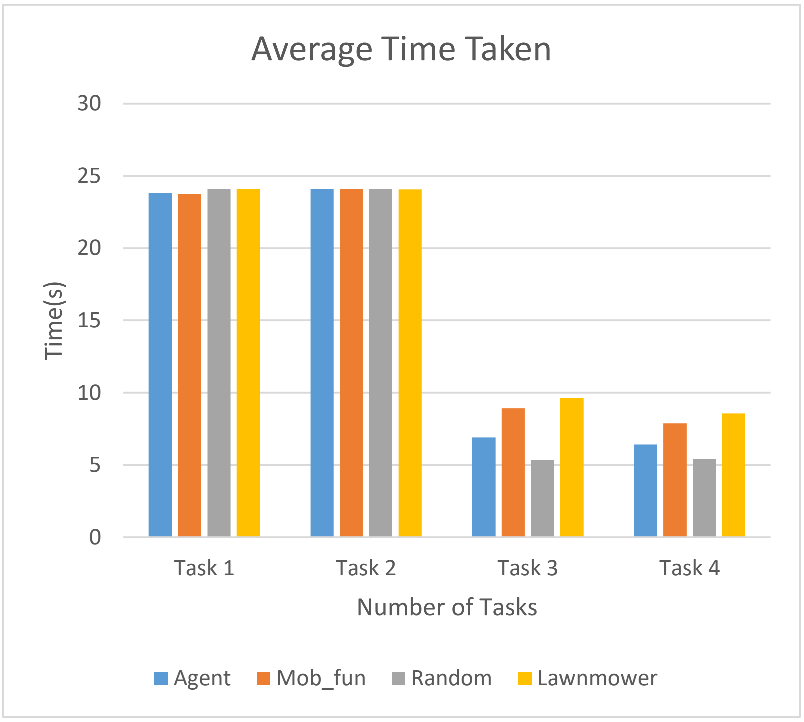

Figure 12: Average Time Taken Across All Tasks with All Systems

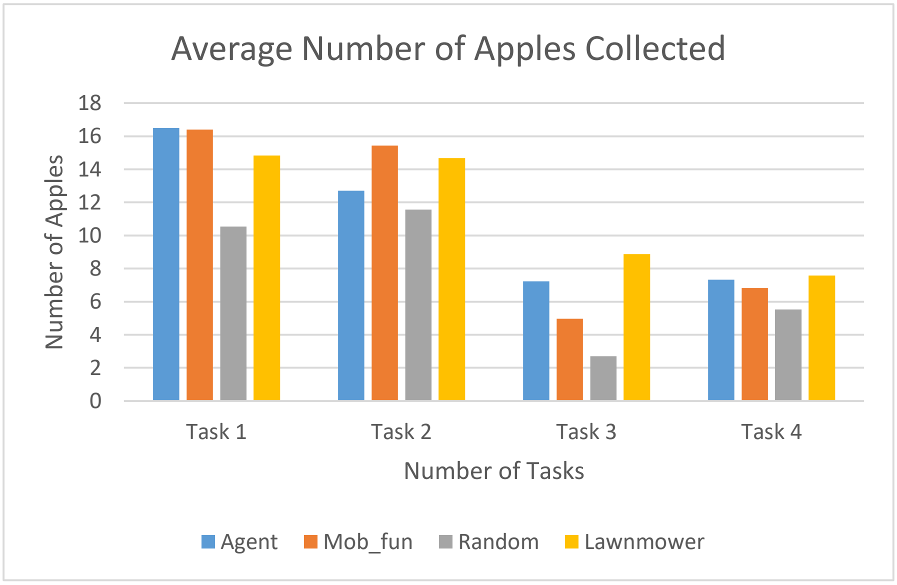

Figure 13: Average Number of Apples Collected Across All Tasks with All Systems

The Mob_fun system received the highest averages in almost all cases in both Task 1 and Task 2 but failed to do well in Task 3 and Task 4 when enemies were involved as it began to avoid the enemies by hugging the wall of the arena which would decrease the total reward received and the number of apples it would collect which can be seen through comparison in Figure 13 and 14. The system did increase the time sur-

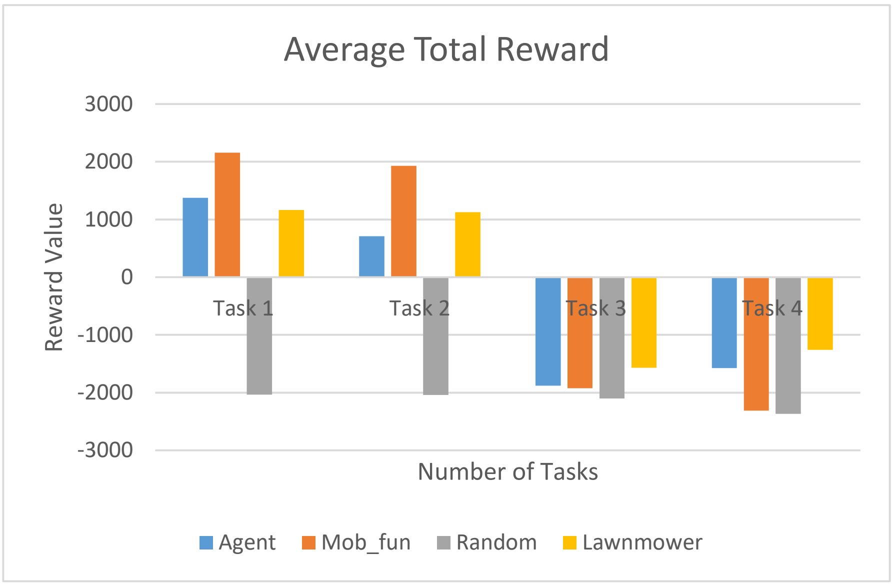

Figure 14: Average Total Reward Across All Tasks with All Systems

vived to over a third of the available task time and was one of the best systems for survival which can be clearly seen through Figure 12. Throughout Task 3 and Task 4, the Lawnmower system received the best results due to the way that it would manoeuvre itself around the arena as the path it runs would usually allow it to run away from the enemies except for brief moments where it would get hit and regain health. But due to the damage the system would take it would be frequently knocked off path and could miss multiple apples which caused a lot of inconsistency in the results for this system which can be seen in task 4 when the standard deviation was as high as 3.99 for apples collected, 721.1 for total reward and 3.36 for time taken.

Although the difference between the results could be quite small such as the average apples collected in Task 4 between the agent and the Lawnmower system was 0.24. A more in depth statistical analysis would have been run to attempt to receive more definitive results but unfortunately there was not enough time to complete more runs for this purpose.

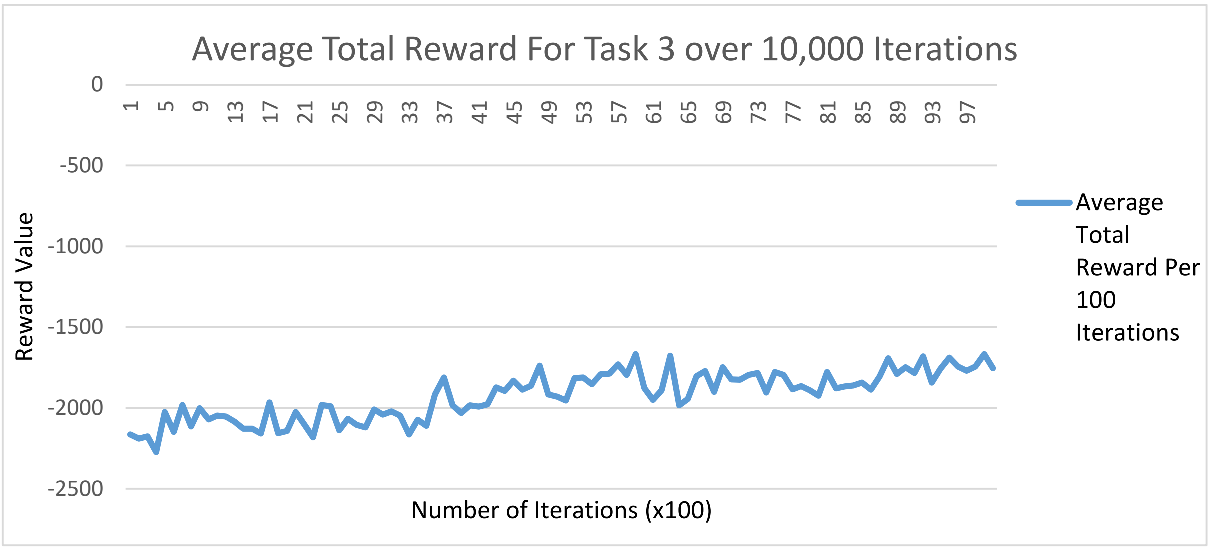

Figure 15: Results of Task 3 parameter Training

Looking back at the training graph shown in Section 4.4.1, we can see that the agent was still improving during the enemy tasks which implies that the agent still could have improved if the time was available to train it further. With this implication I believe that the agent could have improved both the average time survived and the average number of apples collected in both tasks 3 and 4 to achieve better results and have the best results over all of the remaining systems.

With the results from the multiple tasks and from looking at Figure 15 it can be seen that the reinforcement learning agent was overall the best system for use in the above tasks as it consistently received good results while also not having been trained to its full potential before attempting the later tasks. The agent also completed the tasks with no information about its environment except if an enemy was two blocks away which means with further information about its environment the agent could achieve much better results.

After collecting the results to the four tasks, you can see that using a machine learning method might not be the best way to handle simple situations like task 1 or task 2 as the agent did not do as well as the Mob_fun system and is more complex to create for such a simple task. Although, the agent could consistently collect a large amount of the apples while receiving a good average score. This could be useful in a game application as a programmer could program a rule based system to be so difficult to allow some room for the player to succeed while the machine learning method is more genuine and tries to beat the player. This means that through learning the agent can react and learn to try something different and react unpredictably in the game while the rule based system would most likely not and would only be able to do what has been dictated in its rule base.

The benefits of using a machine learning method are shown when in use in more complex environments such as when an enemy is involved in the task. This is shown as the agent received very good results in comparison to the most sophisticated rule based system Mob_fun. The agent managed to achieve the most consistent set of results as Mob_fun had a standard deviation of 4.97 for the apples collected and 5.53 for the time survived which is a big difference from the agents which were 2.89 and 2.31 which were almost half the size. This could show the benefits of using reinforcement learning for more real world applications as the results vary less allowing for more predictability while also achieving comparably decent results which could be improved with either more training or allowing more context about its environment.

6.6. Further Work

I believe that through more training the agent could have improved to be better than the rule based systems used through comparison. In Task 3 and Task 4, it became more evident that the reward system needed further changing as the agent would run into group of enemies to try and get apples due to the differences in rewards from them which means I would try and increase the negative reward from pain or decrease the positive reward from collecting apples to attempt to fix this.

But if this did not work and more time was available I might have ventured into Deep Reinforcement-Learning using neural networks which was discussed by Mnih et al (2015). This is because they had shown optimal results from utilising deep reinforcement learning in their experiment which could have proven to be very useful for this project. Unfortunately, due to the complexity of deep reinforcement learning and the time it would take this was not possible.

7. Considerations

In this stage I discuss any professional, legal, ethical or social issues that could occur after the creation of this project with any improvements to artificially intelligent agents.

7.1. Professional Issues

Through the creation of smarter and safer artificial intelligence there could be an increased use of AI in career roles such as drivers or teachers. Already self-driving cars have started being released onto the roads and have started taking over jobs from driving companies such as Uber. Although currently their use is still very limited, there is an increase in development of self-driving cars which could start taking over more jobs worldwide. This could cause various issues for people as there would be an increase in the number of unemployed people while also lowering the number of jobs available which would begin to cost the government more to fund the unemployed until they can find a job.

For education, the use of artificial agents could be more beneficial as they could lighten the amount of work a teacher must do while also being able to give a student personalised attention and assist in some issues that the students may have. This would benefit the school system as it could allow students to learn more effectively and can get the help they need faster and achieve better results from this help. The artificial agent created in this project should not create any professional issues as it is in a virtual environment but could be adapted to roles such as a self-driving car and create professional issues.

7.2. Legal Issues

Through an increased use of artificial agents there becomes a smaller list of people that can be to blame for certain accidents that occur. For example, if there were two driverless cars in an accident, there would be no driver at fault, but would have to blame the workers who have made all the cars such as the engineers or the programmers or the car making company itself. This means that there would be a legal battle to find a person at fault that would then pay for the damages to the car or the creators of the cars and any car insurance need to create new ways of dealing with these situations.

Another legal issue that could occur is an issue with keeping data safely under the Data Protection Act. This could occur if an artificially intelligent agent was assisting in a school in the education of students and knew personal details about the student such as the name, address or test results. This means the data would have to be stored securely and only accessed responsibly and only when the data is required. The artificial agent created will not be able to create any legal issues such as the ones mentioned before but if adapted into other real-life situations such as education, it could run into legal issues with storing student's data safely.

7.3. Ethical Issues

With the creation of artificial agents there can be certain safety and moral issues that can occur that a programmer might need to ensure results in a certain action. For example, with a driverless car it could come into a situation that either endangers the people in the car, or people outside of the car and it needs to make the decision of what it will do. This means the car must judge whose life is more important and make the tough moral decision of who it should endanger. There could also be a situation that occurs where the car needs to perform an illegal activity to avoid injuring a pedestrian or cyclist. This means that the car needs to morally judge the illegal activity as a worthwhile action to stop any injuries while also not taking this action if it is judged as unsafe. Throughout this project the agent will not have to make any choices quite so severe, but should learn to not die by avoiding the enemies while also travelling towards the apples to achieve the maximum score.

7.4. Social Issues

From the creation of smarter artificial agents there becomes more applications for them in the world where they can replace a human while increasing productivity. This can occur in situations such as a taxi driver because driverless cars are becoming more frequent while also being safer than a human driver and can therefore be used by the government or large companies such as Uber instead of paying for a driver. This can have a beneficial impact on society as roads can become safer from car related accidents and could allow for a faster commute between destinations as a driverless car could know more about which route is the busiest and calculate a faster route through less traffic. But the immediate impact would be negative due to the loss of jobs for the previous drivers and could result in more unemployed which would cost the government until each person can be retrained and find a different job. Another social issue would be the lack of trust in an artificial driver compared to a real driver as a person is more likely to trust another person who they can see driving compared to trusting the car. There should be no social issues with the creation of this agent in the game, but if the artificial agent was adapted for use in real world applications a social issue could arise.

8. References

Alpaydin, E. (2014) Introduction to Machine Learning, The MIT Press, Cam-bridge, Massachusetts.

Box, S. (2014) Supervised learning from human performance at the computationally hard problem of optimal traffic signal control on a network of junctions, Royal Society Open Science, 1: 140211, Accessed at http://dx.doi.org/10.1098/rsos.140211 [Accessed 27 Mar. 2016]

Eladhari, MP, Togelius, J, Smith, G, Cook, M, Thompson, T & Smith, A 2015, AI-based games: Contrabot and What Did You Do? in *Proceedings of the 10th International Conference on the Foundations of Digital Games 2015 (FDG 2015).* Santa Cruz, CA, 10th International Conference on the Foundations of Digital Games, Pacific Grove, United States, 12-15 August.

Kotsiantis, S. B. (2007). Supervised Machine Learning: A Review of Classification Techniques. University of Peloponnese, Greece. Accessed at https://datajobs.com/data-science-repo/Supervised-Learning-[SB-Kotsiantis].pdf [Accessed 16 Nov. 2016]

Levin, J. (2011). Teaching with Minecraft. [online] Minecraft Edu. Technical Report. Available at: http://services.minecraftedu.com/wiki/Teaching_with_MinecraftEdu [Accessed 11 Nov. 2016]

Liao, Y., Yi, K., & Yang, Z. (2012). Reinforcement Learning to Play Mario. Stanford University. Accessed at: http://cs229.stanford.edu/proj2012/LiaoYiYang-RLtoPlayMario.pdf [Accessed 11 Nov. 2016]

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S., Hassabis, D. (2015). Human-Level control through deep reinforcement learning. Macmillan Publishers. Available at: https://storage.googleapis.com/deepmind-media/dqn/DQNNaturePaper.pdf [Accessed 16 Nov. 2016]

Moriarty, C. & Gonzalez, A. J. (2009). Learning Human Behaviour from Observation for Gaming Applications. Proceedings of the Twenty-Second Interational Florida Artificial Intelligence Research Society Conference, May19-21, 2009, Sanibel Island, Florida, USA.

Ram, A., Ontanon, S., & Mehta, M. (2007). Artificial Intelligence for Adaptive Computer Games. In Proceedings of the Twentieth International FLAIRS Conference on Artificial Intelligence (FLAIRS-2007), AAAI Press.

Rhalibi, A. E., Wong, K. W., & Price, M. (2009). Artificial Intelligence for Computer Games. International Journal of Computer Games Technology, [online] Volume 2009, Article ID 251652, 3 pages. Available at: https://www.hindawi.com/journals/ijcgt/2009/251652/ [Accessed 11 Nov. 2016].

Sutton, R. S., Barto, A. G. (1998) Reinforcement Learning: An Introduction, The MIT Press, Cam-bridge, Massachusetts.

Treanor, M, Zook, A, Eladhari, MP, Togelius, J, Smith, G, Cook, M, Thompson, T, Magerko, B, Levine, J & Smith, A 2015, AI-based game design patterns. in *Proceedings of the 10th International Conference on the Foundations of*

Digital Games 2015 (FDG 2015). Santa Cruz, CA, 10th International Conference on the Foundations of Digital Games, Pacific Grove, United States, 22-25 June.