UNIVERSITY of ABERTAY DUNDEE

<u>USES OF REINFORCEMENT LEARNING WITHIN A HELPER AGENT</u>

James David Hogg

University of Abertay Dundee
Institute of Arts, Media and Computer Games
BSc Computer Games Technology (Hons)
May 2012

Abstract

Video game A.I. has been geared towards generating suitable behaviours for in-game enemies first and foremost, but any research into the limits of player-friendly A.I. agents is neither obvious nor substantial. This document details the execution of a project that investigates the position such an agent can take. The objective is to enhance a player's immersive experience within a game by providing direct, efficient advice on how to play better. This objective is sought with the implementation of Reinforcement Learning algorithms. They shall be used to monitor player behaviour and analyse player satisfaction with any advice that the application outputs. The application will be an extension of the open-source Platform Game *Secret Maryo Chronicles* and the *Reinforcement Learning Toolbox* by Gerhard Neumann.

The two chosen techniques selected for study are Temporal Difference Learning methods and Q-Learning methods. The research also examines the efficiency and effectiveness of these chosen algorithms by comparing the results of key variables that indicate their success, such as reward values and CPU ticks. The algorithms are put through multiple experimental trials and average quantities in the form of these variables are taken. A questionnaire is also used to assess the qualities of volunteering participants such as enjoyment and skill improvement: these are important for evaluating the "bot" and measuring to what extent it is achieving its aims.

Results demonstrated trends, but not trends that were expected. Temporal Difference Learning outperformed Q-Learning, but the most "sophisticated" system that they were tested in turned out to be the most efficient and resourceful. The questionnaire data ended up being too insufficient to draw many meaningful conclusions from. Further findings and criticisms are discussed, and fairly stable conclusions in support of the use of reinforcement learning in games are given, but debate is raised over this particular application's suitability with Reinforcement Learning. Recommendations for future video game programmers are also given.

TABLE OF CONTENTS

LIST OF FIGURES

Introduction

Project Concept

If video games are to sell and stay above competition, they are required to achieve and maintain a high standard of entertainment that is as dynamic as possible. Most noticeably, this dynamic quality lies with the game's careful sophistication in Artificial Intelligence. Without an intuitive A.I. system, the game will fail to stand out from the crowd and be impressive in the eyes of the consumer market.

What seems to draw most of the attention in video game A.I. development are the behaviours of "enemies" in a game, or similar tangible objects that obstruct the aims of the player. This choice of focus would be the reasonable approach, considering that the vast majority of "characters" in a video game are rarely on the side of the player. And if they are, their tasks are not extremely complicated. In *Sonic The Hedgehog 2* for example (Figure 1), the character Tails was based off an extremely simple script that mostly copied input actions from the player. The energy spent on A.I. programming is understandably directed at making challenging and adaptable A.I. that will, as far as possible, give the game replay value for the consumer.

Figure 1: ***Sonic The Hedgehog 2***

This project seeks to fill a gap in current A.I. research today and offer a fair insight into the possibilities that video game programming could have in regards to *helping* a player as opposed to posing a threat. There does not seem to be any significant amount of immediate study on this topic, and some academics are calling for more games that consist of decent A.I. systems that help players (Schwab, 2008).

Fundamentally speaking, the best way to do this would be to investigate if a system can accurately guide a player to his or her desired outcome. It is this premise that will form the basis of the project's research. Every player is going to want to experience his or her own unique, personal entertainment from a game. Because of this, the most typical wants and needs of the target players must be established and prepared for by the application. And if the player can be "guided" to this specified experience based on the key behaviours exhibited by that player, it will hopefully form the foundations of future games that may want better performing allies, as one possible example.

The type of application that will attempt to explore the potential uses here will be that of a "helper bot", which will be responsible for analysing the key behaviours, whether the player may be aware of them or not, and offering visual advice in text format on how to better go for the desired experience (explicitly chosen by the player from the list of common wants and needs), which shall hopefully encourage the player to change his or her behaviours. Also, in order to make the advice itself as accurate and appropriate as possible, a related A.I. system will be set up in order to rate the advice given depending on the player's satisfaction with the advice (therefore, this will require the player to give feedback to the application depending on his or her approval). If the player can acknowledge the advice in this way, a good case can be made for taking accurate, quantitative measurements of how "believable" the A.I. is.

If this interactive form of A.I. were to be successful in its objectives, it may invite researchers to investigate if the interaction could have beneficial improvements for player immersion within a game.

Chosen Field of Artificial Intelligence

The range of techniques chosen to construct this Artificial Intelligence system will be that of Reinforcement Learning. The core concepts of reward and punishment drive this style of A.I., which ties in nicely with the need to record behavioural aspects and rate any advice given in a way that will determine if it will reappear in the future. It would also be best to take advantage of what several academics are calling the currently most popular area of A.I. research of recent years (Björnsson,Yngvi 2004).

It is the need for intrinsic systems that leads to a consensus within the gaming industry demanding "appropriate" A.I. for in-game characters, as opposed to "clever" A.I. This distinction is important because it is often the case that most programmers would enjoy aiming for systems that are both genius and efficient. It is also an inevitable distinction, due to the impracticality of incorporating academic levels of A.I. with games of limited resources - development teams will sooner or later realise that they have to constrain their efforts and offer illusions of high intelligence as opposed to the "real" thing. In *Reinforcement Learning: An Introduction* by Richard Sutton and Andrew Barto (2005, p14), they offer a principle that stresses the importance of putting the final result first and foremost: "Rather than directly theorising about how people or animals learn, we explore idealised learning situations and evaluate the effectiveness of various learning methods."

Bearing in mind these issues, we can construct a research question that will best direct how to achieve the objectives.

Research Question

"What are the most practical methods for generating a believable reinforcement-learning A.I. agent that aids players' intentions and improves its responses depending upon player feedback, and what consequences will the algorithms have on performance?"

This research question can be approached by breaking it down into further, specific questions and methods:

Within the field of reinforcement learning, what methods of determining players' consistent behaviours are most accurate?

- Establish the most appropriate behaviours and goals to monitor.
- Investigate algorithms for comparison of data.
- Use the test application to observe the most sensible results.

How can the A.I. agent gather suitable player feedback and use it to adjust its analysis of players' behaviours to behave more suitably?

- Research methods for obtaining meaningful feedback that will successfully let the player approve or disapprove of given aid.
- Treat feedback as variables that contribute to the decisions made from the behavioural database (the "rewarding" variable?).
- Take care when adjusting mature algorithms with new feedback data.

What are the consequences of the reinforcement algorithms on overall computational resources?

- Adhere to a strict set of established benchmarks within the test application.
- Compare and contrast required algorithms against these benchmarks.
- Draw reasonable conclusions from the results.

On what criteria can this A.I. agent be deemed "believable" when it responds to its analysis of behavioural data?

- Interview a suitable range of test subjects and obtain critique.
- Measure the subjects' "ratings" they give for the agent's believability after entering feedback into the test application, and then note its variations.
- Determine to what extent the agent is correctly serving the purpose of helping the player achieve his or her goals.

Literature Review

Foundational Literature and Resources

First, it must be established what is meant by Reinforcement Learning in the context of this application. The process involves iterative calculations in order to find an optimal solution to a defined problem, based on a range of available solutions (Sutton, Barto, 2005). These iterations are influenced according to the adjusting of variables involved with dictating its rate of learning and extent of reward or punishment. Normally, this is simply signified as the *reward* component of the function (Fürnkranz, 2001). There are other components that can work in conjunction with this reward in order to optimise the process, which are described later. The concept is similar to that of the vast majority of A.I. algorithms that work towards an optimal solution using a fitness function (such as Schwab, 2008 and Millington, Funge 2009).

The main choice of literature that will be used to fundamentally guide the project in the right direction will be the book *Reinforcement Learning: An Introduction* by Richard S. Sutton and Andrew G. Barto (2005). The text covers all of the essential principles of the subject, and will be the main reference text when comparing techniques and additional sources. The authors have been actively researching the subject since 1979 and have credible references for the information they have stated. In it, the authors outline three elementary solution methods for solving Reinforcement Learning problems: Dynamic Programming, Monte Carlo Methods and Temporal-Difference Learning. They state:

"Each class of methods has its strengths and weaknesses. Dynamic programming methods are well developed mathematically, but require a complete and accurate model of the environment. Monte Carlo methods don't require a model and are conceptually simple, but are not suited for step-by-step incremental computation. Finally, temporal-difference methods require no model and are fully incremental, but are more complex to analyze. The methods also differ in several ways with respect to their efficiency and speed of convergence." (p 109)

In one of the most recognised and comprehensive Reinforcement Learning toolboxes available (Neumann, 2006), the same toolbox that shall form the framework of this project (a direct link can be found in the References), Monte Carlo methods and Dynamic Programming methods are used as policy evaluators, while Temporal Difference Learning is one policy of Reinforcement Learning itself. Therefore, one other policy will be required for comparative testing amongst the two evaluators. Bearing in mind that A.I. research and conclusions are significantly context sensitive as described previously, perspectives such as the above quotation must always be subject to further criticism. The "context" here would refer to the gaming environment. Therefore, the project will give a fair insight into the elements' capabilities by testing the most recognised and distinguished algorithms on offer from the toolbox. This will contribute to the validity of the theory itself and aid greatly with the project's own purposes.

Reinforcement Learning Requirements

The model that will be used in this application will consist of two separate Reinforcement Learning systems: one will be responsible for analysing the player's in-game behavioural trends as they are by monitoring repetition of these behaviours (called RL1), and the other will be responsible for the rating of specific sets of advice given to the player based on the player's satisfaction with the advice given (called RL2). This also provides a good opportunity to do side-by-side comparisons of algorithms.

The initial intention of the project was to use a Reinforcement Learning algorithm in order to deduce what the player wanted by monitoring behaviours. However, the aims of the project make more sense when the player is asked to explicitly state what he or she wants from the game (from a list of choices), and *then* monitor the behaviours to see if the player is actually achieving what is wanted. If the player is off course from what is wanted, the purpose of the advice is to encourage the player to get back on track. It would also be reasonable to take into account the need for several testing participants in order to optimise this system, so that there is not a large degree of bias involved with the results.

The algorithms usually group specific actions into categories known as "states", and the purpose of Reinforcement Learning is to assign a rating to whichever state is being favoured, or "rewarded". Referring to Sutton and Barto again (2005, p24), they illustrate a diagram that shows how states and actions would be incorporated in a game of tic-tac-toe (Figure 2).

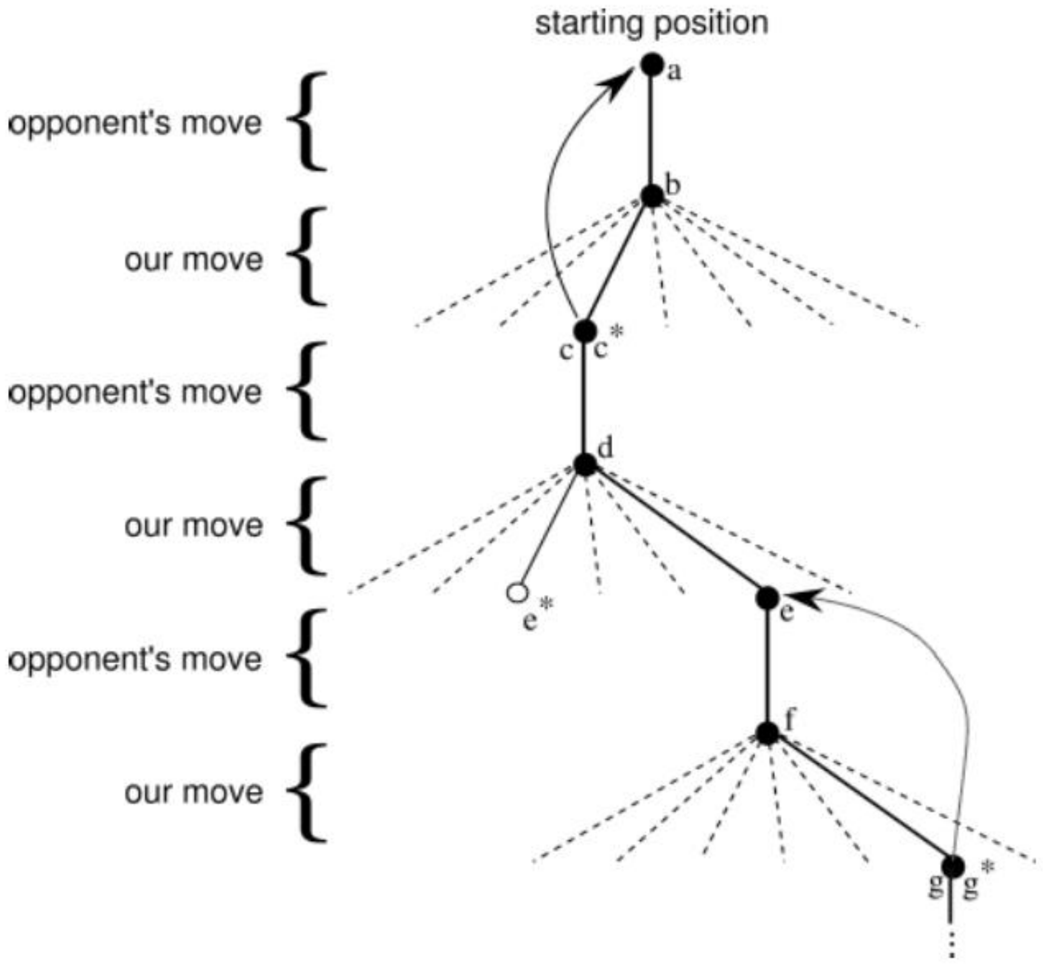


Figure 2: Tic-Tac-Toe Environment Example

The state would represent the board as it is, and an action would be one of several possibilities that could be taken by a player as a result of the board’s state. This process can be represented as a hierarchical tree as shown.

It is reasonable to be sceptical of this approach due to the potential of many combinations of states and actions slowing down performance and making real-time output difficult. But there have been studies on small-scale projects (Madeira, Corruble and Ramalho, 2006) that have demonstrated it can be done to a respectable degree. The research for this project does not need to be carried out on a large scale. Therefore it is safe to assume that there will not be any serious issues in regards to limited resources.

The ratings themselves converge to a limit, and normally this limit is between 0 and 1 (signifying the worst and best ratings of a solution, respectively). However, this is only done if the problem’s solution is continually being rewarded and the problem’s definition itself does not change. If the solution’s value ends up converging to a limit, or can be established as being “close enough” to that limit, this is the point where the problem is considered “solved” with this solution (Millington and Funge, 2009). This range of favourability would prove to be an important indicator of how well the “helper bot” is doing in regards to finding the right advice for the player (the RL2 system). If the player does not like the advice, or does not think the advice is helpful, the negativity will show because the current solution’s value will decline by a considerable rate. This would give justification for the specific unhelpful advice to be removed from a state set, and then the iteration process can continue until an optimal collection of advisories are selected. The aim is to develop an optimal collection for each of the typical wants and needs that will already be prepared, based on ratings from participants willing to provide data. The need for several testers is essential due to the fact that it requires numerous inputs in order to get a reasonable convergence. For RL1, in-game habitual behaviours are bound to be repeated enough in order to get the required degree of convergence.

Chosen Algorithms

An algorithm that can achieve this would have to consist of a mathematical recurrence relation that remembers the value noted from the previous calculation. This logging of experience is also an important contributing factor in determining the success of the system. The Q-learning rule is one popular algorithm for Reinforcement Learning, and is a basic example of rating a solution's success (Millington, Funge, 2009). The algorithm will act as the alternative policy to Temporal Difference Learning:

$$Q(s , a) = (1 - \alpha).Q(s , a) + \alpha.(r + \gamma.\max(Q(s' , a')))$$

This function depends on a state-action combination's previous value "Q(s , a)" and returns it a new fitness value. The parameters passed into this function in order to vary its result would be directly linked to the players' approval of advice (RL2) and frequency of behavioural occurrences (RL1). For the recording of behaviours, it would be based on the frequency of each action. It works with a rate of learning "α" that varies the linear blend, a discount rate "γ" that controls how much the old "Q" state-action value depends on the new "Q" state value, a reinforcement rate "*r*" that depends on the specified rewarding feedback, and a return of the maximum value action from the increasingly favourable state set "$\max(Q(s' , a'))$". All of these components gradually create a convergence towards the fittest state-action combination. In this simple case, "*r*" would serve as the parameter that the programmer and player should vary for approving behavioural-states and advice-states, respectively.

Temporal Difference Learning algorithms are also "fully incremental" functions as previously described by Sutton and Barto, which implies a similar recurrence relationship to Q-Learning. Here, prediction methods are used to estimate current values ("vectors", which in this case may be considered state-action pairs as described previously) based on future values that will come later in the iteration process (Barto *et al.*, 2007). This approach relies on the existence of a trend between previous values: if no trend exists, due to inconsistent actions or altered states, the algorithm cannot achieve any better than guessing. Thus, there must be strict consistency throughout the problem solving.

$$Y_t = y_{t+1} + \gamma y_{t+2} + \gamma^2 y_{t+3} + \cdots = \sum_{i=1}^{\infty} \gamma^{i-1} y_{t+i},$$

If "x" and "y" can be taken as a respective state and action, this function both predicts the next value of "y" and gathers enough data so that a prediction on "x" can be made using $p_t = P(x_t)$.   The aim is to resemble the accuracy of the "x" prediction with the "y" prediction as much as possible. Any proportion in error between these two values can be used as a function for tweaking the prediction's parameters, and this is how the Reinforcement Learning algorithms will attempt gradual tendencies towards fit and stable state-action pair values. By altering the "γ" parameter for instance, once again named a "discount rate", the influence of much later "y" values can be reduced in order to compensate for the gradual inaccuracy that must ensue when deducing later values. Striking the required balance between accuracy and efficiency is key.

Additional Issues

Reinforcement Learning is strictly classed more in the area of *supervised* machine learning as opposed to unsupervised learning. It was previously defined as being unsupervised, due to the fact that the algorithm's answers are not strictly forced to arrive at a given solution. This was justified with Fang and Ting's (2009) statement that "it is one of the unsupervised machine learning methods in the area of artificial intelligence. The methodology is developed based on the concept of 'trial-and-error', and the result of each trial-and-error will be saved as a 'delay reward'. " (p1031). However, it is indisputable that Reinforcement Learning is guided to a solution by the supervised variations in reward variables. The "trial and error" component mentioned by Fang and Ting does not compare in regards to the more essential reward influence. It is better to define the situation in terms of how much control is given to the inputs of the algorithm (Valpola, 2000), and Reinforcement Learning's reward variables are supervised in this case. This is important to understand because it needs to be stressed that parameter values and structures are under control by a “supervisor” in each of the Reinforcement Learning models. And this important applies for the purpose of collecting results and answering the research question.

Because of the high rate at which Reinforcement Learning algorithms consume application resources (Funge, 2004), it is better to try and keep all calculations confined to offline periods. In other words, making the video game’s vital run-time components such as graphics and sound a priority. If the calculations can be made before any execution of the program on screen then that would be beneficial (Millington and Funge, 2009). But if not, the use of parallel threads may be an option requiring consideration. By keeping these compromises, and maintaining a reasonably small-scale project that does not manage unnecessarily large databases for the A.I.’s decision-making, practical and accurate testing can be carried out.

Methodology

The research for this project can be best carried out by directing specific methods towards the four sub-sections of the research question.

Research Question Part One

"Within the field of reinforcement learning, what methods of determining players' consistent behaviours are most accurate?"

First of all, it must be stated that the free and open-source video game *Secret Maryo Chronicles* will provide as a basis for gathering basic in-game behavioural actions. The game is a fair, efficient two-dimensional platform game that is programmed in C++ and uses OpenGL graphics. It is very similar to that of the *Super Mario Brothers* franchise: the aim is to get to the opposite end of the world environment while gathering a high score by collecting coins, defeating simple NPCs, "powering up" the main character, and taking minimal time amongst others, as can be seen in Figure 3. All of these attributes and more are made easily available through the freely editable source code, and shall provide sufficient data for the purposes of determining whether or not a player is achieving his or her goals (RL1).

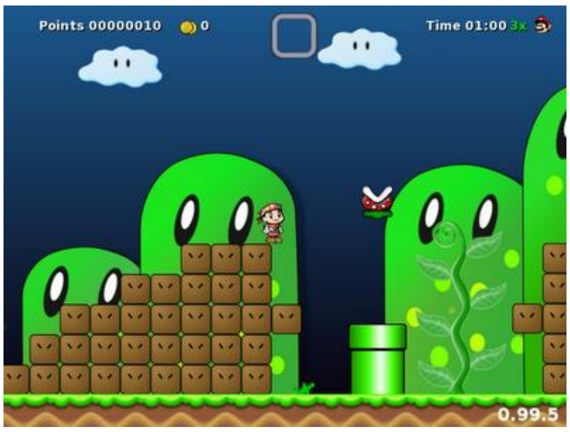


Figure 3: ***Secret Maryo Chronicles***

In order to achieve this with Reinforcement Learning, it must be defined what the player wishes to stay away from and what the player is aiming for: these "states" would be rated as 0 and 1 respectively. This way, the player's current progress towards the desired goal can be rated based on how "close" it is according to these markers (Sutton and Barto 2005). If, for example, the player wished primarily to get the highest score possible from a certain level, this would be the chosen "state" goal, and the actions associated with this state would be each of the required behaviours in question. So for a high score, higher numbers of coins collected or enemies defeated would be the behavioural actions that push the ranking value further towards 1, while lower numbers would push the value towards 0. The algorithms calculate the correct value for all states (which are formed by each possible combination of actions) by first of all giving each an initial random value between 0 and 1, and using Mathematical recurrence relations to converge to the final values. It is then a case of examining which state is currently being exhibited by the player in order to rate his or her progress towards the desired goal.

Since it can be quite rightly argued that an exponential number of states will be generated if all possible behavioural combinations are measured, steps must be taken in order to cut down the numbers. In the case of behaviours measured by quantity, more abstract definitions shall be given (e.g., "no coins", "few coins", "many coins"). Since the question asks for the greatest accuracy of behavioural analysis, it can be answered by how far this abstraction can be minimised depending on the behavioural definitions themselves, the quantity of behaviours and the choice of algorithms.

Research Question Part Two

"How can the A.I. agent gather suitable player feedback and use it to adjust its analysis of players' behaviours to behave more suitably?"

Once RL1 has carried out an analysis of the recorded behaviours, the system can now begin to output the advice scripts using RL2. The advice itself will be a database of simple text outputs consisting of no more than three sentences each. For each available goal, five advice units will be allocated, and will be outputted one after another in order to obtain a rating for each. In order to cut down on computational time the ratings are abstracted very simply: “Good”, “OK” and “Bad” are the values chosen. Originally, these values would range between 1 and 10 (where 1 was bad and 10 was good) but it was decided that this would bear too much strain on the iterative algorithms.

The same idea as RL1 is used for this part of the application. As before, the “goalposts” used here (1 and 0) would be the complete approval of the advice system (all “Good” ratings) and the complete disapproval of it (all “Bad” ratings). A rating of the agent’s overall performance can then be given by how close it is towards the perfect approval.

Upon reflection, this sounds somewhat more than what is necessary because it can be argued that a simple mean calculation will suffice. This is a legitimate criticism, and later on it shall be discussed how the concept of Reinforcement Learning does not practically fit with the initial project idea. However, the structure is still robust enough to be included with the algorithms and yet give reasonable, experimental results.

Research Question Part Three

“What are the consequences of the reinforcement algorithms on overall computational resources?”

Each of the algorithms undergoes several “episodes”, which are recurrence iterations that slowly tweak the states’ values to their converging results (Pfeiffer, 2006). An accurate indicator of the efficiency of the algorithms in question would therefore be the number of these episodes: if it can be shown that there is a general trend for one algorithm to have fewer episodes than another, then it can be established that particular algorithm has superior efficiency as far as this variable is concerned. This is because of the evident need to use less computation power. Also, a comparison between the two Reinforcement Learning systems will need to be done in this way.

However, it can be correctly argued that an indication of episode cycles says nothing about the time it takes for each episode to complete their respective processes. Therefore, CPU "ticks" will also be measured for the duration of the algorithms' executions. This will help to support any conclusion drawn from examining the number of episodes. "Tick" measurement is the recording of milliseconds that a CPU has been in operation for, and is a recognised benchmark regularly used for program analysis (Dai, 2009).

Because Reinforcement Learning algorithms are offline algorithms, there is no need to obtain data on variables such as frame rate or memory consumption. The application is expected to execute calculations without any interference from graphical or game-play functions, so a frame rate measurement would not be necessary or meaningful. And since it can be established that there is plentiful memory resources with today's standard computers, and that the application does not have to worry about real-time interference from the open-source game it is based on, memory would not appear to be a meaningful variable when it comes to critical performance analysis either.

Also, another important component that will be compared amongst each of the Reinforcement Learning techniques and systems will be the average reward given throughout the iterative process. Generally speaking, if the reward is high, the system is learning faster due to the greater tendency to converge towards an optimal value (Sutton, Barto, 2005). If an algorithm is said to have a higher average reward than another, it can be established that the algorithm is the faster learner.

Research Question Part Four

"On what criteria can this A.I. agent be deemed "believable" when it responds to its analysis of behavioural data?"

The best way to test for “believability” is to ask questions of participants willing to test the system. The questions themselves must assess important quantities that will justify what can be called “believability” within the application. They must also not delve too deeply into any unethical territory.

Most sources agree that in order to get unambiguous, quantifiable answers, the questions must be “closed” (Economics Network, 2011). Taking this into account, we can gather accurate data in a quantifiable, measureable form that will offer a sincere analysis of the participants’ answers. In comparison to “open” questions, where the participants can write whatever they wish, “closed” questions can guarantee certain answers and hence be organised systematically.

In this case, participants were asked to note down whether or not the application was helpful with achieving the goal that he or she selected from it, and also how much the application contributed to the overall “enjoyment” of the game as a whole. These quantities can be reasonable factors in evaluating how well the application is seen to be “believable” when it comes to its tasks.

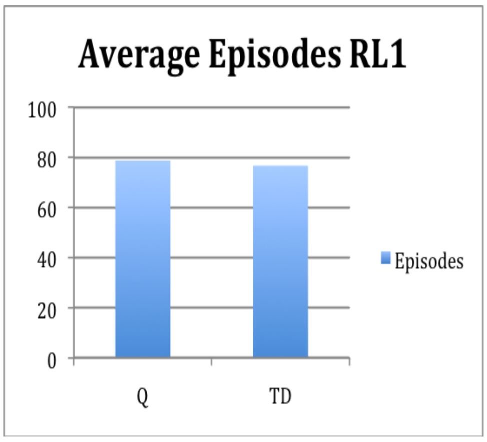
Also, questions are asked about the changes that participants think are occurring in relation to their “skill” at playing video games in general. If the “helper bot” is succeeding in increasing the skill of the participants, it will show in the answers that are given. These questions consist of asking each participant to rate their skills in general gaming genres and the platform game genre specifically, and this is done both before and after the testing of the application itself. If there is any difference between the ratings, this will serve as the indicator of the application’s success. Of course, this requires the accurate discretion from the participants themselves, which may not be accurate or objective. However, considering that there does not seem to be a better way to achieve feasible ratings of the skills themselves other than thorough analysis of the participants’ behaviours after the interview, this is the best opportunity available.

Slightly varied questions are asked in order to give greater chance of reducing this bias. The participants are also asked to state after the testing itself whether or not skill has improved using a scale from “much worse” to “much better”. This subtle difference may give way for different conclusions to be formed.

Results

The following averages were taken from three trial executions each consisting of five iterations for the two Reinforcement Learning models and the two techniques that were successfully programmed (Q-Learning and Temporal Difference Learning).

Average Episodes



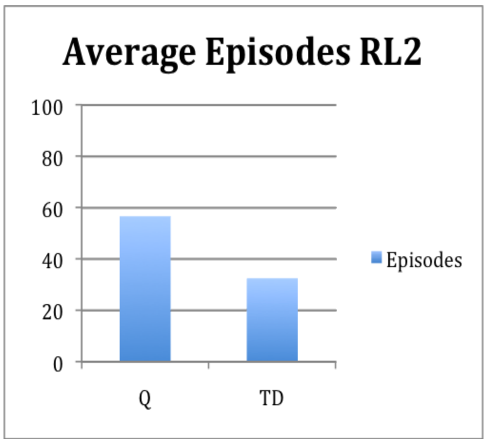


Figure 4: Average Episodes RL1 **Figure 5: Average Episodes RL2**

The indication here is that RL2 takes fewer episodes to learn the problem. It must be remembered that RL2 has more state combinations to learn from (three to the power of five, as a result of giving five advice units one of three ratings each) than that of RL1 (which is only three to the power of four, since four behaviours are being monitored per goal with one of three ratings each). Temporal Difference Learning also appears to use fewer episodes than Q-Learning.

Average Rewards

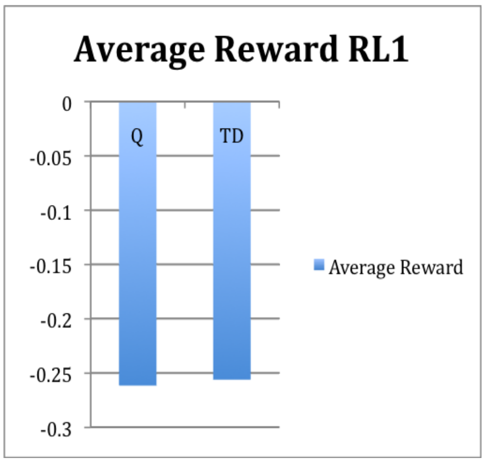


Figure 6: Average Reward RL1

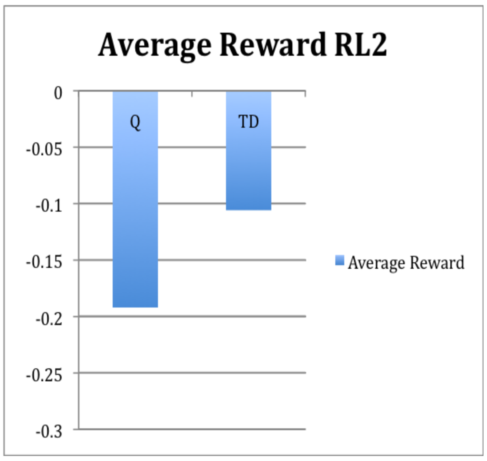


Figure 7: Average Reward RL2

From these results, RL2 appears to have higher values than RL1. And the Temporal Difference Learning algorithms seem to have higher values than Q-Learning algorithms. From what can be gathered so far, there seems to be a general trend indicating that RL2 is the more superior system, despite having more states to contend with. And the differences between Temporal Difference and Q-Learning seem to be clearer when that comparison is made with RL2.

<u>Average Ticks</u>

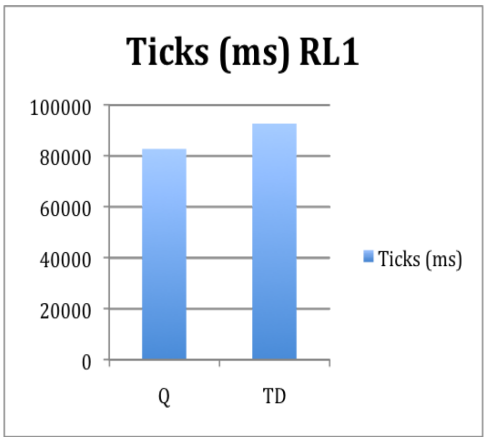


Figure 8: Average Ticks RL1

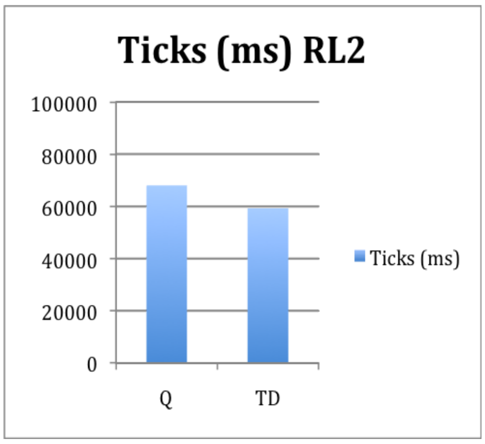


Figure 9: Average Ticks RL2

As now expected, RL2 takes up less computational time to learn problems than RL1. This points towards the same general trend and will form the basis of a very stable conclusion that can now be drawn. But in regards towards the difference between Temporal Difference and Q-Learning, a variation in results has presented itself with RL1: TD seems to take more time to compute than Q. However, this is too subtle an anomaly, and not really worthy enough to be considered seriously.

The following results were taken from the questionnaires that were issued to six participants who offered to test the application. This sample size is small, so it will be hard to draw a significant conclusion. But it may give way to some insight nonetheless.

“Helpful” Results

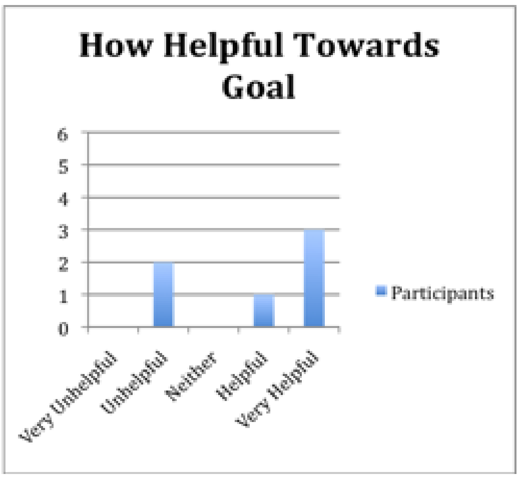


Figure 10: Goal Help Values

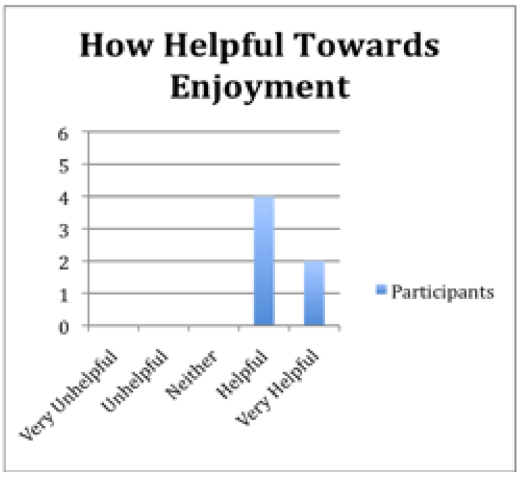
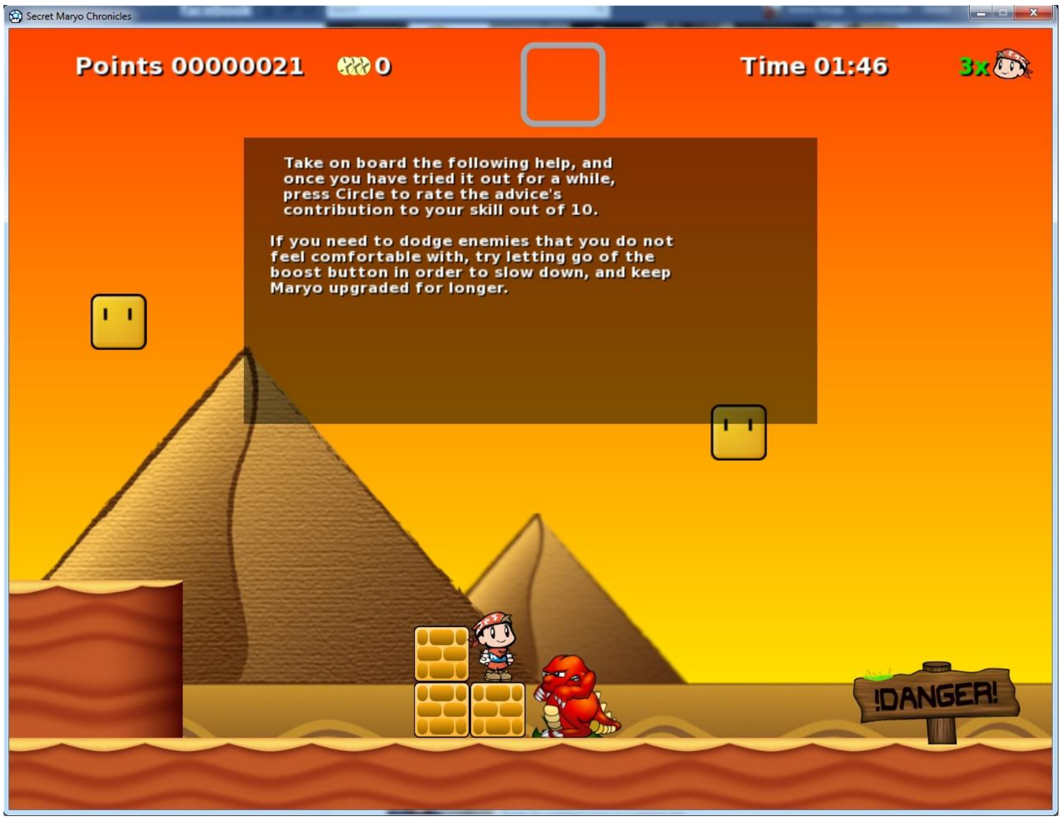


Figure 11: Environment Help Values

The other results as well as the questionnaire itself can be found in the appendices. From what can be gathered, these results seem to demonstrate a clear approval of the games capacity to boost the enjoyment experience for the participants. However, data concerning the application's helping of the participant in reaching the specified goal is not as strong.

Screenshots

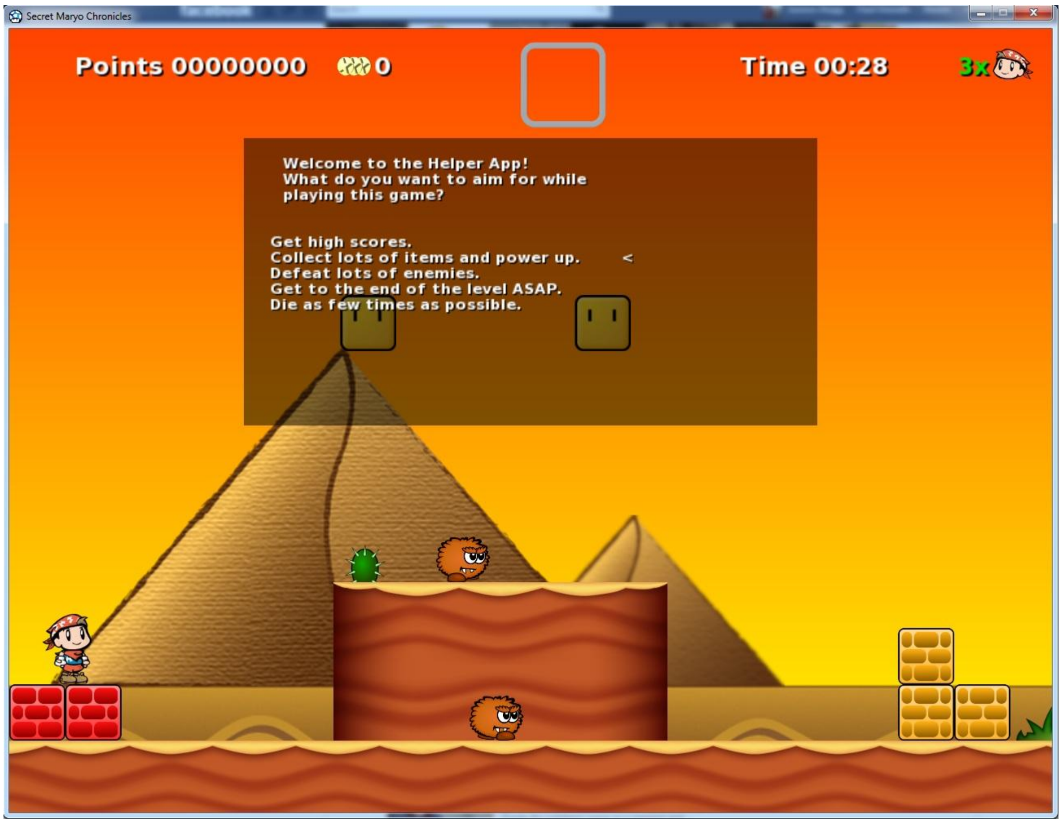
Finally, the following example screenshots show the application running and outputting the correct instructions to the participants.



Take on board the following help, and once you have tried it out for a while, press Circle to rate the advice's contribution to your skill out of 10.

If you need to dodge enemies that you do not feel comfortable with, try letting go of the boost button in order to slow down, and keep Maryo upgraded for longer.

Figure 12: "Helper Bot" Screenshot One



Welcome to the Helper App!
What do you want to aim for while
playing this game?

Get high scores.
Collect lots of items and power up. <
Defeat lots of enemies.
Get to the end of the level ASAP.
Die as few times as possible.

Figure 13: "Helper Bot" Screenshot Two

The "helper bot's" main functionality works with virtually no bugs or errors. The prototype can be easily recompiled beforehand in order to get the application's resulting outputs for both the Temporal Difference Learning method and the Q-Learning method. Both versions are compiled already on the submitted CD.

Discussion

Analysis

The results used to evaluate the Reinforcement Learning algorithms were very fairly obtained due to their offline nature. There was no live-game code such as graphics and sound to contend with: this means that there was no varying interference from the CPU that would have needed to deal with these issues if they had been taken into account. On this basis, the findings can be trusted. Any replication of experimental testing with the project ought to yield similar findings. And this observation means that the need for multiple threaded executions was unnecessary (using multiple threads for the algorithms regardless would have self-evidently improved efficiency (Eggers *et al.*, 1997)).

There appears to be a contradiction between what the results were expected to show and what they did show. For each goal, RL1 offered three values for each of the four monitored behaviours, the values simply being 0, 1 and 2 where 0 is “occurring infrequently”, 1 is “occurring occasionally” and 2 is “occurring frequently”. A state was defined as a combination of the four behaviours’ values (while an action was defined as altering one of these frequency values). Therefore, there were three to the power of four possible state combinations (81). And with RL2, there were five advice units with again three values (the ratings “Good”, “Okay” and “Bad”), giving three to the power of five possible state combinations (243).

The contradiction is that because there are more state combinations with RL2, that ought to have presented itself as the lesser inefficient system since there are more state probability values to learn, but it does not. However, there seems to be a reasonable explanation for this. As noted in the tutorials presented on the website accompanying the Reinforcement Learning Toolbox, such as the *Cart-Pole Problem* and the *Shortest Path Problem* both written by Gerhard Neumann in 2006, the process calls a reset function to start a new episode if the actions lead the system into a “failed state”. Taking the *Cart-Pole Problem*, the code specifies this important step in the learning process (p5):

```
// determine wether the episode has failed
if (x < -2.4 || x > 2.4  || theta < -twelve_degrees ||
   theta > twelve_degrees)
{
         reset = true;
         failed = true;
}
// indicate that a new episode has begun
if (reset)
{
         printf("Failed State: x = %f; theta = %f\n", x,
            theta);
}
```

The system is based off a classic A.I. problem involving balancing a pole on a moving cart. In this example, a “failed state” constitutes a situation where the pole cannot possibly be balanced any longer. In the systems RL1 and RL2, a “failed state” is defined as a state combination that would resemble either the player enacting all the incorrect behaviours that would favour the achievement of a set goal or a “Bad” rating for every advice unit (a perfect “0” score). This implementation allowed for the systems to be tested in situations where a reset would be required in the event of an episode “failing”. This is important for giving each state combination a value that signifies how probable it is that the next action will make the new state a failed one, hence obtaining a meaningful value for rating the player’s performance towards the goal or the validity of the advice depending on the player’s feedback.

The contradiction seems to have presented itself because even although one of the 81 state combinations of RL1 is a failed state, there is also only one failed state within the 243 state combinations of RL2. Therefore, RL2 has a lower *probability* of reaching the failed state and does not need to reset itself as often as RL1 despite having more states to contend with learning, which explains the lower number of average episodes. It also explains the fewer average ticks: there are fewer calls to the reset function in the C++ code and hence less cycles for the CPU to execute.

RL2 also appears to have a higher convergence because the average rewards are higher than RL1’s. This suggests that there is less interference with the learning process as a result of the lower probability of stumbling upon a failed state.

With the exception of RL1's ticks, Temporal Difference Learning appears to outperform Q-Learning every time. Looking at what was discussed before, Q-Learning makes an accurate but resource-consuming calculation for each state-action value. It takes previous values and calculates new ones on a consecutive basis. With Temporal Difference Learning, the focus is on prediction, and the formula allows for a choice of accuracy depending on the number of future "y" values. The Temporal Difference theory seems to suggest a linear, formulaic setup where the number of discounted "y" values used for estimation can be initiated as a variable. Q-Learning does not appear to have this facility as it explicitly depends on one single previous value, and has more parameters to work with. These factors most likely account for Temporal Difference Learning's superiority as an algorithm.

The differences between the two algorithms seem to be more obvious with RL2. Taking into account what was discussed about failed states, it appears that because RL2 essentially calls the reset function fewer times, the data does not lose meaning as often since there are fewer abrupt changes in the state altering process. Therefore, the Temporal Difference algorithm makes more robust use of the data and makes more accurate predictions, helping all of the states to converge on a value faster.

The first three parts of the research question appear to have been addressed fairly and comprehensively. Each important aspect has been covered and fair conclusions have been drawn. Moving onto the fourth, which is perhaps not as strong due to the ambiguous nature of the results and the small, anecdotal sample size, participant data shall now be discussed.

From what can be established, there was no participant who felt like the application was intruding in the enjoyment of the game itself, hence the results displayed from the enjoyment graph. But it would not be quite justified to draw any conclusions in regards to how well they felt the application was helping them with their goals. This may be due to the advice itself not serving its purpose since there was a limited availability to what could actually be said in order to boost platform game skills: some participants suggested that the advice could not amount to more than common knowledge, especially amongst those who had played platform games regularly.

The latter therefore did not quite contribute to the “believability” question. And it is hard to acquire any meaning from the values that describe the changes in the participants’ skill. Even although the descriptive words “better” and “worse” produce fairly clear results indicating that participants suggested an increase of skill, the more qualitative skill measurements seem to contradict these findings.

Criticisms

It was found that player feedback itself could not be integrated with the Reinforcement Learning algorithms themselves for the simple fact that there was no way to acquire this feedback meaningfully during the offline period. The player would need to go through the process of playing the game and using the system beforehand, which could not be done. A multi-threaded solution could not have solved this problem because the application would have still lost its potential to give immediate responses: the algorithms were found to take about ten minutes each to process, and it would have been too much waiting time on the player’s part. However, this does not reduce the project’s potential for a dynamic system: by updating the success value of the advice-giving system, depending on the methods prepared by the algorithms, and based on the feedback from the player, the program could still successfully utilise it for selecting different batches of advice units.

It should be noted that according to the Toolbox, the deduced probability values of a Reinforcement Learning environment are organised in terms of the actions instead of the states. This does not seem to be too much of an obstacle in the way of obtaining behavioural and feedback ratings: in order to get a value for a state, it is a case of averaging each action value, or simply holding all of the values and making comparisons as they are.

There appears to have been a successful integration overall between the Reinforcement Learning algorithms and the structure of the “helper bot” despite the noticeable setbacks. Significant results were drawn as a result of comparing two sets of algorithms amongst two systems. Hopefully, most questions that the reader may have in mind have been answered about the results and their implications. Temporal Difference Learning appears to be the preferred algorithm choice due to the estimative nature and the reliance on data patterns that resemble functions. This is a similar approach that is taken by dead reckoning systems in order to predict future data, and it has also been proven useful there (Aronson, 1997).

Findings

In contrast to what was suspected, the addition of more states to the systems defined for this project helped to contribute to the performance of the agent, and helped to show a clear difference between Temporal Difference Learning methods and Q-Learning methods. Variables such as the average reward and the average number of episode cycles have answered questions primarily about the first two parts of the research question that deal with both RL1 and RL2 in regards to their tasks, and contributed further still to the third part dealing strictly with computational performance. The variable "ticks" has answered required questions on the performance of the system as a whole, thus greatly contributing to the third part of the research question. The fourth part that concerns qualities on the player's end, however, could have been met with improvement through the use of more trustworthy data.

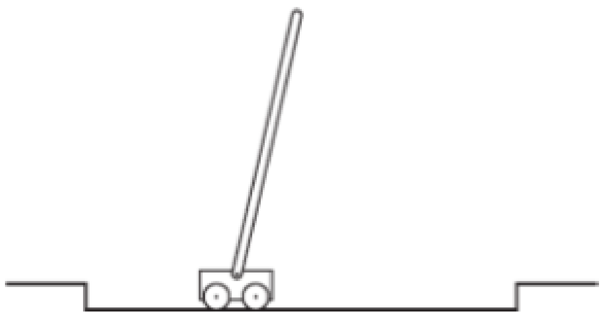


Figure 14: Cart-Pole Illustration

It was discovered during the course of this project that Reinforcement Learning is a concept that is more suited towards models that have *continuous* state variables as opposed to *discrete* state variables. Referring to the Cart-Pole Tutorial example again (Neumann, 2006), the variables being modified there were physical components such as velocity and force of the cart (see Figure 14): these continuous variables differ from discrete ones in that they can take *any* real number between two extremes, whereas discrete variables have a limited set of rational values such as "0", "1" and "2" in the case of RL1, but not "1.5" or "1.2439". It is still feasible to use discrete variables, but these are better used in environments with outcomes that are not as clear as merely comparing a state with what would be a "perfect" rating of an advice output, such as the *Shortest Path Problem* Tutorial presented on the Reinforcement Learning Toolbox website (Neumann, 2006).

Issues such as this perhaps suggest that the project's goal of developing a "helper bot" application has put Reinforcement Learning algorithms in a context that they do not belong in. They have definitely contributed reasonable results suited for their own area of research, but practically they have been made to fit a purpose not explicitly built for them, and that has limited the project's capabilities. The problem can be best described as the following: Reinforcement Learning is a problem solving area designed to work out the probabilities of success of certain state-action combinations when used in certain situations; it is not primarily suited for providing these values in more deterministic, confined settings where simpler, mathematical functions can calculate the values just as easily.

Future Research

In the wider context, this setback is not really important because the agent as a whole met the definitions required by the algorithms. In fact, the project may have contributed research that is still respectable since the more confined settings may have given more objective evidence about the algorithms.

Further uses of this type of application might include Reinforcement Learning systems that select the best advice from a much larger and much improved database where multiple scripts can be tested. A number of contributors and developers could get together in order to develop plenty of advice that is more constructive than the examples given in this project. They may also regularly update the advice selections over time in order to use what is best appreciated by player, and offer these selections to other platform games that could do with improving their tutorials or hints.

Also, it might be worthwhile to build a Reinforcement Learning system that controls the character *Maryo* in the *Secret Maryo Chronicles* game with the aim of getting to the end of a level: actions such as "jump" and "shoot" could be attempted at certain areas in the level, and these areas might be thought of as the states. This approach could be a worthwhile alternative to determining what advice should be outputted as a direct comparison can be made between the player’s actions are certain times and areas in the level, and whatever the algorithms worked out were the best actions. This way, stronger advice can be selected depending on how “off course” the player is from the agent, and success can be measured depending on the behavioural adjustments and the approval of advice from the player.

Scenarios such as this give lots of opportunity for Reinforcement Learning techniques to be used on a wider scale. Bearing in mind what can be taken from this project, the techniques are more primed for the act of solving problems, and it may be a necessity when it comes to optimising NPC opponents in a video game environment. Of course, care must be taken to make the A.I. “appropriate” and not “clever”, as mentioned in the Introduction. Algorithms such as Reinforcement Learning may present enemy behaviours that make the enemy too difficult for the player to win over, for example. Any future programmer looking to incorporate techniques from this field of research is advised to remember this principle, and pay close attention to the design of the game environment: what constitutes a state, what constitutes an action, and what is required of the agent. Reinforcement Learning is a broad subject: it can incorporate most systems that are designed with these clarified quantities, and as long as the application designers have a specific problem or goal in mind that cannot be solved by simpler, more typical means, they ought to attempt some of the techniques that have been discussed.

APPENDICES

Appendix A - Questionnaire

YOUR FULL NAME:

BEFORE PARTICIPATION

How often do you play computer games? Type one answer:

At least:

- Once a day
- Once a week
- Once a fortnight
- Once a month
- Rarely
- Never

Answer:

How well do you rate your playing skills towards games generally? Type a Value

Bad		Average		Good
1	-	5	-	10

Value:

Do you play platform games? Type "Yes" or "No".

Yes/No:

<u>If so, how well do you rate your playing skills towards those games? Type a Value</u>

Bad		Average		Good
1	-	5	-	10

Value:

AFTER PARTICIPATION

<u>How helpful do you feel the application was in regards to achieving the goal(s) you specified at the start? Type one answer.</u>

Very Unhelpful---Unhelpful---Neither---Helpful---Very Helpful

Answer:

<u>How helpful do you feel the application was in regards to making the game an enjoyable experience? Type one answer.</u>

Very Unhelpful---Unhelpful---Neither---Helpful---Very Helpful

Answer:

<u>How well do you now rate your playing skills towards games generally? Type a value.</u>

Bad		Average		Good
1	-	5	-	10

Value:

<u>How would you now judge your skills in regards to playing games generally?</u>

<u>Type one answer.</u>

Much Worse---Worse---No Difference---Better---Much Better

Answer:

(If YES to platform game question asked before participation)

How well do you now rate your playing skills towards platform games? Type a value.

Bad		Average		Good
1	-	5	-	10

Value:

How would you now judge your skills in regards to playing platform games? Type one answer.

Much Worse---Worse---No Difference---Better---Much Better

Answer:

Appendix B – Additional Questionnaire Sample Data

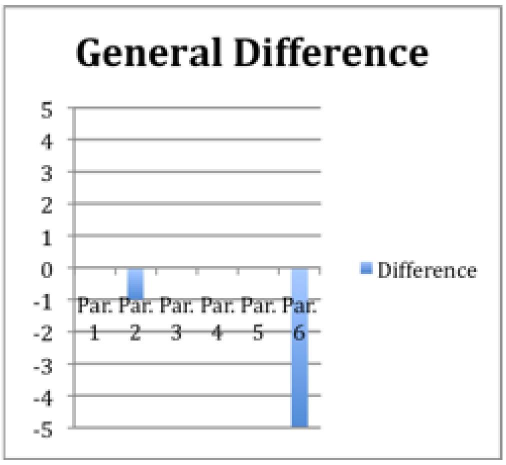


Figure 15: General Skill Difference

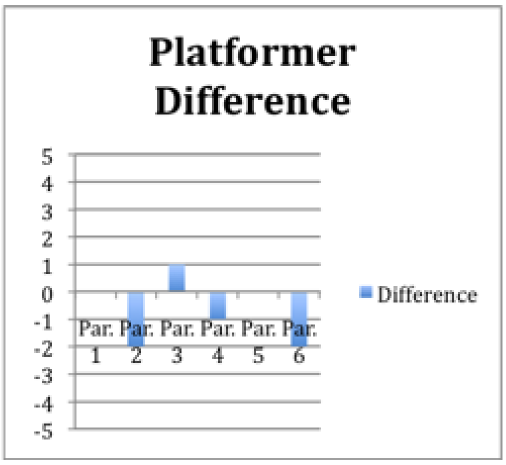


Figure 16: Platform Skill Difference

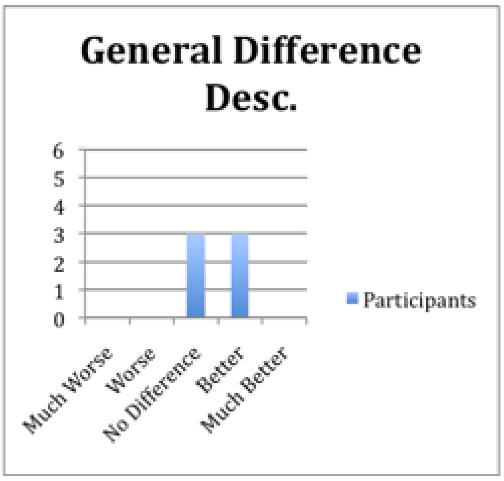


Figure 17: General Skill Difference Description

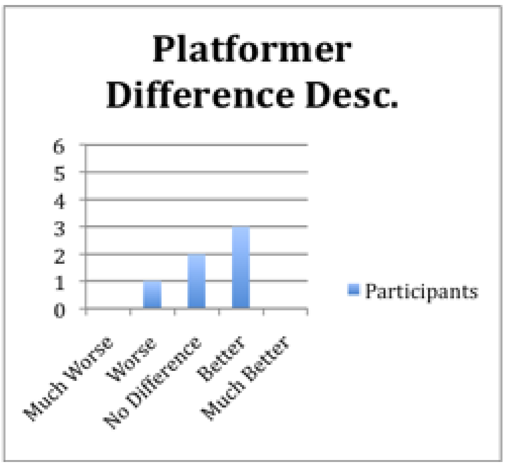


Figure 18: Platform Skill Difference Description

This data turned out to be fairly meaningless due to both the contradictory nature of the answers and the lack of any general trends that would have prompted clear conclusions. Therefore, the data was judged to be unfit for discussion and was not included in the main dissertation.

Appendix C – Advice Examples

Goal: Get to the end of the level ASAP.

“If you see coins, do not be tempted to go out of your way to collect them. If you wish to take little time to get to the end of the level, ignore them as they will slow you down.”

“There is not much need to get items or powerups unless you wish to shield yourself against one hit. Avoid them where you can and only grab a powerup if it is convenient.”

“As soon as a level begins, begin to move in the right direction immediately. There is not a lot of need to study the level from what you see initially, so start as soon as you can.”

“Do not hesitate to jump between platforms and do not take too much effort into dodging enemies: these things are best done when there is less care given with jumping precisely.”

“Hold onto the boost button (L1) and do not let go unless it is absolutely necessary. This approach takes practice, but it will greatly reward your time taken to finish the level.”

Goal: Die as few times as possible.

“If you power up as soon as you can, and stay powered up, you will survive better. Spare powerups will be given to you automatically by the item box if you take a hit.”

“Since the aim is survival, you will want to take care moving, so avoid holding onto the boost button wherever you can. This way, you will be more accurate with your jumping.”

“As you are crossing Maryo between moving platforms, or between risky enemies, hold onto the jump button for as long as you can in order to keep control of Maryo.”

“If you see multiple enemies in an area of the level, try not to take the risk of getting a “combo” hit with them: focus on staying alive and killing few enemies.”

“If you need to cross dangerous platforms, try jumping on the spot a little bit first before you cross. You will have a better feel for the controls and will be able to cross safer.”

Appendix D – Project Timetable

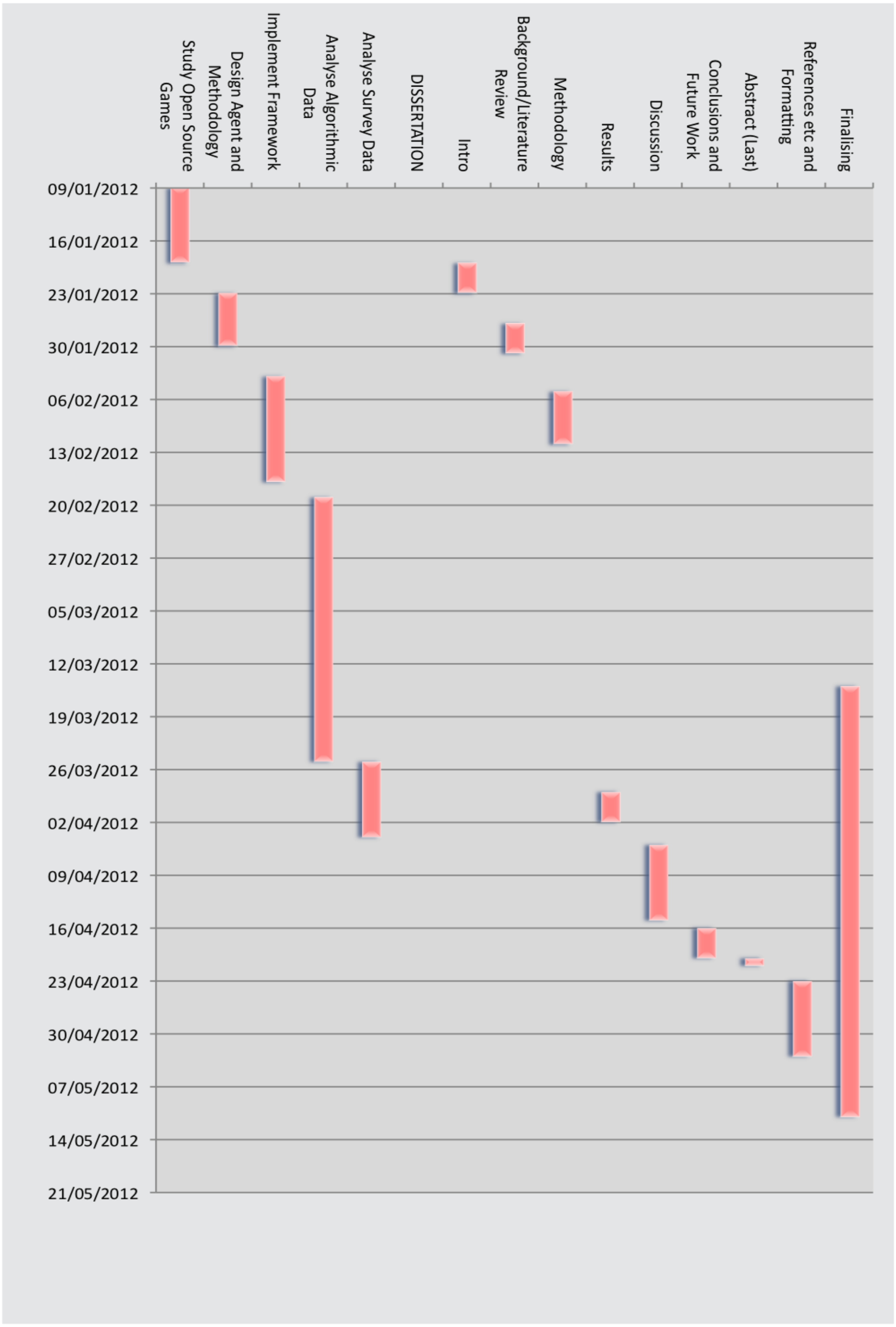


Figure 19: Gantt Chart of Timeline

Appendix E – Project Proposal

UNIVERSITY *of* ABERTAY DUNDEE

James David Hogg

Honours Project Proposal

Using Reinforcement Learning to Develop an A.I. Agent that Aids a Video Game Player's Intentions by Studying Behaviours and Receiving Feedback.

Institute of Arts, Media and Computer Games

BSc Computer Games Technology (Hons)

9th December 2011

Abstract

Video game A.I. has been geared towards generating suitable behaviours for in-game enemies first and foremost, but any research into the limits of player-friendly A.I. agents is neither obvious nor substantial. This document proposes a project that investigates the position such an agent can take when enhancing a player's immersive experience within a game according to what he or she chooses it to be. The "bot" will learn and adapt depending on player behaviour and approval through feedback. Development will be amongst an open-source Platformer Game for means of practicality, and will utilise advancements in technology previously not available to the genre. The prediction is that the results will raise consciousness about the potential benefits such a system could bring to gaming interaction, storytelling and likewise areas of game development. The project will aim to give the topic credibility by presenting a believable and efficient final result.

The approach will focus on reinforcement learning techniques as a means of determining the outcome that best suits the player's needs. Alternative techniques include that of decision tree learning through ID3 algorithms and a combination of both this and reinforcement learning. Justifications for this choice are highlighted in the Literature Review, as well as the choice of video game genre. A framework, strategy and design have been suggested for carrying out the research itself. These briefly discuss the basis for putting the components together, and a method for obtaining results and forming conclusions follows along with a risk assessment. The final study is set to contribute its findings and limitations to future programmers and researchers who may also be interested in applying reinforcement algorithms for A.I. learning agents.

Contents

Figures

1. Introduction

1.1. Background

Video game players have always had a longing need for their in-game characters to behave not just in an appropriate manner but also in a clever and intriguing one. The scope of the Artificial Intelligence field is large just for the technology of video games alone. No doubt, there is plenty that can be researched on enemy behaviour or even level manipulation. But curiously, there does not seem to be much immediate inquiry into the role that an artificial intelligence agent can play when it comes to aiding a player's journey within a game as opposed to obstructing it. This role could provide considerable, beneficial improvements for player experience and immersion.

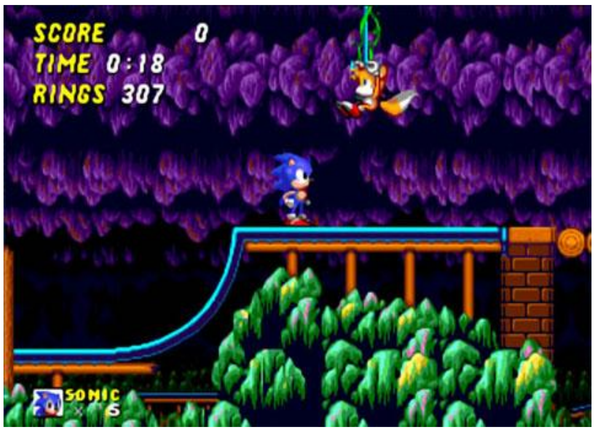


Figure 20: ***Sonic The Hedgehog 2***

A "helper bot" such as this may take the form of a tutorial that gives hints to players, or as a sidekick or ally (such as Tails from *Sonic The Hedgehog 2* (1992), Figure 1). This particular investigation will explore the possibility of generating believable sidekick aid that will be reinforced according to the player's habits and, equally important, the player's *conscious approval* of the aid itself. These vital factors form each player's unique experience, depending on what in-game goals are favoured.

The aim is to discover these kinds of desired goals in real-time and then provide help: such as getting through a level, gathering items, exploring the world, shooting enemies, etc. The core A.I. techniques involved will be that of reinforcement learning and possibly decision learning trees, which have been chosen from Artificial Intelligence for Games (Millington and Funge 2009) after careful study. The focus aims in these directions because the success of this system will depend on its ability to learn and/or deduce the player's intentions.

It should be noted that in the field of gaming A.I., we are primarily concerned with *illusions* of an intelligent agent as opposed to the more genuine and sincere approach, despite increasing demand for “truer” A.I. (Ponsen 2004). Developing an academic level of A.I. for real-time video games would be disastrously costly, so we must make do with taking shortcuts, especially how they will most likely suffice anyway. In *Reinforcement Learning: An Introduction* by Richard Sutton and Andrew Barto (2005, p14), they relate to this principle: “Rather than directly theorising about how people or animals learn, we explore idealised learning situations and evaluate the effectiveness of various learning methods.”

1.2. Research Question

“What are the most practical methods for generating a believable reinforcement-learning A.I. agent that aids players’ behaviours and improves its responses depending upon player feedback, and what consequences will the algorithms have on performance?”

This research question can be approached by breaking it down into further, specific questions and methods:

Within the field of reinforcement learning, what methods of determining players’ consistent behaviours are most accurate?

- Establish the most appropriate behaviours and goals to monitor.
- Investigate algorithms for comparison of data.
- Use the test application to observe the most sensible results.

On what criteria can this A.I. agent be deemed “believable” when it responds to its analysis of behavioural data?

- Interview a suitable range of test subjects and obtain critique.
- Measure the subjects’ “ratings” they give for the agent’s believability after entering feedback into the test application, and then note its variations.
- Determine to what extent the agent is correctly serving the purpose of helping the player achieve his or her goals.

How can the A.I. agent gather suitable player feedback and use it to adjust its analysis of players' behaviours to behave more suitably?

- Research methods for obtaining meaningful feedback that will successfully let the player approve or disapprove of given aid.
- Treat feedback as variables that contribute to the decisions made from the behavioural database (the "rewarding" variable?).
- Take care when adjusting mature algorithms with new feedback data.

What are the consequences of the reinforcement algorithms on overall application performance in real time?

- Adhere to a strict set of established benchmarks within the test application.
- Compare and contrast required algorithms against these benchmarks.
- Draw reasonable conclusions from the results.

2. Literature Review

2.1. Reinforcement Learning Overview

Reinforcement learning is the process of taking an application or agent, using it to execute actions, and rating the success of those actions according to a feedback function from the environment (Sutton, Barto, 2005). The value that influences this function, which resembles a fitness function that is often discussed in A.I. selection algorithms (such as Schwab, 2008 and Millington, Funge 2009), is often called the reinforcement, or the *reward* (Fürnkranz, 2001).

If the function that calculates an action's worth is returning a value that is slowly converging towards the limits (usually between -1 and 1) then the problems and rewards are being consistent. If the algorithm reaches the highest limit or an asymptote to that limit, it can safely be established that it has "learned" the problem's solution (Millington and Funge, 2009). The "helper bot" idea could work on this concept reasonably well: the experimentation will come from the player's ability to manipulate the reinforcement value and lead the algorithm down towards his or her favoured path, albeit on a very encapsulated level.

Usually the actions in question are associated with a specific state. Game programmers may immediately see a reasonable problem with this approach: excessive combinations of states and actions are bound to result in slow performances even with very efficient algorithms. However, some studies on this approach within the Real Time Strategy genre have demonstrated that it can be done well (Madeira, Corruble and Ramalho, 2006). Also, the choice of genre for this project is classic Platformer, which requires far fewer resources. This will be discussed further in the methodology section. Of course, sensible care must be taken to prevent the application from overloading the hardware.

One particular well-known reinforcement algorithm that updates and utilises quality information on state/action combinations, the Q-learning rule, is as follows (Millington, Funge, 2009):

$$Q(s, a) = (1 - \alpha).Q(s, a) + \alpha.(r + \gamma.\max(Q(s', a')))$$

This function depends on a state-action combination's previous value "Q(s , a)" and returns it a new fitness value. It works with a rate of learning "α" that varies the linear blend, a discount rate "γ" that controls how much the old "Q" state-action value depends on the new "Q" state value, a reinforcement rate "*r*" that depends on the specified rewarding feedback, and a return of the maximum value action from the increasingly favourable state set "max(Q(*s*' , *a*'))". All of these components gradually create a convergence towards the fittest state-action combination.

There are numerous algorithms that revolve around this principle. Fang and Ting (2009) experiment with a couple of variations in their research in order to influence the behavioural difficulty of NPCs within a tank battle game, and also state that, "it is one of the unsupervised machine learning methods in the area of artificial intelligence. The methodology is developed based on the concept of 'trial-and-error', and the result of each trial-and-error will be saved as a 'delay reward'. " (p1031) "Unsupervised" means that the algorithm is not constrained to arrive at a specified result: it can arrive at any final outcome. For the "helper bot" this may well be necessary because each player is going to want something different from playing the game. Therefore we want the algorithm to attempt to find this for itself.

An important criticism of the reinforcement learning algorithms is the rate at which it consumes application resources (Funge 2004). Because of this, it is recommended that the algorithm perform offline when it can, i.e. making calculated results before their required use by the application (Millington and Funge 2009). It is also stated that if strategies tend to change overtime within the application (in this case the "strategies" would refer to the player's behaviour) then the learning algorithm must be stable and robust enough to handle them, otherwise the results will be skewed out of proportion. Therefore, when the project is executed, care must be taken to allocate tasks correctly for online and offline purposes, and to become very familiar with the algorithm's components.

The algorithm must also be restricted from learning undesired or stupid behaviours. What could be a more serious situation is if a small, unwanted behaviour has a ripple effect throughout the rest of the algorithm. For instance, *Black and White* (2001) by Lionhead Studios (Figure 2) had trouble preventing the creature from learning unwelcome behaviours that became frustrating to the player.

Figure 21: ***Black and White***

2.2. Alternative Learning Techniques

There are some significant alternatives to the reinforcement approach of learning behaviours that must be discussed in order to justify the chosen approach. If the problem of developing a believable, practical A.I. agent is to be taken seriously, this justification must be necessary.

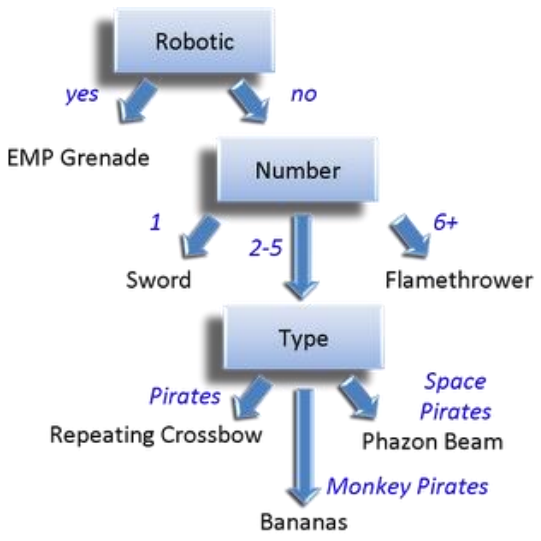


Figure 22: A Decision Tree Example

Decision trees (Figure 3) are used for leading the intelligent agent from one state into another depending on the results of a logical decision. The states eventually arrange themselves into the structure of a branching tree. *Black and White* (2001) is noted for implementing this approach as part of its creature system (Wexler, 2002). Depending on the scenario, the trees can be manually set up by the programmer. However, it is often required that the program dynamically constructs trees with an algorithm by evaluating chosen state tables (that note every possible state transition). This algorithm is called ID3 (Millington and Funge 2009) and makes the tree the least complicated it has to be, which is similar to Occam's Razor: the simplest explanation (tree representation) is probably the best one (Online Decision Trees Tutorial). Another way of expressing this famous heuristic is "do not multiply unnecessary constants", and the ID3 algorithm's job is to remove all unnecessary state transitions.

The A.I. agent will most likely head in this direction as a backup in case reinforcement learning does not make any progress in the required time period. The ID3 trees can be utilised by constructing the tables as the game progresses (strictly ID4) according to the occurrences of significant events.

Genetic Algorithms were discarded as an option: they involve the crossover-reproduction, mutation and selection of a population set in order to achieve the fittest solution, similar to that of biological evolutionary theory by natural selection (Carr, 2005). With this technique it is hard to measure the experience that is built up overtime, like reinforcement learning. Considering how the "helper bot" requires this experience, such as making successful observations based on player mood changes and taking appropriate actions, it was decided that this approach not be used.

Neural Networking techniques were also examined. These involve dividing states or behaviours into multiple interconnecting nodes that output a result depending on a state input, allowing for the nodes to be modified online and to generalise decisions. Even although they can be successfully implemented in some games (Funge, 2004) it suffers from producing unstable results that worsen further during its learning. For this A.I. agent it is preferred to have more control over the outputs. And given the significant complexity involved, they are probably best left.

If multiple fields of techniques end up working together in the project, it must be remembered that it will be difficult to obtain the required results needed for the project. The article *Combining A.I. Methods for Learning Bots in a Real-Time Strategy Game* by Baumgarten, Colton and Morris (2008) notes game performances from the A.I. such as wins, weapon use and equipment use. However, there is nothing there that specifies how well each specific technique is performing. Perhaps this does not matter since the end results take priority, but if there were room for more efficiency it would be wrong to be closed-minded towards it. So if more techniques end up working with reinforcement learning for the A.I. agent, strict rules for analysis must be applied.

Reinforcement learning works best because it arrives at a solution through an easy, practical direction of rewarding what works. Even although it is strictly unsupervised there can still be robust controls implemented, as opposed to what could be considered as unnecessarily over-computational methods.

2.3. Game Genre Choice

It has been decided that the best way to express this "helper bot" will be in conjunction with a Platform Game. Noted as "the primary staple of the console world" (Schwab, 2008, p143), this long-lasting genre would be suitable because of the simplicity in obtaining behavioural patterns (*Ratchet and Clank* (2002), shown in Figure 4, shows a world where data collection would be fairly trivial). Schwab also notes the need for these games to have better "helpers": even although the genre does not have the complexity levels of the Real Time Strategy genre (in particular the simpler A.I. techniques for enemy behaviours), perhaps a degree of complexity could be introduced as a "helper bot", especially considering the technological advances that older Platform Games did not have.

Figure 23: ***Ratchet and Clank***

2.4. Note On Believability

There has always been a need for increasing appeal from interactive entertainment. Beal *et al* noted this as far back as 2002 in *Intelligent Modelling of the User in Interactive Environment.* In order for this A.I. agent to be appreciated by the players it must correctly deduce what goals the player wants from the game and help accordingly. This ought to be the way forward in terms of believability, and will also be judged by testers. Adaptation and learning has been recognised by the A.I. community as critical concepts for decision-making (Madeira, Corruble and Ramalho, 2006), therefore the premise is supported.

3. Methodology

3.1. Framework

An open source platform game will host the intelligent agent. The aim is to get the agent to control a real character in the game and give physical help by executing algorithms that evaluate behavioural patterns. At the very least, text output will be given. This will also be necessary for allowing the player to rate his or her approval of the agent's responsive behaviour. Possible games include *Yo Frankie!* (2008), a 3D platform game written in the Crystal Space framework (C++) and *Secret Maryo Chronicles* (2003), a 2D platform game written in SourceForge (web-based). These are shown in Figures 5 and 6 respectively.

Figure 24: ***Yo Frankie!***

Figure 25: ***Secret Maryo Chronicles***

3.2. Strategy

The project will initially focus on developments that monitor and interact with basic behaviours. This makes sure that the algorithms can be tested early on for their workability, that a base framework can be added to the current one within the source code, and that the game runs the script without trouble. After this, more behavioural states will be included and then professional testing and evaluation can begin. This gathering of results will continue until near the deadline date and the beginning of the dissertation.

(Attached to the appendices is a Gantt Chart that plans the currently perceived stages of the project execution.)

3.3. Design

Typical moves, speeds, and even button presses will be noted as some of the first habitual variables. The most appropriate behaviours will be planned for and listed in table form. Meanwhile, the game will be studied closely: both its source code and its rendered output on screen. Suitable pseudo-code and class UMLs will be drawn up on paper that accommodates the provided resources. Testing scripts that print statistics on the screen will be included in the design right from the start, and standard conditional benchmarks will also be established during this stage.

3.4. Data Collection and Analysis

In order to obtain data, a wide comparison of the required algorithms must be carried out in order for performance to be measured, such as run-time costs, memory usage, CPU usage, frame rates, fitness values, etc. In the case of noting how well the agent is doing on a practical level for players within the game, interviews and questionnaires will be carried out with willing participants. There will also be performance ratings in this area too, such as the variables recorded by McPartland and Gallagher in their 2011 study ("Table V, Statistical Results For Arena Map Experiment"). Literature such as this must be examined for any relevant criticism aimed at the chosen analytical methodology.

3.5. Risk Assessment

As previously stated, there is significant risk involved with the reinforcement learning approach to this project due to the field's tendency to record and output unwanted behaviours, and depending on the structure of the open-source program, it may be possible. If this issue cannot be resolved within a suitable period, the focus of the project will shift towards an alternative (while recording any significant content from the past). These include decision tree learning and dynamic scripting approaches. But reinforcement learning will still be critically discussed: the failures and choice of direction change will all be justified in the final dissertation.

4. Summary

The project as a whole will try to raise interest in this significant gap of video game A.I. research that involves A.I. agents assisting a player. It is hoped that encouraging results can be obtained that demonstrate the possibility of an agent that can accurately deduce what the player wants out of a particular game in terms of his or her own experience with it, and aid that effort. The "helper bot" has a reasonably healthy and plausible field of learning through reinforcement at its disposal that is worth exploring. If this field successfully fulfils its role and overcomes the stated obstacles then hopefully it will give grounds for future developers to take notice and make their own advancements and discoveries. Because the platform genre has much more technological progression than years before, a call for more sophisticated A.I. behaviour is most likely just. At the very least, it is hoped that respectable methods of analysis and evaluation can be developed during the execution of the project. If future programmers were to attempt something similar, the conclusions found could give guidance for their investigations. Taking into account the criticism, the prospect of reaching encouraging results seems very promising.

5. Participants' Consent

Anyone willing to participate for research will be guaranteed ethical protection as stated by the University of Abertay Dundee's guidelines.

References

Aronson, J. , *Dead Reckoning: Latency Hiding For Networked Games*. Available: http://www.gamasutra.com/view/feature/3230/dead_reckoning_latency_hiding_for_.php [2012, 05/09].

Barto, A.G. 2007, "Temporal difference learning", *Scholarpedia,* vol. 2, no. 11, pp. 1604.

Baumgarten, R., Colton, S. & Morris, M. 2009, *Combining AI Methods for Learning Bots in a Real-Time Strategy Game*.

Beal, C., Beck, J., Westbrook, D. & Cohen, P. 2002, *Intelligent modeling of the User in Interactive entertainment,* University of Massachusetts at Amherst.

Björnsson, Y., Hafsteinsson, V., Jóhannsson, Á. & Jónsson, E. 2004, "Efficient Use of Reinforcement Learning in a Computer Game", *Computer Games: Artificial Intelligence, Design and Education (CGAIDE'04),* pp. 379.

Carr, D. 2005, , *Applying reinforcement learning to Tetris*. Available: www.colinfahey.com/tetris/ApplyingReinforcementLearningToTetris_DonaldCarr_RU_AC_ZA.pdf [2012, 05/09].

Dai, W. , *Crypto++ 5.6.0 Benchmarks*. Available: http://www.cryptopp.com/benchmarks.html [2012, 05/09].

Eggers, S.J., Emer, J.S., Leby, H.M., Lo, J.L., Stamm, R.L. & Tullsen, D.M. 1997, "Simultaneous multithreading: a platform for next-generation processors", *Micro, IEEE,* vol. 17, no. 5, pp. 12-19.

Funge, J. 2004, *Artificial Intelligence for Computer Games : An Introduction,* A K Peters, Limited, Natick, MA, USA.

Fürnkranz, J. 2001, "Machine Learning in Games: A Survey", *Machines That Learn to Play Games, chapter 2,* pp. 11.

Gasser, M. 2012, , *Introduction To Reinforcement Learning*
. Available: http://www.cs.indiana.edu/~gasser/Salsa/rl.html [2012, 05/09].

Madeira, C., Corruble, V. & Ramalho, G. 2006, , *Designing a Reinforcement Learning-based Adaptive AI for Large-Scale*
Strategy Games [Homepage of American Association for Artificial Intelligence], [Online]. Available: http://www.aaai.org/Papers/AIIDE/2006/AIIDE06-026.pdf; [2012, 05/09].

McPartland, M. & Gallagher, M. 2011, "Reinforcement Learning in First Person Shooter Games", *Computational Intelligence and AI in Games, IEEE Transactions on,* vol. 3, no. 1, pp. 43-56.

Miller, N. *Questionnaires,* University of York. Available: http://www.economicsnetwork.ac.uk/handbook/questionnaires/21 [2012, 05/09].

Millington, I. & Funge, J. 2009, *Artificial Intelligence for Games, Second Edition,* Morgan Kaufmann Publishers Inc.

Nashvili, M. 2004, , *Decision Trees Tutorial* [Homepage of Department of Computer Science, University of Birmingham, UK;], [Online]. Available: http://decisiontrees.net/?q=book/export/html/16; [2011, 12/04].

Neumann, G. 2006, , *Reinforcement Learning Toolbox 2.0*. Available: http://www.igi.tugraz.at/ril-toolbox/links/index.html; [2012, 05/09].

Neumann, G. 2006, *Reinforcement Learning Toolbox Tutorial (Cart Pole Example)*, Institute for Theoretical Computer Science, TU Graz.

Neumann, G. 2006, *Reinforcement Learning Toolbox Tutorial (Shortest Path Problem in a Grid-World)*, Institute for Theoretical Computer Science, TU Graz.

Neumann, G. *The Reinforcement Learning Toolbox*, *Reinforcement Learning For Optimal Control Tasks*, University of Technology, Graz.

Pfeiffer, M. 2006, *Tutorial For The Reinforcement Learning Toolbox*, Institute for Theoretical Computer Science, TU Graz.

Ponsen, M. 2004, *Improving Adaptive Game A.I. With Evolutionary Learning*, Faculty of Media & Knowledge Engineering, Delft University of Technology.

Rabin, S. 2002, *AI game programming wisdom*, Charles River Media.

Schwab, B. 2004, *AI Game Engine Programming*, Charles River Media, Hingham, MA, US.

Sutton, R. & Barto, A. 1998, *Reinforcement Learning: An Introduction (Adaptive Computation and Machine Learning)*, The MIT Press.

Valpola, H. 2000, *Bayesian Ensemble Learning for Nonlinear Factor Analysis*, Helsinki University of Technology, Neural Networks Research Centre.

Wexler, J. 2002, *A look at the smarts behind Lionhead Studio's "Black and White" and where it can and will go in the future*, http://www.cs.rochester.edu/~brown/242/assts/termprojs/games.pdf; edn, University of Rochester. [2012, 05/09].

Yung-Ping Fang & I-Hsien Ting 2009, "Applying Reinforcement Learning for Game AI in a Tank-Battle Game", *Innovative Computing, Information and Control (ICICIC), 2009 Fourth International Conference on*, pp. 1031.

Bibliography

AI Game Engine Programming (2nd Edition), 2008, Course Technology, Boston, MA, USA.

Chrisley, R. & Begeer, S. 2000, *Artificial intelligence : critical concepts. Vol.4,* Routledge, London.

Du, Y., Cui, S. & Guo, S. 2009, , *Applying Machine Learning In Game AI Design*. Available: http://cs229.stanford.edu/proj2009/DuCuiGuo.pdf;. [2012, 05/09].

Erev, I. & Roth, A.E. 1998, "Predicting how people play games: Reinforcement learning in experimental games with unique, mixed strategy equilibria", *American Economic Review*, vol. 88, pp. 848-881.

Gowda, S. 2011, , *5 Best Open Source/Free Platform Games for Linux* [Homepage of Solancer], [Online]. Available: http://solancer.blogspot.com/2011/06/5-best-open-sourcefree-platform-games.html;. [2012, 05/09].

Kirby, N. 2010, *Introduction to Game AI,* Course Technology, Boston, MA, USA.

Lake, A. 2010, *Game Programming Gems 8,* Course Technology, Boston, MA, USA.

Lecky-Thompson, G.W. 2008, *AI and Artificial Life in Video Games,* Course Technolgy, Boston, MA, USA.

Livingstone, D. 2006, "Turing's test and believable AI in games", *Comput.Entertain.*, vol. 4, no. 1, pp. 6.

National Research Council Staff 1997, *Computer Science and Artificial Intelligence*, National Academies Press, Washington, DC, USA.

Palmer, N. 2007, , *Machine Learning in Games Development*. Available: http://ai-depot.com/GameAI/Learning.html;. [2012, 05/09].

Siler, W. & Buckley, J.J. 2005, *Fuzzy Expert Systems and Fuzzy Reasoning*, Wiley, Hoboken, NJ, USA.

Spronck, P. & Herik, J.V.D. 2004, , *Game Artificial Intelligence That Adapts To The Human Player* [Homepage of ERCIM News], [Online]. Available: http://www.ercim.eu/publication/Ercim_News/enw57/spronck.html;. [2012, 05/09].

Spronck, P., Ponsen, M., Sprinkhuizen-Kuyper, I. & Postma, E. 2006, "Adaptive game AI with dynamic scripting", *Mach.Learn.*, vol. 63, no. 3, pp. 217-248.