Juan Miguel Huertas Delgado

Date: 27/03/2015
Lecturer: Professor Jeremy Gow

[USER INPUT PREDICTION]

Assignment for the module "Artificial Intelligence in Games" for the Master of Science in Computer Games & Entertainment @ Goldsmiths, University of London. - 16 pages -

Contents

1. Introduction

1.1 Description assignment

The topic of this assignment was to develop a small demo implementing one of the concepts studies along the course of Artificial Intelligence in Games. For this assignment, after doing some extensive research on Montecarlo Tree Search (MCTS) techniques (and different kind of applications in 'real-time games'), I've realized that it would be wise to 'change' the topic initially thought for this assignment. I consider that it was a right decision to change the topic, as it was getting close to the dead-line and there was little progress regarding MCTS.

The final topic of this assignment is the implementation of a small fighting game and an AI to control the characters. To do so, I have done research on 'prediction models'. I have researched with Markov models and N-gram Language Model (techniques widely used for 'next word prediction' problems).

1.2 Montecarlo Tree Search, a few words

I will spend a few words on some ideas during the research of this initial topic for the assignment. After reading an extensive Survey on different MCTS methods (Cameron Browne, 2012) I invested some research time in reading about Multi-player MCTS, Multi-agent MCTS and Real-time MCTS. After some further research, I simplified the initial assumptions for the project (multi-agent simulation with MCTS) to a single agent simulation in a 'complex' world.

To face this problem I started to produce some 'models' on how would the world be represented following an example of MCTS applied to StarCraft (Soemers, 2014). Following this models, I had to design a set of states for the world and a set of actions. I was working on simplifying the possible actions of the model just to "Go_To(X)" and "Interact_With(X)", where X would be a set of different elements or positions (Water, Fire, Door, Food…). Therefore the Agent would be able to receive damage from the environment, interact and change the environment, fill its supplies and renew its energy.

Sadly, after doing some further research on MCTS (Association for the Advancement of Artificial Intelligence, 2014) and some planning for the implementation of the project, I've realized that it would require more time that the time I had available and, even though I had have already done a considerably amount of work (modelling the system), I decided to change topic for the sake of being able to produce a good-quality demo.

1.3 Input user prediction topic

The core motivation of choosing this topic was the potential of such techniques. If we are able to predict accurately the input that the user is going to produce, we can 'adapt' our system to offer a desired experience to the player. This technique can be used for a big variety of topics, from helping 'autocompleting' text in smartphones or searches on the internet, passing to creating more believable agents (humans always try –without succeeding completely- to predict the outcome of their actions and the reaction of other people), up to trying to adapt an user interface by how the user is interacting with the system.

Therefore I wanted to cover the topic of 'learning in games' but I wanted to stay away from Neural Networks, given that the outcome produced is difficult to control and that we would need lots of training time to teach our AI how to behave. I wanted to be able to produce good-quality results in the less available time. To produce such behaviour I did some research mainly focused on N-grams and on their different 'real-world' applications. I looked as well some Markov models. (In the section 2. User Input Prediction I will explain in further detail regarding the NGram research).

To be able to solve this problem in such a short time, I decided to simplify my 'system' to a number of around 10 possible inputs (as I will explain later in section 3. Game: Puppet).

2. User input prediction

2.1 Description

As I specified before, the final outcome of this project is to produce an AI system that can 'read' inputs from a player and predict what is going to be the next input that he will produce. Then, there will be a reactive agent (not a deliberative with a developed plan but an agent that will react depending on the inputs and the state that he finds himself).

For the 'predictive' system, I have done some research regarding n-grams, Markov models and statistical models for user input prediction. In the next sections I will explain briefly each one of the studied approaches.

2.2 n-Gram

The n-Gram approach offers a simplistic and easy approach to the user input prediction. This technique is based on the idea that humans like to repeat actions. Therefore, this system tries to predict the next input of the player by looking at the previous actions that he has done.

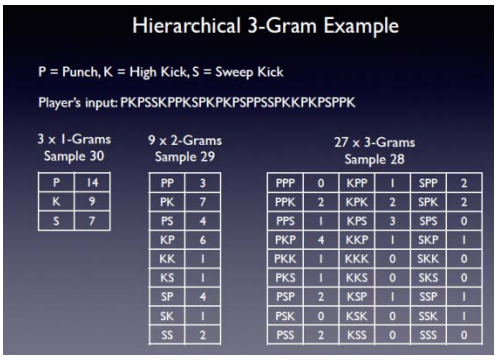

Figure 1 - Example of Hierarchical 3-Gram. (Jeremy Gow, AI in Games slides)

To solve this problem more efficiently the technique used is the hierarchic n-grams. To do so we have a set of n-grams of different dimension (1-gram, 2-gram... n-gram) and we try to check if he have enough information to predict the next input, given the previous n-1 inputs. If it's not enough information in the n-gram information, we check the n-1, n-2... until we arrive to the 1-gram, and we obtain the position with the bigger number of repetitions.

I will go in more detail further in the report, explaining how I particularly have implemented this solution.

2.3 Markov models

The basic idea of a Markov model (Cambridge University Press, 2004) is to represent the possible sequence of inputs as you can see in the figure 2. In a Markov chain each input is represented as a 'state', and the transitions between different states happen following a given probability.

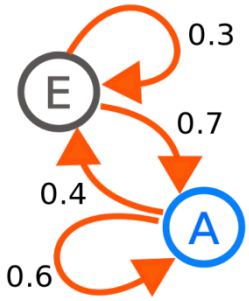

Figure 2 - Marov model representation (Wikipedia/Markov chain)

Following this kind of models, having a complex but accurate representation of the possible occurrences of the different inputs, we could obtain a probabilistic approach to the prediction.

I would have to create a representation of the sequences by a Markov chain. Even though that would have been an interesting comparison (between nGrams and Markov models) due to the short time to implement this project (because of investing time researching on MCTS) I opted to ignore this possibility.

This will remain in the section of further work expanding this project.

2.4 Other solutions

In the short time that I available to spend in research for this project, I looked for different samples in real-world applications, to check what techniques are widely used. I research as well on different papers and I focused my research on Google publications on User Input Prediction. I checked different sources to try to figure out how the 'autocomplete suggestions' is implemented (Sullivan, 2011), (Fishkin, 2012) and even some interesting articles on some problems with its 'autocomplete' feature (Mahdawi, 2013).

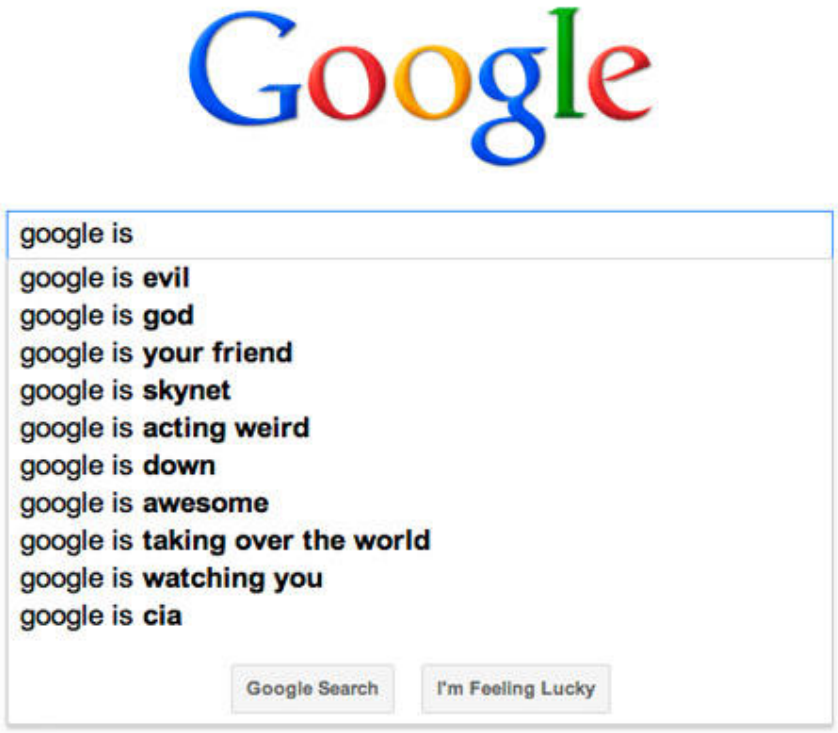

Figure 3 - Funny example of a Google 'prediction'

One interesting approach (Hirokuni Maeta, 2012) that I studied how to implement was a statistical approach. In that paper, they suggest the utilization of statistical way of store the pronunciation of Japanese characters with a n-Gram model. Doing so they pretend to improve the language model of Kana-Kanji Conversion by the utilization of n-Gram models to produce predictions.

The potential of these prediction techniques can be applied to other problems, for instance I found and read an interesting papers where they explain how to produce a 'next word prediction' based on an N-Gram Language Model (Harb, 2014).

3. Game: Puppet

3.1 Game idea

Motivated as the previous research, I wanted to develop a small demo that would show the power of n-Gram prediction. To do so I thought about some memorization game, where the computer should be the one trying to memorize what the user is doing. After designing some concepts on papers I decided to go for a small 'fighting' game, where the player can do a few set of attacks and the computer has to predict from them.

I will spend some words on the design of the game-demo to show n-Gram in action.

Indiana Jones and the Fate of the Atlantis

The 'Puppet Fight AI' game has been "inspired" in the actual battle system of the old classic graphic adventures of Indiana Jones (Lucas Arts TM) in concrete of 'Indiana Jones and The Fate of The Atlantis'. During this game (and its predecessor 'Indiana Jones and the Last Crusade'), Indiana has to solve an innumerable set of riddles but, sometimes, it can solve problems using his fists instead of his brains. In such scenes you control 'Indi' by a simple set of movements: LEFT, RIGHT, HIGH PUNCH, MID PUNCH, LOW PUNCH, HIGH BLOCK, MID BLOCK, LOW BLOCK. For the 'Puppet Fight AI' I wanted to follow that simple example, being able to produce a simpler version of that game.

Figure 4 - Screenshots from Indiana Jones and The Fate of the Atlantis

In the original game, Indiana and its opponents have a 'power' bar and a 'health' bar. The first version of my fighting game was made only with the life bar.

Figure 5 - Main screen of my demo

Puppet Fight AI[1]

I wanted to do a 'fighting game' but I wanted to try to do something that I could show around without nobody feeling that my demo was 'violent' or 'aggressive'. After thinking a while I thought of making a game with 'puppets' fighting. That would even simplify more my work, by letting me produce some good assets without having to worry to much on the animation of the legs.

3.2 Evolution of the game dynamics

The initial version of the game, as stated before, was a simpler version where there were only 8 actions and the life bar as 'interesting parameters' defining the game.

[1] To justify the inspiration of the 'theme' of the game, please, feel free to check the sketch 'The Flat' by The Umbilical Brothers.

After throwing some simulations with the AI, I've realized that after a while the agents will stop blocking. Actually, it was really easy to realize that 'blocking' was completely pointless. The only way to 'win' is to hitting the opponent. If the AI predicts that it's going to be hit by the enemy, the best thing that he can do is not try to 'guess' the position where to block, but the 'cleverest' thing to do is to attack as well.

At this moment I came to realize why the original game has the 'power' bar, and therefore I decided to include the feature of 'energy' bar (power, stamina...). With this feature, the damage that a player would do to the enemy, would be multiplied by the percentage of energy that he has. Each punch will reduce the energy. The energy will be reloaded with time, but blocking will increase slightly the ratio of reloading.

Therefore the reaction of the agent is richer and depends purely on the prediction given by the n-Gram, but the 'puppet AI' takes into account his own status while taking decisions.

3.3 Reactive AI Agent

After explaining the main concepts of the game, I will explain the different AI implemented to show the capabilities of the n-gram visualization.

Random

The first of the AI used to test the key inputs was a 'random' AI. This is a really basic AI that was used to test the prediction of the n-gram system. Interestingly enough, after checking the n-gram produced by the n-gram prediction fighting against a random AI, those were more or less evenly distributed after a very long time.

This Random was a really interesting AI to implement as it let me experience how the n-gram are powerful as long as the player against whom the n-gram is taking inputs is human or is following a given set of rules.

Predictive: mimic and block

Mimic and block AIs where implemented just to show a display of the n-gram prediction capabilities. These two AIs react fairly similarly to the prediction. The mimic one just tries to imitate what the player will do, and the block AI will just try to block always (will never counterattack the player).

These two AIs are powerful to show the player the interesting of the n-gram prediction.

Predictive: Utility System AI

After adding the 'energy' bar, the system was complex enough to facilitate the implementation of a further AI technique in this project. The basic technique demoed is the n-gram prediction, but, given that I was creating a game and I wanted the AI to let the user 'play' a game, I wanted to add an interesting enough AI.

This AI takes into account different factors, such as energy, life and position with the opponent to react accordingly (and given the predicted input).

The first discriminatory element to the reaction of the Agent is the predicted input. After that input is estimated, the AI will take into account the distance with the player, and then, probabilistically, it will combine the amount of life and energy, and will obtain a random value, to take decisions regarding 'rest' or 'punch', 'attack' or 'retreat. Doing so, the AI looks clever enough and it can be a real challenge to the player.

4. N-gram

4.1 N-gram prediction

To implement the n-gram prediction I've designed a module that can be attached to any AI system. This module is just an AIPrediction with a Sequence class and an AI Prediction class.

Any other C++ project could benefit of the n-gram prediction. There would be only the need to add the inputs obtained from any device into the AIPrediction class, and then ask for a prediction. Inside the class the number of actions and dimensions of the n-gram and threshold are easy to modify.

Figure 6 - Screenshot of the AI in action

4.2 Data structures

For the first assignment of this master we had three different options, develop a game with octet framework, develop a demo with octet framework or develop an importer for the open Game Exchange Game format for octet. As I have already done some projects with openGL and several C++ programing projects, I decided to design and implement the ope

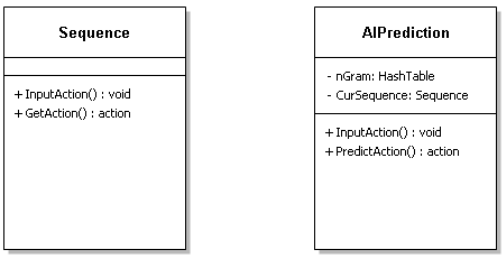

Figure 7 - Data Structures of the AI Prediction module

4.3 Some graphs

This first graph offers the representation of the 1 gram values along time on a single player match.

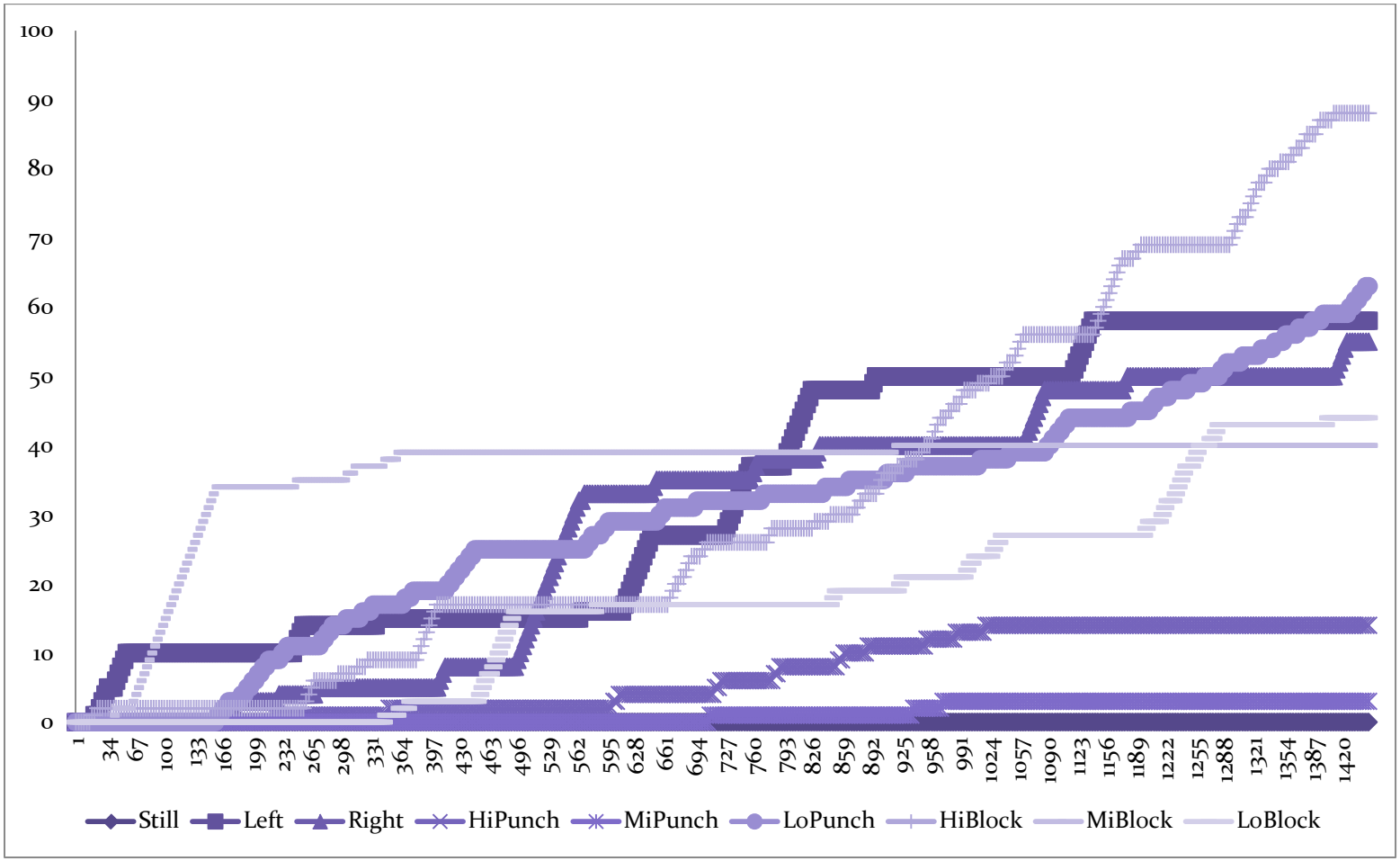

Figure 8 - Graph showing the evolution of a 1-gram along time

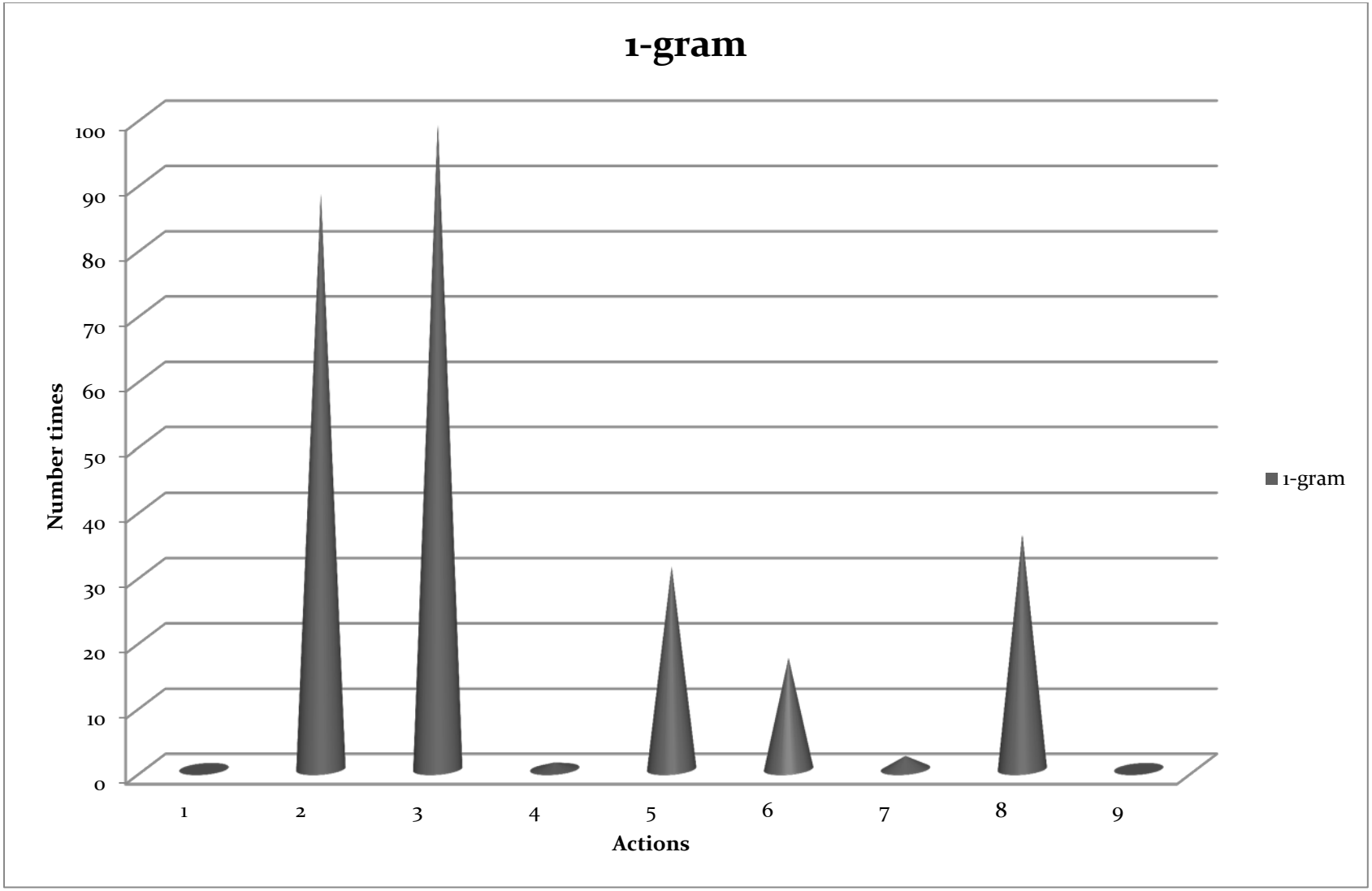

Figure 9 - 1- gram graph after a whole combat AI vs AI

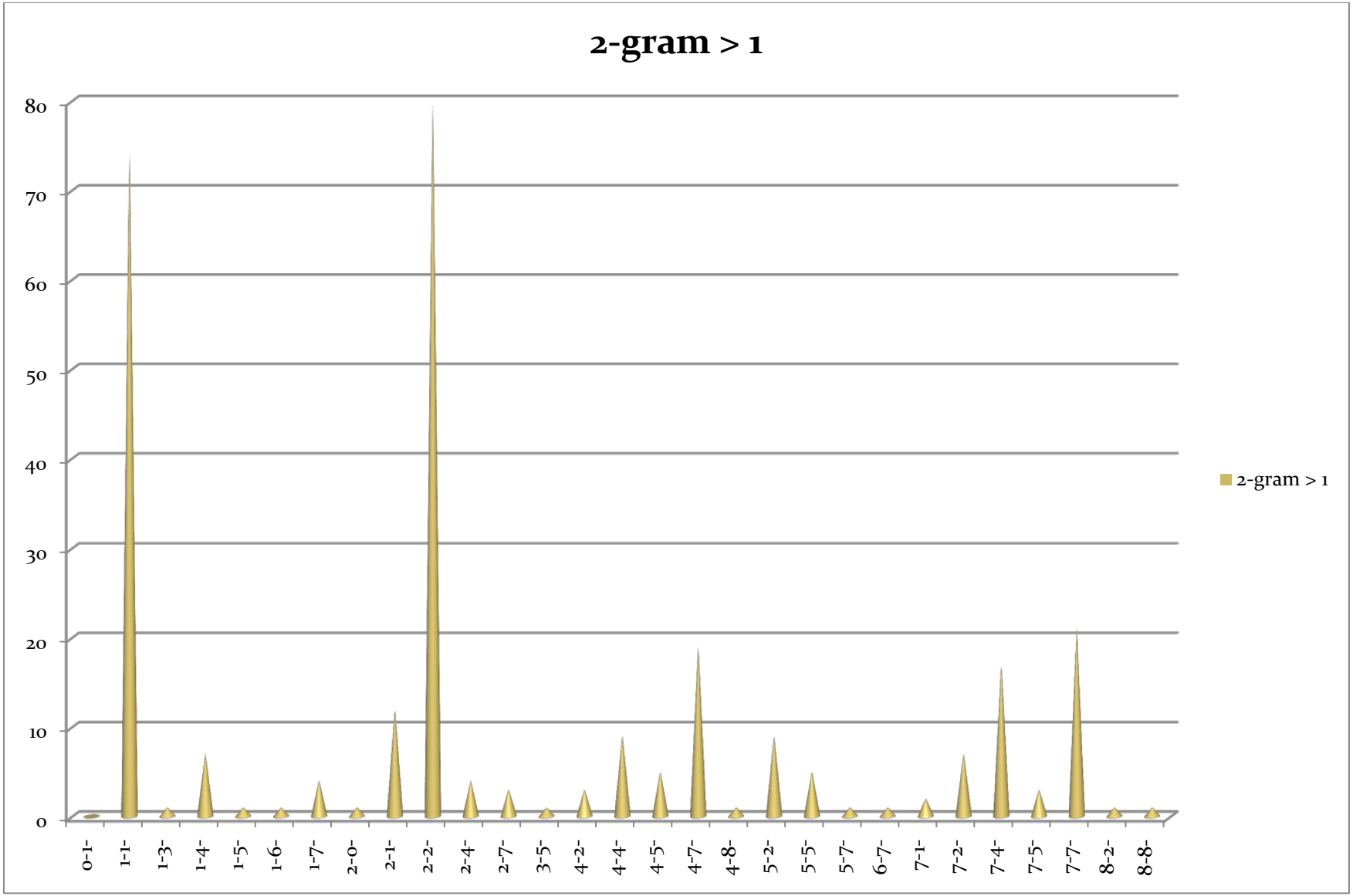

Figure 10 - 2-gram of all values bigger than 1

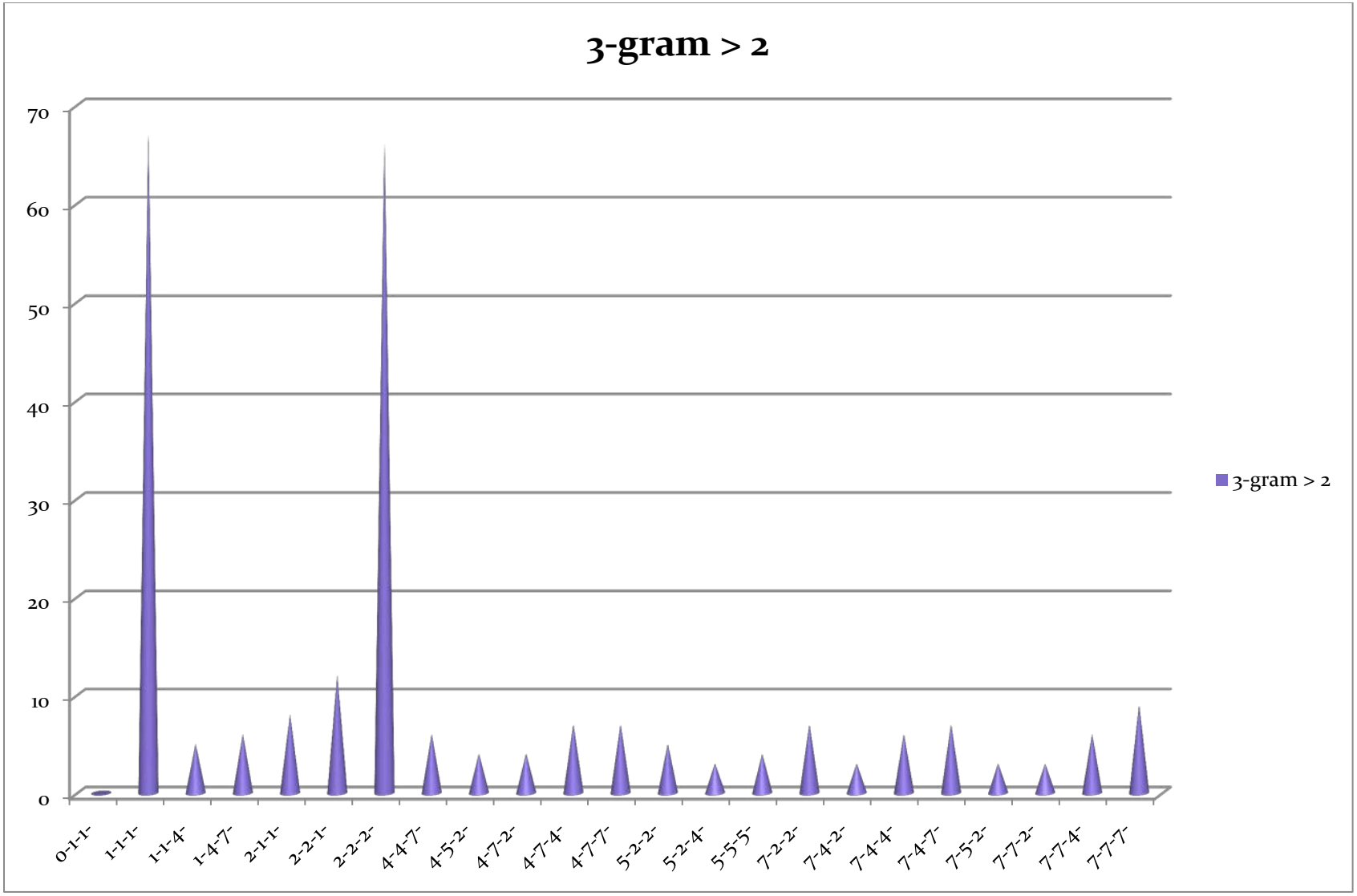

Figure 11 - 3 gram of all values bigger than 2

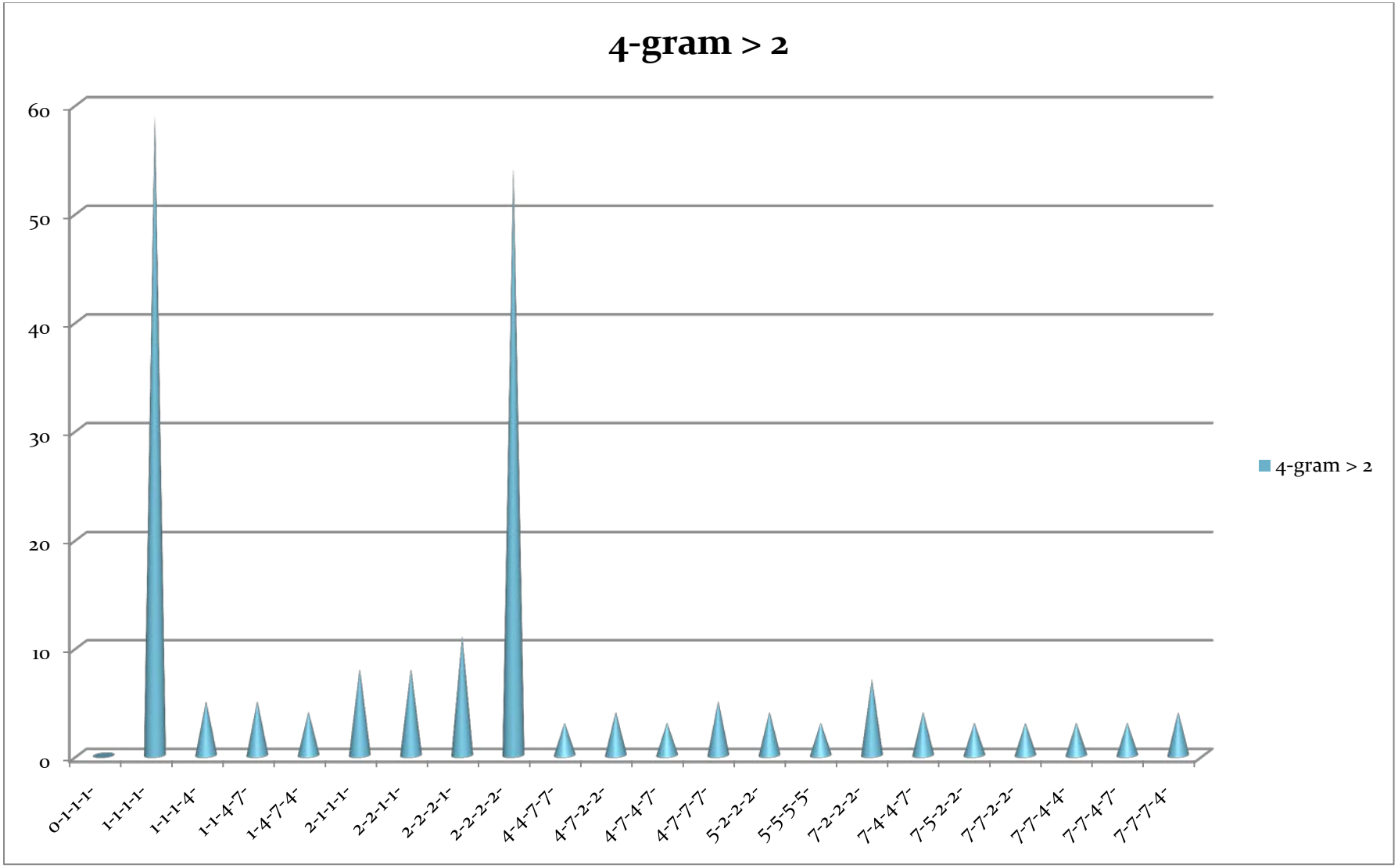

Figure 12 - 4 gram of all values bigger than 2

5. Conclusions

5.1 Conclusions

This assignment has given me the possibility of research in a big set of different techniques. The initial research in MCTS was really productive. After figuring out a system to model the world, I realized that I would have to implement a big set of different elements (or use third-party libraries).

After thinking seriously and realizing that the time wouldn't be enough to implementing everything from scratch I thought of changing the topic of the assignment, to produce an easier demo where the amount of work to produce a 'game' or 'demo' as a container to an interesting AI technique.

I have learned on being more 'reactive' and less 'planner' (not only for the AI technique chosen) but as to react to the reduce amount of time and changing topic soon enough.

On the other hand this swaping between projects has given me the chance to make even a wider research for this assignment.

Interesting enough I've thought about using Utility System as a way to model the AI before even knowing that was a wide used technique in AI.

5.2 Further work

The further work that I intend to develop for this assignment is a network module that will let users to play this game through html and the possibility to choose among other prediction techniques (such as Markov Models) or adding different kind of reactions for the AI (removing mimic and changing that slot for other technique). Another easy improvement for this project would be to re-do the user interface in a cleverer and more aesthetic way.

6. Readme

6.1 How to install it

To try this demo it's needed a MS Visual Studio to compile the code. The code can be obtained from the my personal git repository:

git clone https://github.com/Juanmihd/PuppetFightPrediction .

After pulling the repository, there is only needed to compile the project (src/examples/MiniFightAI).

Just execute it from the bin folder, and enjoy the play (or feel free to check this video with some gameplay!)!

6.2 How to use it

6.2.1 Keyboard

The key to play as player are:

Player 1 (left – bear)

A – left

D – right (or punch if close to other player)

W, S, X – High, Mid and Low block

E, C – High, low punch

Player 2 (right – lion)

J – right

G – left (or punch if close to other player)

Y, H, N – High, Mid and Low block

T, B – High, low punch

Bibliography

Association for the Advancement of Artificial Intelligence. (2014). *High-level Representations for Game-Tree Search in RTS Games.* Association for the Advancement of Artificial Intelligence.

Cambridge University Press. (2004, September). *Markov Chains*. Retrieved from University of Cambridge: http://www.statslab.cam.ac.uk/~james/Markov/

Cameron Browne, D. W. (2012, March 1). A Survey of Monte Carlo Tree Search Methods. *IEEE Transaction on Computational Intelligence and AI in games*, pp. 1-49.

Fishkin, R. (2012, March 12). *How Google's "Search Suggest" (Instant) Works*. Retrieved 2015, from The Moz Blog: http://moz.com/blog/how-googles-search-suggest-instant-works-whiteboard-friday

Harb, M. (2014, December 7). *Next word prediction Based on an N-gram Language Model*. Retrieved from Presentation for the Coursera/JHU Data Science capstone project: https://rstudio-pubs-static.s3.amazonaws.com/46938_197b3fac2d2647a2867c75dc405ab9dc.html#/

Hirokuni Maeta, S. M. (2012). Statistical Input Method based on a Phrase Class n-gram Model. *Proceedings of the Second Workshop on Advances in Text Input Methods*, 1-14.

Mahdawi, A. (2013). *Goggle's autocomplete spells out our darkest thoughts*. Retrieved from The Guardian: http://www.theguardian.com/commentisfree/2013/oct/22/google-autocomplete-un-women-ad-discrimination-algorithms

Soemers, D. (2014, June 16). Tactical Planning Using MCTS in the Game of StarCraft.

Sullivan, D. (2011, April 6). *How Google Instant's Autocomplete Suggestions Work*. Retrieved 2015, from Search Engine Land: http://searchengineland.com/how-google-instant-autocomplete-suggestions-work-62592