

Learning Dynamic Stereotypes for Effective Autonomous Agents

By

W. Joseph MacInnes

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy

at

Dalhousie University
Halifax, Nova Scotia
December 2003

© Copyright by W. Joseph MacInnes, 2003



National Library
of Canada

Bibliothèque nationale
du Canada

Acquisitions and
Bibliographic Services

Acquisitions et
services bibliographiques

395 Wellington Street
Ottawa ON K1A 0N4
Canada

395, rue Wellington
Ottawa ON K1A 0N4
Canada

Your file Votre référence

ISBN: 0-612-89810-5

Our file Notre référence

ISBN: 0-612-89810-5

The author has granted a non-exclusive licence allowing the National Library of Canada to reproduce, loan, distribute or sell copies of this thesis in microform, paper or electronic formats.

L'auteur a accordé une licence non exclusive permettant à la Bibliothèque nationale du Canada de reproduire, prêter, distribuer ou vendre des copies de cette thèse sous la forme de microfiche/film, de reproduction sur papier ou sur format électronique.

The author retains ownership of the copyright in this thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without the author's permission.

L'auteur conserve la propriété du droit d'auteur qui protège cette thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.

In compliance with the Canadian Privacy Act some supporting forms may have been removed from this dissertation.

Conformément à la loi canadienne sur la protection de la vie privée, quelques formulaires secondaires ont été enlevés de ce manuscrit.

While these forms may be included in the document page count, their removal does not represent any loss of content from the dissertation.

Bien que ces formulaires aient inclus dans la pagination, il n'y aura aucun contenu manquant.

Canada

DALHOUSIE UNIVERSITY
INTERDISCIPLINARY STUDIES

The undersigned hereby certify that they have read and recommend to the Faculty of Graduate Studies for acceptance a thesis entitled "Learning Dynamic Stereotypes for Effective Autonomous Agents" by W. Joseph MacInnes in partial fulfillment of the requirements for the degree of Doctor of Philosophy.

Dated: December 19, 2003

External Examiner:

Research Supervisors:

Examining Committee:

Departmental Representative:

DALHOUSIE UNIVERSITY

DATE: Dec 19 2003


AUTHOR: W. Joseph MacInnes

TITLE: Learning Dynamic Stereotypes for Effective Autonomous Agents

DEPARTMENT OR SCHOOL: Interdisciplinary

DEGREE: Ph.D. CONVOCATION: May YEAR: 2004

Permission is herewith granted to Dalhousie University to circulate and to have copied for non-commercial purposes, at its discretion, the above title upon the requests of individuals or institutions.


Signature of Author

The author reserves other publication rights, and neither the thesis nor extensive extracts from it may be printed or otherwise reproduced without the authors written permission.

The author attests that permission has been obtained for the use of copyrighted material appearing in the thesis (other than brief excerpts requiring only proper acknowledgment in scholarly writing), and that such use is clearly acknowledged.

Table of Contents

List of Figures	vi
List of Tables	vii
Abstract	viii
List of Abbreviations	ix
Acknowledgments	x
Chapter 1: Introduction	1
Chapter 2: Background	9
User Modeling	9
Stereotypes	10
Machine learning for user modeling	15
Multi agent systems	17
Adversarial Modeling	19
Adversarial problem solving	20
Gender and spatial ability	24
Chapter 3: Proposal	28
Dependent Variables	34
Independent Variables	36
<i>Within Subject</i>	36
<i>Between Subject</i>	37
Summary and Predictions	38
<i>Software</i>	38
<i>Human</i>	39
Chapter 4: Methods	41
Arena	41
Agents	45
<i>Deterministic Finite-State Automata(DFA)</i>	48
<i>Neural Network</i>	51
<i>Human</i>	55
<i>Transfer Control Protocol (TCP)</i>	56
<i>Mixture of Experts</i>	58
Recursive Modeling	62
Experiment Design	65

Chapter 5: Results and Discussion	71
Overview and Descriptive Statistics	71
Accuracy	72
Effectiveness Score	83
<i>Experiment 1</i>	83
<i>Software perspective</i>	89
<i>Human perspective</i>	91
<i>Experiment 2</i>	95
Believability Rating	99
<i>Experiment 1</i>	99
<i>Experiment 2</i>	105
Combined Performance	106
Chapter 6: Conclusion	110
Experiments	110
<i>Software Perspective</i>	110
<i>Human Perspective</i>	113
Contributions	115
Future Work	116
Appendix A - UML class diagram	118
Appendix B Screen Shots	119
Appendix C Sample code	123
Appendix D Data transformation/training pseudo-code	124
Appendix E Agent Summary	125
Appendix F Definitions	127
References	128

List of Figures

Figure 1	Comparison of MOE theory and implementation	32
Figure 2	Overhead of Hebb/Williams maze	43
Figure 3	Multi agent interaction	47
Figure 4	DFA state diagram	49
Figure 5	Neural Net structure	52
Figure 6	Prediction triangulation	54
Figure 7	Standard deviations of cluster/stereotypes	72
Figure 8	ANOVA for accuracy	76
Figure 9	Accuracies of algorithms with gender effect	77
Figure 10	Accuracy data split by cluster	78
Figure 11	Accuracy with experience and gender interaction	80
Figure 12	Histogram of experience	81
Figure 13	Histogram of game completion time	85
Figure 14	Effectiveness with time by gender interaction	86
Figure 15	Histogram of completion time by gender	87
Figure 16	Effectiveness for ten algorithms	88
Figure 17	Four effectiveness diagrams with recursion effects	89
Figure 18	Effectiveness for algorithm and gender	91
Figure 19	Gender by experience interaction	92
Figure 20	Effectiveness vs Believability scatter plot	94
Figure 21	E2 Effectiveness with order and time	97
Figure 22	E2 Effectiveness with algorithm and order of match	97
Figure 23	E2 Effectiveness order of match and recursion	98
Figure 24	Effect of knowing and prior gaming with opponent	101
Figure 25	Believability of algorithms	103
Figure 26	Believability, recursion and prior LAN	103
Figure 27	Believability, recursion and experience difference	104
Figure 28	Believability in experiment two across match	105
Figure 29	Various combined performance results	106
Figure 30	Combined performance by experience	107
Figure 31	Recursion, algorithm and network experience	108

List of Tables

Table 1	Neural net results	61
Table 2	Division of cluster data	79

Abstract

This research looks at novel combinations of machine learning techniques and models of human performance to create software agents that were both efficient and believable. It is the proposal of this dissertation that by using machine learning techniques that mimic cognitive theories of game playing, opponent modeling and deception, that improvements can be made with the performance of software agents. In addition, human traits will be examined as between-subject variables to explore individual differences in this spatial environment.

Experiments were run pitting various combinations of human and software agents against each other in duels within the virtual arena. Three different intelligent software agents, each using three levels of recursive modeling were tested against human participants with three dependent variables. Each algorithm was tested for base accuracy in predicting its opponent's position, its effectiveness in fighting a human opponent and, its believability in portraying a human opponent (this last measure can be seen as a limited scope 'Turing Test').

It is interesting to note that the algorithm that performed the best on the accuracy measure was not able to translate that result into effectiveness. The algorithm that modeled human stereotyping and deception, however, not only performed far better on the measure of believability, but on effectiveness as well. Due to the lack of a direct link between the three measures, researchers and programmers should consider their objectives carefully before choosing an algorithm for their agents. In addition to these software results, it was shown that of the individual differences observed in this study, experience in spatial computer games played a larger role in performance than the gender of the participant.

List of Abbreviations

MOE	Mixture of Experts
DFA	Deterministic Finite-state Automaton
ANOVA	Analysis of Variance
AI	Artificial Intelligence
ML	Machine Learning

Acknowledgments

To Ray Klein, for creating an oasis of true learning within your lab. You showed me what it means to be a scientist in every sense of the word. I will always remember the friendship and trust you offered by letting a programmer in on your research. You will continue to be a very big influence in my life and career for many years to come. To Norm Scrimger, for being a great co-supervisor, and for making me sweat the details when I didn't want to. Sometimes the advice you appreciate most is the advice you disagree with. (And I take it all back if you correct the grammar in this acknowledgment page...).

Many thanks to the rest of my committee, Evangelos Milios, Peter Gregson, and Paddy McMullen. To Evangelos, thanks for being on my committee, for great advice, and for going out of your way to make sure I always felt like I had a home at Computer Science. To Paddy for the advice, understanding, and for giving me a home base in Psychology. Peter, I am grateful for the perspective that you brought to my degree. Mike Jenkins for agreeing to come to Halifax to be my External during a very busy time of the year. You've given me some excellent questions to ponder in my future research. To Fay Cohen and Carolyn Watters, I couldn't have picked two better associate deans. You both made my years as an interdisciplinary student enjoyable.

Special thanks to Irina, for being part of my work, my school and my life. I think you are the only person who read my thesis that didn't have to. I am looking forward to reading yours. Brad, your thorough code testing saved me embarrassment on more than one occasion. It meant even more considering how much you hate computer games. Tom (The Bhudda), for always pointing out the obvious, especially when I couldn't see it. (And for reminding me it's the journey, not the destination).

Tracy for advice, friendship, advice, and the occasional grad house beer when I really needed it (and did I mention advice?). You paved the way by showing that it could be done (and occasionally even how to do it...) Mary Ellen for being an awesome office mate, and for reminding me to stick to what's important both in and out of school. And Martin for giving me such a cool nick name (Joey Mac in case anyone wants to start using it...). Save me a seat in Belize if you get there first. Elizabeth for helping me when I really needed it. To Patti, for occasionally having to organize yet another absent minded academic. To Susan for helping me run off a little steam. Billy Schmidt and John Christie for paving the way for the computer Science invasion to Ray's lab. Billy, your friendship and ethical standards were an inspiration throughout my degree. Kori for swapping stats help for clustering advice.

To Natalie, Ron and Terry; the friends of my youth who stuck with me. Thank you for pretending not to be surprised by what I've been able to accomplish.

Special thanks to my mother, Ruth, and in memory of my father, Ron. They believed in me when I was still skipping high school. To my Grandmother Ruth whose warm heart and generosity has made her 'Nanny Ruth' to an entire lifetime of friends. In memory of my Grandmother Muriel, I hope I still have half the wit and intelligence at 45 that she had at 90. In memory of my Grandfather Bill, who always said I would either be a millionaire or in jail by the time I was thirty. I'm still not sure which would have made him prouder.

Last but not least, special thanks to Lynn.

"Don't let school get in the way of your education", Mark Twain.

Chapter 1: Introduction

Research in Human-Computer Interaction (HCI) is unique in that it offers insight from two different perspectives. Not only does the software provide information about human traits, but the human participants can shed light on the software algorithms. The goal of this dissertation is to implement and test a variety of opponent modeling solutions in a novel multi-agent environment. It is hypothesized that implementations which consider human modeling and deception will hold a distinct advantage over other implementations, and that different factors may influence performance depending on what criteria is used to measure the success of an algorithm. The software agents will also be used to model human traits during these interactive duels. In particular, the more advanced agents will be trained with regard to the current spatial task and recent literature on gender and experience. Both human and software perspectives will be tested by observing duels between a variety of human and software combatants in a virtual arena.

Traditional Human Computer Interaction (HCI) consists, in generic terms, of a command by the human user, a response to that command by the software on a computer, and possibly some sort of feedback to show the user the results of the action. For consistency, the user will hereafter be referred to as a human agent and the software within which any agent resides, the environment. An agent (or autonomous agent) is any entity that is capable of *instigating* an action in the environment and perceiving the results of that action. This process of agent interaction would repeat indefinitely until the program was shut down or the agent ceased entering commands.

Earliest computing machines relied on punch cards for this user input, but such input was eventually replaced by keyboards and command line interfaces. With research into interface tools

such as the desktop and the mouse, graphic user interfaces allowed for more intuitive control of the environment. Although environments have progressed a great deal in the way that they interact with the agent, they still follow the basic flow of agent input, response and feedback.

Other changes in computing environments influenced the number of agents that could interact in a single environment. The earliest punch card systems, of course, were restricted to a single user and a single task at a time. Early command line and even GUI's started off the same way, but eventually progressed so that multiple users could perform multiple tasks at the same time.

The goal, however, was to maintain the illusion that any given human agent was the sole operator of a single environment whose focus was directed entirely by that agent. The usual scenario includes creating multiple, virtual desktops for each of the human agents to control. Any shared resources between these desktops are kept hidden from the users. It could be said that multiple programs, processes and threads could easily be interpreted as agents in these environments. Even though these multiple processes bore a strong relation to multiple autonomous agents in practice, they were usually deemed successful if this nature was transparent to the user. Another key difference is that these threads, while existing autonomously, were initiated (ultimately) by a single human user.

Multi-agent environments differ from typical interaction in a number of significant ways. Two or more agents inhabit these environments, and both have the ability to independently interact (inhabit, modify, perceive) with the environment at essentially the same time. The independence is a key criterion, suggesting that neither agent has complete control of the environment since one or more agents are changing that environment simultaneously. Given these conditions it is impossible for an agent to completely predict the full results of its own actions, let alone those of

another agent.

Although some environments may allow for many agents with differing levels of control over the environment, this research will be limited to such environments where all agents have equal ability to change and observe the environment. There is no direct control over the environment, since multiple agents may be attempting to change and interact with this environment, *independent* of the goals and actions of the other agents. Secondly, multi-agent environments allow for interaction between both human and software agents. Although a software agent ultimately had its origins with a human programmer, it is still defined by its ability to interact with the environment, independent of other agents. It has its own goals and objectives, and the same ability to sense the environment and act upon it.

It is important to note at this point that not all environments have a true separation of agent and environment as is suggested here. Programs can 'cheat' by allowing software agents access to information that human agents could not know, and allow actions that are not possible. Some environments do keep all agents on an equal footing, while for some it is only important that they appear to do so. This research focuses on environments which treats all agents equally. This not only offers a true comparison, but it also provides for a more modular, replaceable nature of the agents. Although the process by which each agent arrives at its actions will clearly be different, they are identical from the perspective of the environment.

Examples of this type of multi-agent system do exist throughout the history of computers, but usually in very specialized systems where such interactions are common. Very early computer games, such as Pong, allow for many combinations of both human and computer agents interacting

with little concern for which is which. Both are capable of playing virtual ping pong, and both are capable of simultaneously observing and effecting the state of the virtual table. The state of the environment is not defined by the actions of either agent, but by a combination of both.

Large scale multi-agent domains first started appearing in the early days of the internet. Multi-User Domains (MUDs) achieved a cult status among many when the internet was still text based. Users would create agents with personalities to interact with other agents on-line. This tradition continues today in the form of Massively Multiplayer Online Role Playing Games (MMORPGs) that add virtual avatars and a healthy mix of human and software agents.

Although it is not the focus of this research, it is not hard to imagine a different theme for the common desktop approach to traditional computing. An environment with obvious, autonomous agents could transform the user into guiding the interface rather than controlling it. New autonomous agents could be created to perform tasks, and report once completed. Current research has already begun on shared desktops that allow multiple human agents simultaneous control over the same desktop environment.

Software agents become critical in all of these environments to maintain a minimum level of interest and functionality. They can provide a way to communicate to the software environment without destroying the feel of the virtual world (reporting a rules violation in an on-line 'western' to a software agent sheriff). Software agents are often provided as opponents in competitive multiplayer games. They can also provide a base level of interaction when human agents are in short supply.

Although game-like environments are the most popular multi-agent environments, they are

not the only ones. Training simulations often require the trainee to interact with others in the learning process. Simulated operating rooms require software patients that react like the real thing and combat flight simulators require computer opponents to battle. On-line trading environments where software agents are being used to improve on human stock predictions are also being researched.

The world wide web itself can actually be seen as a very large scale multi-agent environment. Human and software agents already browse, search and modify the web on a daily basis. Since a multitude of agents are perceiving and affecting the web simultaneously, it is in a constant state of flux and very hard to predict. The web contains examples of collaborative interactions (a software search agent guiding a human agent) as well as numerous types of competitive interactions (a virus agent trying to 'fool' an anti-virus program).

The contributions of this dissertation are as interdisciplinary as its origins. The tools developed to make this research possible are no small part of what this dissertation offers to the field. 'OpenRT', the graphic engine developed to test the theories put forward is a fully functional, open source environment for conducting cognitive research. Many options and modules were developed and made available for this tool including two and three-dimensional environments, real time components, basic physics, object loading, space partitioning, Fourier transforms, and cross-platform and multiplayer capability. Although all of these modules were not used in the experiments discussed in this dissertation, they are available components of the overall project. Experiments in this dissertation make use of the three-dimensional engine, basic physics modeling and multiplayer support. Although real time support was not added for various reasons, the timing

capabilities proved more than sufficient.

The other significant tool set developed for this research was a collection of Artificial Intelligence (AI) and Machine Learning (ML) techniques used to control the autonomous, software agents in the multiplayer game. Although not all techniques were novel in and of themselves, combinations in which they were used demonstrate interesting possibilities for many of these algorithms. The Mixture Of Experts (MOE) solution in particular goes through a number of stages that take it from raw observational data to a very effective agent's 'brain'. This process makes use of unsupervised clustering algorithms to partition the data. These partitions were then used to create the specialized experts used in the MOE. The neural network solution as well as the smaller sub-experts are basic feed-forward/back-propagation neural nets whose input layers have been modified to help interpret the temporal component of the opponent modeling. This dissertation also demonstrates that these particular algorithms were not arbitrarily chosen. Cognitive theories on game playing and success in opponent modeling among humans were researched and aided in the choice of machine learning solutions. Literature on stereotypes, deception and recursive modeling have all played a role in the development of these solutions. The MOE in particular was chosen as the more advanced modeling technique due to its high degree of overlap with the processes that are believed to be important in human opposition modeling.

Finally, but certainly not least, this research uses the tools described above to get a better appreciation of the factors that are important in certain types of multi-agent environments. A variety of human participants were tested and traits were recorded for each one. The aforementioned algorithms (Neural network, MOE) as well as a simple Deterministic Finite State

Automaton (DFA) were used as the brain for software agents competing against human agents in game-like three-dimensional duelling arenas. Performance against all of these algorithms was measured along with the recursive depth of the modeling. In addition to these controls, a number of between-subject, independent variables were observed to determine their influence in these multi-agent environments.

A significant contribution actually lies in the dependent variables chosen to measure performance of the different algorithms and recursive levels. The first two measures were chosen both as a tie-in to current literature and for their usefulness in many agent environments. The *accuracy* of the algorithm, measured as the simple distance from the opponent's predicted location provides a straightforward measure of an agent's capabilities. The *effectiveness* of an agent, measured by its ability to defeat its opponent, is arguably a more meaningful measure, but also more difficult to determine the cause of any success since it is influenced by factors outside of the agent's control. Finally the *believability* of an agent is a dependent variable that is rarely measured, but very important in a number of situations. This research measures all three variables since it may be just as desirable for a software agent to mimic a human's performance as it is to exceed it.

Not only will this research look at these performance measures with regard to all of the variables mentioned above, but also to the relationships between the measures themselves. There may be situations for which all three measures cannot be optimized simultaneously and in fact, may even be conflicting. It is not hard to imagine a scenario where the most effective solution also displays the least human-like behavior and vice-versa. Since human beings are not always the most

efficient, a perfect algorithm will not capture the natural variance inherent in human behaviour.

Chapter 2: Background

This dissertation calls on theory from a variety of disciplines within computer science and cognitive psychology including user modeling, machine learning, agents, multi-agent environments, machine learning for user modeling, opponent modeling and recursive modeling. An overview of relevant literature from each of these areas will be provided, followed by a summary and proposal of how they will be tied together in this dissertation.

User Modeling

Applications of user modeling usually fall under one of three categories. User interfaces (especially adaptive or intelligent user interfaces) attempt to model the user in order to improve the ease and/or effectiveness of the (human/computer) interaction. Data mining also benefits from modeling since an accurate model of the user's behaviour and preferences can significantly reduce the scope of searches. Finally, multi-agent systems will benefit from successfully modeling human agents with whom they interact. Collaborative systems could cooperate more effectively by modeling their team-members and competitive environments would benefit by correctly anticipating their opponent's actions. It is in this final scenario - multi-agent, competitive environments - that this dissertation proposes to develop and test dynamic stereotypes. Specifically, modeling performance will be tested using competitive multi-agent games, building on the environment and work from MacInnes (2001).

While many collaborative modeling projects make extensive use of user questionnaires as a base for the user model, competitive environments offer an additional difficulty/challenge by not

being able to get assistance from the user being modelled. In fact, the user may hide or masquerade their behaviour. Also cooperative modelers may suggest options to the user that may or may not be taken; the competitive modeller has only its own recommendations with which to make its decision. The modeller, in effect, needs to achieve the same (or higher) accuracy with less information.

Stereotypes

One of the pioneer works in the user modeling literature and definitely one of the most cited is Rich's paper on using stereotypes for user modeling (Rich, 1979). Although stereotypes had been used prior to 1979, this prior work focussed on stereotypes as they applied to human cognitive understanding. Rich, however, suggested that the usefulness of stereotypes could be extended to models of computer users. Since the rationale behind stereotypes is that the world is far too complex to understand (and remember) without simplification and categorization of the details, the same benefits could be achieved by computers.

Rich implemented 'Grundy', a system that attempted to model users in order to predict the user's preference in books. User models were implemented as a series of 'facets' that represented individual facts about the user. Contained within each facet is a rating that shows the bias towards or against this attribute, a rating that reflects the certainty (probability) of this aspect, as well as the justification for the belief in this aspect. Therefore, the grouping "education;5;900;INTELLECTUAL" would represent a facet for a user that suggests an interest in educational material with a value of 5 (high) and a rating of 900 (also high) with the justification

being that the user activates the INTELLECTUAL stereotype. Stereotypes get activated in this system by one of a number of triggers. A trigger can be any event in which the user interacts with the system. Many triggers will be activated at the beginning of a session in which Grundy asks the user a series of personal questions. Triggers can also be activated later in the session during the system's normal interaction with the user, either confirming, adding or modifying assumptions that were made earlier.

Stereotypes also contained these facets, but of the 'typical' member of that group. Whenever a stereotype is triggered, the values of that stereotype are added to the user's profile. The value and rating of facets that coexist in the user's profile are adjusted according to whether they confirm or conflict with those found in the stereotype. All stereotypes were arranged using a Directed-Acyclic Graph (DAG) going from the most generic to specific. The DAG was implemented as apposed to a tree to allow for any child stereotype to be specifications of more than one parent (more generic) stereotype. This is important since activating a stereotype also activates all of its generalizations.

Although much has been said recently about the inflexibility of stereotypes in user modeling, Rich did make attempts to implement learning within the stereotypes. As new information is acquired about members of a given stereotype, the values and ratings of the facets may be adjusted to reflect that new information.

Although it was not implemented in Grundy, Rich even suggests ways to which new stereotypes may be learned using classification techniques in use at the time. Although no statistical analysis was done on the effectiveness of this system, Rich did demonstrate that the number of

successful book recommendations nearly doubled by using Grundy compared to random selection.

Machine learning

Machine learning is an area of computer science and a specialization of artificial intelligence (AI). While traditional AI tended to focus on systems that made extensive use of expert knowledge (expert systems), machine learning techniques attempt to focus on how the computer can solve problems with less need for expert guidance. In fact, machine learning algorithms are often classified by the extent to which they need human intervention. Although all machine learning algorithms learn by observing examples of the type of problems that they are to solve, some algorithms need extra information in the form of what those samples mean.

Supervised algorithms are those that need to be shown a solution (or result, or label) for every sample problem seen during training. The goal of these algorithms is to be able to determine the result for similar but unknown problems once the training is complete. For example, a supervised machine learning algorithm used to solve math equations would be shown a number of equations along with their solutions. The algorithm would learn from these, and then be able to predict the answer to previously unseen equations.

Unsupervised learning algorithms however, are used to divide training samples into similar groups without any need for labelling or supervision (Russel, 1998). These algorithms are often called clustering or mixture modeling algorithms. The algorithm is merely provided a distance measure that can be used to determine the degree of separation between samples, and it partitions or clusters the samples based on this measure. Unsupervised learning has no need of human input

to pre-label examples as one category or another as supervised learning does, but neither does it come up with such labels on its own. The result of the clustering is a collection of groups that are statistically similar, but it may be difficult for human analysis to determine *why*.

Reinforcement learning also tries to predict the result of an action in order to maximize the chances of achieving an outcome, reward, or utility. One difference between this and previous learning styles is in the inability of the reinforcement algorithm to determine the immediate reward of any single step. Chess is a good example, since the consequences of any single move of a game are unknown at the time of the move. Even when the consequences are known, (checkmate) it may not be clear which prior moves caused that particular consequence - each move contributes to the out come of a game, but very rarely decides it. This would be difficult for a supervised learning algorithm, since the results of every action are needed to train the algorithm. Reinforcement learning recognizes this difficulty and tries, through various means, to determine the connection between actions and longer term results.

The duelling arena used in these experiments is robust in that it combines potential examples of all three machine learning problems. Firing and accuracy could be handled by a supervised learning algorithm since the results of any action are clear and precise. Strategy and move selection could fall under reinforcement learning since the results of a choice at the current time may or may not effect the outcome of a match at a later time. The area of machine learning that relates most closely to the learning of stereotypes comes from the clustering literature (also called concept learning or mixture modeling). The clustering algorithm that will be the focus of this research allows not only clustering of data but novel arrangement and use of those clusters that will be shown to fit

with Rich's stereotypes.

Jacobs and Nowlan (1991) proposed that dividing a data set into clusters or subsets may be beneficial as initial processing for a learning algorithm. Using neural networks to cluster letters in a vowel discrimination task, the authors first divided the data using a single neural network ('gating network') into four subclasses of data, then used each of these subsets to train four other neural networks as experts ('expert networks') for their individual areas. The series of local experts were shown to have performed at par with a single multilayer neural net, but the local expert solution reached this performance with half of the number of training epochs.

Although standard neural networks are ideal for recognizing patterns, temporal pattern recognition (patterns that develop over time) has typically been seen as a problem for neural networks. Mozer et al. (1993) demonstrate a number of possible neural net architectures that are well suited for adapting to this type of problem. All of these models contain the typical features associated with a neural network, but in addition, they also incorporate some form of short term (implicit) memory from which temporal patterns may be detected. The two solutions for solving this problem that were considered for this research were recurrent networks and traditional feed-forward/back-propagation networks with discrete time stamps. Recurrent networks (such as an Elman network (Elman, 1990)), model temporal information by feeding some portion of the hidden layer back as input for the next time stamp. In this way, training samples from a previous time can influence the current training cycle. A second solution uses traditional feed-forward/back-propagation networks, but for each training sample feeds some discrete number of samples through as input simultaneously. Although these methods may seem similar, the recurrent net feeds pre-

processed information (hidden layer probabilities) back as input, while the discrete-time network uses multiple instances of the raw data.

Machine learning for user modeling

Recent work in machine learning for user modeling has attempted to learn not only group membership, but the group models themselves. Doppelganger (Orwant, 1996) was an attempt to learn and adapt these group models based on user behavior, and in turn to use these models to predict the behavior of future users. This system was used to model user preferences across a variety of applications. Doppelganger will be covered in greater detail than the rest since it bears the greatest resemblance to this dissertation.

Doppelganger used a pragmatic (bottom-up) user modeling system as opposed to the cognitive models used in many previous systems. Although inferences may have been made about a user's cognitive state, they were done without forming a formal cognitive model. Doppelganger was also implemented as an on-line distributed agent, meaning that a version of the modeler existed on many machines and each modeled a user in slightly different ways (eg. a user's home and work computer would share data, but not necessarily form the same model). Applications or other agents would then query Doppelganger using a client/server architecture. Although this opened some security concerns over the privacy and access of a user's information, the author did take a number of steps to achieve a high level of security.

Information was fed to Doppelganger via a number of software and hardware sensors. Since the information from these sensors may have often been incomplete or erroneous, an

accuracy estimate was provided with each sensor. Although the majority of Doppelganger's sensors were unobtrusive, the system did allow for user prompting by sending a query to the user over email and parsing the response, if any, for useful information.

Doppelganger attempted to borrow from the stereotype literature by using what the author called 'communities'. These were typical stereotypes found in many user modeling applications with the difference that membership in these communities was not "all or nothing". Users belonged to all communities based on a matter of degree. These communities were also not static, as they were computed as the weighted averages of their membership (or user models). The communities were therefore somewhat dynamic, changing as the user base changed. As with Rich's stereotypes, however, the communities were still limited to a pre-defined number. A set collection of labeled communities were defined at program creation, and only the membership of these groups was adjustable.

Doppelganger used a variety of learning techniques linked, in part, to the different sensor input streams. Examples of these techniques given in Orwant (1996) include: beta distribution, linear prediction, and Markov models. The beta distribution was used for boolean inputs (sensors) and accepted a string of examples of the boolean variable and used the mean and variability of the distribution to determine the expected value and its probability. Note that there was no order given to these samples and therefore, equal weight is given to all sensor input regardless of recency.

Linear prediction was a method used by Doppelganger which did contain memory. In particular, cyclical patterns such as login times were used to help predict a user's behavior. Much work could be done in advance if the system knew when the user was likely to log in next so this

learner predicted future patterns (with decreasing accuracy into the future) based on previous repetitive (though not necessarily perfect) patterns.

Finally Markov Models were also used (Orwant, 1995) for situations that were dependent on the current state of the system and the user. Weighted probabilities were once again used to determine (predict) which state the user (or system) was likely to enter based on the current state information (busy, idle, etc.). Doppelganger used eight Hidden Markov Models to describe the types of states that were likely for its users. These states resemble stereotypes very closely and include, for example: hacking, idle, frustrated, writing, learning, playing, concentrating, image processing, and connecting. The system used its sensors to determine which state a user was likely to be in, and from there, which Markov model/stereotype most closely described the user's behavior pattern.

There are two areas in which the Doppelganger system may be compared to stereotypes: Communities and Hidden Markov develop models of the user's state. Although Doppelganger did allow for dynamic models in the former, the latter were static. Even with the dynamic communities, however, there were a fixed number of communities (22) with set labels (artists, children, students, etc.). The communities were only dynamic in that their predictions may have changed based on the changing behavior of their membership and that their membership was not binary.

Multi agent systems

Alexandros Moukas (1997) used user-modeling techniques in an evolving multi-agent

system. The multi-agent system was developed to model user's preferences in web browsing and suggest new web pages to improve the user's performance and experience. The system created an evolving system of agents that competed and cooperated in a limited-resource environment. The two types of agents that inhabited this environment were information filtering agents', which were responsible for the user's profile, and 'information discovery agents', which controlled the gathering of information.

Information was gathered about a user by observation (browser bookmarks, history files) and direct questioning. In order to address ethical and privacy issues, the author also allowed users to view and modify their profile at any time.

Since these agents were adaptive, the filtering agents responsible for a given user changed as that user's goals and preferences changed. Since each user had their own closed system of agents, these changes only applied to the current user. Agents adapted and mutated using genetic learning techniques and performance of the agents was inferred by observing the user's interaction with the environment. Filtering agents received credit for good choices and lost credit for bad choices. Filtering agents also 'payed' information retrieval agents that were used for good results.

Although stereotypes were not mentioned in this article, similar constructions called 'packages' were used. These were pre-trained collections of agents that reflected an interest in a particular area such as 'Agents', or 'Soccer'. The user could, at any time, add one of these packages to their profile to express a new interest. As with Doppelganger, these packages (stereotypes) were pre-defined (pre-labeled and set number).

Adversarial Modeling

Burns & Volkmeyer (1998) researched participant's models of other human opponents and their utility in competitive games. The authors extended this study to include recursive models up to the second order ("What I think that my opponent thinks of me"). They referenced Thagard (1992) for proposing the need for recursive modeling in competitive games.

Pairs of participants were asked to play a purely adversarial two-player game in which an 'avoider' selects a number from one to three and the 'chooser' tries to select the same number. A sliding payoff matrix assigned positive and negative scores based on the success of each opponent. It was a 'zero-sum' game from a game theory perspective, meaning that random selection of responses should have led to a tie. Optimal strategy, however, as shown by game theory, put the advantage with the avoider.

Modeling was measured by self-report using descriptive assessment pairs (humorous-serious, negative-positive, hard-soft, rational-intuitive, and risk taking-risk avoiding) on a seven point scale. Each participant was asked to rate themselves (zero-order), their opponent (first-order) and how they thought their opponent would rate them (second-order). The authors argued that relative performance was more important than absolute performance which they used as justification for calculating all scores as relative for the two players involved. Only relative first-order and second-order modeling accuracy was used for analysis and only second-order was positively correlated to performance.

Adversarial problem solving

Thagard (1992) proposed to demonstrate the cognitive processes necessary for successful opponent modeling. He used ECHO, a connectionist model of explanatory coherence, to simulate examples of opponent modeling and deception. Thagard looked at a wide variety of examples for opponent modeling from diverse areas such as war strategy, business and game playing.

The author began the analysis by listing some common ‘rules’ of Adversarial Problem Solving (APS). They included:

- a. Constructing a model of the opponent (including situation, past behavior, competitiveness and attitude toward risk among others)
- b. including second degree modeling. (Opponents model of you)
- c. use of this model to infer an opponent’s plan, and inclusion of this plan in your model
- d. use of the revised model to devise plans that the opponent may not expect
- e. finally, taking steps to conceal this plan from the opponent. Thagard claimed that only the last two steps are unique to adversarial problem solving.

This research considers Thagard’s most important contribution to be his discussion of deception. It was in this article that the author suggested second-order modeling was critical for successful deception. It is necessary to understand how the opponent will interpret ones actions in order to cause an erroneous interpretation.

Thagard raised a very important point when he suggested that this recursive model required very little extra representation (space/memory) due to the necessary assumption that the opponent has roughly the same cognitive abilities as oneself. In competitive-agent environments where human and software agents were designed to be interchangeable, this assumption was often not the

case. Thagard's research does not suggest the likely outcome of the modeling however, when the opponent's cognitive ability is not known such as with the proposed limited scope Turing test (believability measure).

Laird (2000) used agents from the popular game 'Quake' in an attempt to add anticipation to the autonomous agents (QuakeBots). QuakeBots have been used with increasing frequency for competitive agents due to their complex environments, well defined rule sets as well as the interchangeability of human and computer agents. The Quake environment also had the added difficulty of being a real-time (as opposed to turn based) competition. Any model of a plan must have included time information to have been effective.

The author saw a need for anticipation with his intelligent bots after many attempts to improve performance by adding very specific 'scenarios' as models of opponent behavior. These scenarios were fixed, pre-programmed and relied completely on the author's observation of the bot's past performance against opponents. 'Anticipation' was thought to allow the Quake-bot to determine and respond to its opponent's moves dynamically.

The author didn't explain the architecture of the intelligent Quakebot, but it seems to have been a hierarchical state machine with some measure of planning incorporated for the anticipation. Planning may have been implicit, however since it was stated that there was no automatic progression from one state to the next, with each action selected by continually testing the current situation. Although the algorithm could 'look ahead', its decision was based on the current state of the environment.

In order to add anticipation to the Quakebot program, the author first had to add the

capability to model the opponent. Assuming the opponent used a similar state machine to its own (the author not only assumes a similar representation but also similar goals and tactics), the Quakebot assumed the role of the opponent until it produced a useful prediction or determined that there was not enough information with which to predict. This was determined by how much information the bot knew about its opponent, as well as how useful the prediction was likely to be. For example, anticipation was not used when the bot did not know where its opponent was or when the next move was obvious (avoid a shooting opponent). Also, the opponent model did not appear to be a complete duplicate of its own state machine, but only the uppermost level of the behavior hierarchy.

The author creates three new sub-states to take advantage of this anticipation; Hunt, Ambush and Deny-powerup. These new states were activated if it was seen that the bot could reach a position prior to the anticipated arrival of its opponent. Some learning was incorporated into the algorithm in the form of 'chunking'. This process allowed the system to pre-compile a set of rules that were shown to be effective on repeated occasions and avoided the need for the modeling step under repetitions of this situation, possibly saving critical time. The author also mentioned possible extensions to his algorithm using; recursive, enemy specific and adaptive anticipation. No results were shown on the effectiveness of adding anticipation, but it was mentioned repeatedly throughout the article that an agent's believability should play at least the same importance as its effectiveness.

MacInnes et al. (2001) demonstrate the use of recursive modeling of software agents in an adversarial environment. In many adversarial environments, agents need to model their

opponents and other environmental objects in order to predict their actions and outperform those opponents. In this work, the authors used Deterministic Finite Automata (DFA) for modeling agents and assumed that all the actions performed by agents were regular. Every agent assumed that other agents used the same model as its own but without recursion (a necessary assumption to avoid infinite recursion). The objective of this work was to investigate if recursive modeling allowed an agent to outperform its opponents that were using similar models.

The authors developed a 3-D “Quake-like” engine to test recursive modeling of autonomous “Quake-Bots” or in their case “Maze-Bots”. Each agent had imperfect knowledge of the world, using the same perceptions that a human agent would have, although in mathematical form. Sight was limited to a fifty-five degree field of view, perception was blocked by walls, and agents could ‘hear’ the other agent if a gun was fired. Agents had unlimited bullets in their guns, but a second shot could not be fired until the current shot hit an object, making accuracy a very important variable.

Each bot differed only in the depth of recursion it used in its modeling process. Levels from zero to three were tested with each bot fighting the others a total of fifteen times. Level zero recursive bots did not model their opponent at all; level one modeled their opponent; level two also modeled what their opponent thought they were going to do and so on. During the first experiment, opponent modeling did improve maze-bot performance but the optimal depth of recursion was one. Depth two and three recursion, while better than zero, were significantly worse than depth one.

Considering the importance of shooting accuracy to the prediction function, the authors made an attempt to improve this function to test the unexpected first results. Since any bot who

modeled where their opponent was going to be, had to use this location in an attempt to try to 'shoot ahead' of their target, this calculation was critical. In fact, any slight error in this function would be compounded with deeper recursion since it would be used at every recursive depth. In a second experiment the authors improved the accuracy and retested the Maze-bots. With the small increase in accuracy, the recursive bots still performed better than with no opponent modeling, but the optimal recursive depth actually improved depth two before a decline in performance was noticed. Since depth two in this study was the theoretical level stated in cognitive research, the optimal depth was found to be at the level where deception was to occur (Thagard, 1992).

Gender and spatial ability

Although a variety of variables will be discussed with regard to individual differences in the course of this research, none have the volume of research background associated as with gender differences in spatial ability (O' Keefe, 1978; Galea, 1993; Astur, 1998; Gron, 2000; Shore, 2001; Choi, 2002). Although this may seem a controversial topic at first glance, research (this dissertation included) is focused on understanding what differences, if any, exist and how to use this information to avoid bias in future spatial tasks.

A few of the following references deal with navigation strategy and not spatial ability itself. It should be noted that these are included insofar as these tasks are a sub-group of spatial task and that this task contains a navigation component. It is not suggested that this is a navigation task, but only that skill in maze navigation will aid in finding and defeating an opponent.

Early research in this area (O'Keefe, 1978) often focused on the traditional view that men tended to excel at math and spatial tasks, while women tended to excel at language. Although these gender differences are still reliably found in most tasks, recent evidence (Voyer 1995; MacInnes, 2001; Voyer, 2000) exists for learning effects and mitigating factors in the results in addition to some biological causality. One of the major factors that has been suggested is a difference in strategy employed by men and women (Dabbs, 1998; Harris, 1978; Lawton, 1994). Where women tend to use a strategy of landmarks, men are believed to use more cardinal coordinates. This is also referred to as 'Route' (F) versus 'Map' (M) strategy with neurological differences based in the Hippocampus (O'Keefe and Nadel, 1978). This explanation was adopted in recent papers on spatial tasks within virtual environments including the Hebb/Williams mazes (Shore, Stanford, MacInnes, Klein and Brown, 2001) and The Morris Water maze (Sandstrom, 1998; Astur, et al. 1998).

Biological and neurological accounts of these differences have been explored using novel (non-standardized) virtual mazes. Grøn et al. (2000), used complex computer generated mazes with landmarks while subjects' brain activity was monitored using functional Magnetic Resonance Imagery (fMRI). In addition to significant gender differences in completion time, the authors found that men and women use different areas of the brain during navigation. While both sexes had a great deal of overlap in areas that were common in spatial navigation, they differed in that men also showed activation in the left hippocampus and women activated the right parietal and right prefrontal cortex. An (oversimplified) summary of these results may reflect strategy differences since the prefrontal activation in women would be useful as working memory kept landmark information,

and the reliance on the hippocampus for men may reflect episodic memory needed to process multiple geometric cues.

Choi et al. (2002), have also discovered links between spatial ability and testosterone levels in men. Men and women were tested for maze performance while recording strategy information and testosterone levels. Men with higher levels of testosterone showed a stronger inclination to report using cardinal directions, while testosterone in women had no effect. It was not reported, however, if this also lead to improved performance on the spatial task.

Recent behavioral experiments have also looked into this question with equally interesting results. Three studies (Sedighian, 1996; MacInnes, 1999; Voyer, 2000) have demonstrated that experience in spatial tasks is just as, if not more, important than gender in some of these spatial experiments. Voyer, using a number of spatial tasks, also asked subjects about their preferences for a variety of toys and sports involving a spatial component. It was found that the gender differences were not significant when considered with certain combinations of preferences for toys and sports. MacInnes (2001), using the same Hebb/Williams mazes as Shore et al. (2001) also found evidence for experience as a limiting factor on gender differences in maze completion time. Although men tended to be faster than women, it was experience in three-dimensional computer games that was the critical factor. MacInnes hypothesized that these results differed from Shore et al. in two respects. MacInnes had one female subject who rated herself as 'high' for gaming experience (and this subject performed at par with men of the same rating), while the Shore et al. study had none. Slight changes to the maze itself made the environment closer to the games which it emulated (collision detection was modified to be the same as the first-person shooter 'Quake'),

which may have removed undue variance in the experienced players masking an effect of experience in the Shore et al. experiment.

Finally, a meta-analysis of results in spatial tasks by Voyer (1995) has shown a reduction in gender differences over recent years. This is certainly due to short term changes (learning, upbringing, lifestyle, environment, task, etc.) since evolutionary explanations are very unlikely to progress so quickly. It should be noted that evolutionary explanations for gender differences should not be confused with the biological or neurological results found above. It has not been shown (in this research) whether testosterone and brain differences between men and women were a product of nature or nurture. What is clear from this data is that experience and other social factors influence gender differences in spatial ability but that some of these differences may be genetic.

Chapter 3: Proposal

It was the intent of this dissertation to test a series of AI algorithms in their ability to model human opponents in a game-like, competitive environment. A number of software tools were implemented including a three-dimensional, multi-agent environment and all intelligent algorithms used as the brains for the software agents. Rationale for choice of algorithms will be described here, but the implementation will be discussed in chapter four (Methods).

The first algorithm was based on a straight forward, state machine known as a Deterministic Finite State Automaton(DFA) (Minsky, 1968). DFAs are often an attractive option since they are (relatively) simple to implement and provide behavior that is very easy to understand and model. All behavior is hard-coded during implementation and the decisions of the algorithm can be completely predicted by the internal state of the algorithm and the external state of the environment at any moment in time.

The attraction of this type of algorithm, apart from its simplicity, is in knowing exactly how an agent will behave in a known situation. The designer of an agent can tailor the behavior of the agent exactly as they see fit under certain key conditions, such as determining the optimal turn ratio to fire at an opponent. The particular DFA in question also had the advantage of having data available from DFA versus DFA recursive matches in the proposed environment (MacInnes, 1999).

These advantages quickly turn into hindrances, however, since all possible combinations of state and environment must be hard-coded into the algorithm. It quickly becomes apparent that for large or complicated environments, the only choices are to create very complex rule sets, or

to make them more generic and less correct for any specific scenario. The lack of variety can also cause problems in competitive situations since the DFA's patterns, no matter how complex, are rigid and open to discovery and exploitation by an opponent. DFA opponent modeling suffers similar restrictions since the DFA must assume that its opponent is another finite state machine.

It was decided that the DFA would consist of two primary states, *search* and *fight*. While the fight state used information about the opponent in its decision-making process, the search state did not. Since the DFA could only use its own state and the state of the environment in its decisions, it could treat the opponent as any other environment variable, but only when the opponent was visible.

The second agent tested was based on a neural network, trained on observations of human opponents. Attempts were made to maintain the advantages of DFAs while improving on their shortcomings. The base algorithm maintained the DFA for much of the agent's decision making, but a predictive neural network was used in certain areas to improve the algorithm's core performance as well as its opponent model.

The focus of the neural network was to predict the opposing agent's current and future location in the arena. The neural network accepted information about a human agent's recent locations and used this information to predict the opponent's location. To simulate a temporal component while keeping the benefits of a feed forward/back propagation neural net, the five most recent time samples were used for the input layer (see methods and figure 3 for details). The network could theoretically predict to an infinite time in the future by using its current prediction as input to guess at two (and so on) time samples in the future, but it would reach a stage where

compounding errors would negate any benefit from such a prediction.

Since the 'pure' DFA was unable to use any information apart from the environment in its search state (effectively blind search), the neural net took over and tried to predict where an opponent could have been when the sensors provided no information. Since this would only be effective for a limited time after the last sighting, a time limit was used before the agent went back to blind search. Since the neural network could easily be tested to determine its accuracy over time with limited information, the time limit was determined by choosing an acceptable degradation in accuracy.

A second spot where the neural network was used was in the firing calculations of the fight state. Where the DFA assumed the opponent was another DFA, it shot at a location where it would go if it were being fired at. Although it could calculate distance and proper lead time for a moving target (for recursive levels higher than one), it always assumed that the target was traveling straight ahead (usually correct for a DFA). The neural net however, used its predictive ability to shoot where it thought the opponent was likely to be.

Traditional stereotyping, as it applies to user modeling, refers to an attempt to model user behavior by determining a user's membership in one of a set of pre-defined groups. Behavior of these stereotypes could be used to supplement or replace the model of an individual when that model was found to be lacking or erroneous. Early work in stereotypes restricted learning to determine group membership (Rich, 1979) from a set of predefined models. Models such as Beginner, Novice, and Expert would be developed off-line and an attempt would be made to match each user to their appropriate stereotype in an on-line matching process.

Static stereotypes (Rich, 1979; Orwant, 1995), however, lead to a number of potential problems. The most serious of which is usually the problem that the stereotypes themselves need to be known beforehand. Since the groups must be defined during development, there is usually little data from which to base the groupings. It usually fell to the developer's biases alone to decide on the quantity and labels of the stereotypes. These stereotypes tended to be very vague and potentially poorly representative of a user's actual performance. Also, an accurate stereotype may degrade as a user's performance and preferences change over time. A user may gradually become more experienced, they may suddenly shift goals or preferences, or an opponent may change tactics in an attempt to utilize deception. All of these shifts reflect the difficult machine learning problem of 'concept drift' (Widmer and Kubat, 1996) where the concept to be learned or modeled changes over time.

This proposed research attempted to extend earlier work on user modeling with stereotypes (Rich, 1979; Orwant, 1995) by allowing a clustering algorithm learn these groups dynamically. A hybrid, K-means clustering/ Kohonen Self Organizing Map (SOM) was used on the training data (observations of human patterns). A single dimensional, ten unit k-means clusterer was implemented to provide distinct edges for the ten potential clusters. This algorithm was then modified to allow for the smoothing process found in typical SOM algorithms (Kohonen, 1982) to allow for higher mobility of training samples early in the learning process.

The same data that were used for the neural network was fed through this clustering

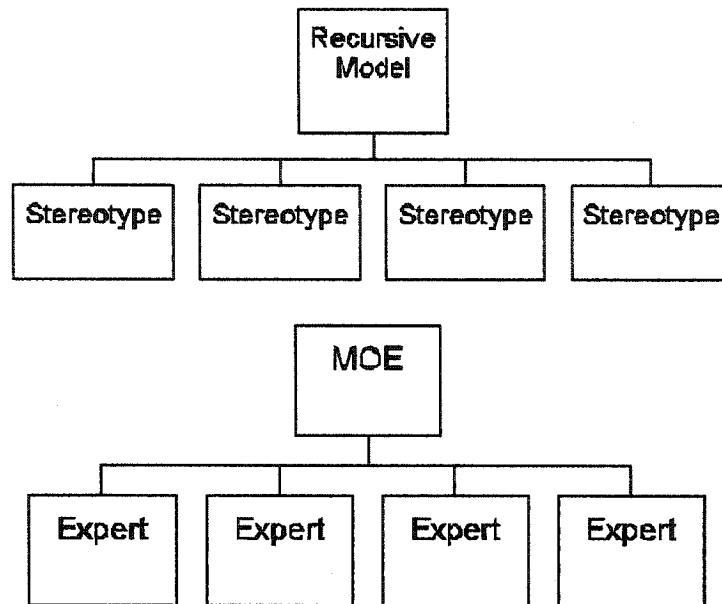


Figure 1 The similarities between theory of recursive modeling and implementation of an MOE were critical in the choice of machine learning technique.

algorithm producing a series of smaller, yet more similar and specialized sub-groups. The theory was that these clusters would represent different styles of play, or opponent stereotypes. Since these clusters represent opponent strategies, the theory and the mathematics of the clustering algorithm both predict less deviation within these smaller groups than in the overall data. This prediction was tested and reported in the results section. Although it is tempting to label these clusters as ‘styles of play’ or ‘strategies’ it should be emphasized that formal labeling of these groups is a very difficult process. Although clustering algorithms in general are excellent at grouping similar behavior, it is not always obvious what these groups actually mean.

The algorithm that most closely resembles recent theory on user modeling and deception, however, is the Mixture of Experts (MOE) with its similarities to stereotyping and recursive

modeling (figure 1). The clustering stage, mentioned above, was used to create specialized data sets which became a series of experts.

The motivation and expectation for using this algorithm was that these experts would provide for more precise modeling of user behavior. At the very least, it should (Jacobs, 1991) provided the same accuracy with less training required due to the reduction of variance in the training samples. This separation of behavior had the additional advantage of allowing the agent to choose one of these experts to determine its own choice of action.

The problem of concept drift was also handled by only modeling the opponent's recent history. Once the clustering produced workable stereotypes in the off-line phase, it was simple to compare an opponent's current behaviour to these stereotypes on-line. Again, the accuracy of the algorithm suggested that the ten most recent samples be used by this process.

The stereotypes themselves were implemented as a series of feed-forward/back-propagation neural networks. Since the environment was dynamic and opponent's patterns may only become clear over time, a time-sensitive neural net was used as with the neural net agent. The nature of the domain (real-time games) and the possibility of concept drift suggest that a time sensitive neural net should be used. This was also an ideal way to *implicitly* incorporate planning in the model. (Klapper-Rybicka et al., 2000; Mozer, 1993). Classification was performed in two stages: Off-line clustering used to determine which samples will be used to train the different Net/Stereotypes; and on-line classification used to determine to which Net/Stereotype the current user belongs (Jacobs & Nowlan, 1991).

Every agent solution was tested in a series of matches against human opponents according

to the following experimental design. Volunteer participants were brought in two at a time and asked to compete against a series of agents (one opponent at a time) in a multi-agent competitive arena. A potential human opponent was present in an adjoining room for all matches, so each participant could not determine whether they were competing against a human or software agent. This arena was based on the virtual maze used for MacInnes (1999), but modified to allow for new software agents as well as Human/Human agent matches across a TCP network.

Dependent Variables

Since the success of an agent depends as much on the goals of the environment as the quality of the algorithm, a number of dependent variables will be measured in this research. Accuracy, effectiveness and believability usually play a role in the performance of a software agent, but often in differing degrees depending on the goals of the environment and the preferences of the human agent.

The *accuracy* of the software agent was measured by its ability to predict its opponent's position given its current state and (for some algorithms) its past behaviour. This measure most closely resembles what is typically used to judge the performance of a machine learning algorithm, and is also the most clear cut in interpretation. The algorithm that predicts its opponent's position with the smallest error, measured in two-dimensional Euclidean distance was judged to be the most accurate algorithm. Although this dependent variable was the simplest to measure, its usefulness on its own may be limited. In a dynamic environment, the ability to foresee an opponent's location is only as useful as its ability to make use of that information. An agent whose sole purpose is

prediction of location would be best weighed by this measure. A good example would be an on-line trading agent whose only purpose is to predict price trends and report the results to a human agent for final decision.

The *effectiveness* of an algorithm, although linked to accuracy, is more useful in determining the performance of an agent as a whole in this environment. This measure looks at how well an agent is able to compete with human opponents and is determined by the agent's score (number of kills) out of five in each of the matches. Although this is influenced by the accuracy score from above (a more accurate algorithm should be more effective), it also includes the ability of an algorithm to make use of the accuracy information. Both of these measures were used since accuracy can be said to measure the algorithm where effectiveness can be said to measure the agent. This is the typical measure of an agent's goodness in many multi-agent environments including computer games. It is usually assumed that the tougher an agent is to beat, the better it is.

The final dependent variable, *believability*, looked at how well the software agent can impersonate a human agent under similar conditions. At the end of every match, participants were asked to rate their opponent as to whether they thought it was human or computer and how confident they were in that guess. Since every participant played both human and software opponents, there were three possible scores that could be used for this variable. First, the original score as entered by the user gives a number in the range from one (very uncertain) to five (very certain) with a positive number reflecting a guess of human and negative reflecting a guess of computer opponent. Second, an absolute difference from the 'perfect' answer would contain all

of the information above, but also allow a comparison of human and computer opponents. For example a score of zero would be a correct guess and negative ten would be the lowest score (highest error) possible.

The final measure would again be for computer opponents only, but would be the difference between two Turing scores by that human agent. Earlier scores could have been faulty or variable since different people may give varying scores to the same level of believability. The idea behind this last score was that the Turing score for a software opponent reflected how well an algorithm mimicked human behaviour, and the Turing score for a human opponent reflected how observant or varied the person was doing the rating. By subtracting the Turing score for the Human opponent from that of any particular software opponent we get a score of how 'Less than Human' (LTH) any software algorithm was for each participant. ($LTH = \text{Human Turing} - \text{Software Turing}$; and was calculated separately for each of the nine software opponents) This would be a relative score showing how much more or less human the software bots were versus the humans themselves. This score was more closely related to the typical Turing test since it takes into account individual differences in how accurate people are in spotting human opponents in this task. The theory is that the believability of an agent should be tempered by how critical a person is in judging this attribute (reflected by this rating on actual humans).

Independent Variables

Within Subject

Every subject played the same set of ten matches as every other player. One match was

against a human opponent to achieve a base line for the variables mentioned above, and the other nine were against software agents. In addition to the human/software split for opponents, each of the remaining nine matches were fought against the three agent algorithms at three different levels of recursion. DFA, neural net, and MOE were all implemented with zero (no recursion), one and two levels of recursive opponent modeling. Level two recursion was chosen as the highest level due results from MacInnes (1999), which suggest that optimal results are achieved at recursive level one or two depending on the accuracy of the prediction. This was deemed sufficient since none of the algorithms are likely to achieve better prediction than when the opponent's strategy is known (DFA vs DFA). Finally, one pair of expert subjects was run in the study a number of times in a single evening to determine if the results found would be reliable over prolonged exposure (this final variable was for Experiment two only, see methods and results for details).

Between Subject

A number of other variables, while not able to be controlled in the design, were still observed as between-subject variables. Since learning advantages can be important in any study, three different variables for this were recorded for consideration. Each subject was asked to rate their computer game experience as a measure of learning in these environments prior to this study. The order of the opponents was randomized for each pair of subjects to determine if learning influenced performance within the ten matches themselves. This randomization was the same for each pair of subjects since it was necessary to give each pair the same sequence to synchronize the human versus human match.

Due to previous literature of gender differences in spatial ability, the sex of each participant was recorded for analysis. Even within this environment itself, Shore et al. (1999) found significant gender differences while MacInnes and McCabe (2001) did not. Since the question in both cases became the importance of gender versus game experience, every effort was made to acquire a strong selection of participants with all possible combinations. While it is often difficult to find male participants who rate themselves as having low game experience, it is even more difficult to find females who rate themselves as high. It was decided that the best chances to find a strong cross section without biasing the sample too greatly was to recruit participants from the Faculty of Science and the Faculty of Computer Science at Dalhousie University.

Other information was gathered about the participants to ensure that it was not influencing the results. Although not necessarily of interest to the study, it was necessary to determine the effect, if any, on the dependent and even other independent variables.

Summary and Predictions

The approach, summary and predictions of this dissertation is approached from two perspectives, the human and the software, and are discussed below

Software

The Mixture of Experts (MOE) algorithm, then, uses a variety of machine learning and AI algorithms to achieve its goals. Traditional AI is represented by the DFA which still forms the core of all of the algorithms tested. Unsupervised learning is used to cluster user behaviour into dynamic

stereotypes, which may enhance the algorithm in terms of effectiveness and believability. Modeling the other's behaviour should assist in defeating that opponent, and using models learned from human observation should make for a more human-like agent. Supervised learning is used to train the neural networks to predict the next location of an opponent. This should have an impact on both algorithms (Neural net and MOE) that use these predication methods. Finally, reinforcement learning uses these predictions to help the agent succeed in its longer-term goals against the opponent. A variety of solutions are used for this, including temporal input to the neural nets as well as recursive modeling itself.

Although the primary goal of this research is to determine which factors influence the 'goodness' of software agents in competitive, multi-agent systems, there are also a number of expectations that need to be examined. It is often believed that machine learning techniques that model cognitive processes can be superior to those that do not. Given this outlook, it is reasonable to assume that the neural network should outperform the DFA and the MOE in particular should outperform them both in many of the important measures. Given the nature of the dependent variables, however, it is not unreasonable to assume that different algorithms will perform differently under different measures. Although accuracy and effectiveness are closely tied, believability is more elusive in its origins, and may conflict or even interact with effectiveness and to a lesser extent accuracy.

Human

From the human perspective, our agent interaction should be able to speak to individual

differences in spatial ability. Since computer games are highly spatial tasks, this research will look at a number of individual traits as between subject-variables, with focus on gender and experience. Recent evidence has pointed to a combination of biological and learning explanations for gender differences in spatial tasks and spatial strategy (usually focusing on navigation). Although confounds exist in nearly every study looking at experience and gender (due to confounds that exist in the population), it is expected that the gender effect will be reduced or eliminated based on participant's prior experience.

Chapter 4: Methods

Two agents at a time were able to traverse the maze while engaging in a shooting duel with each other. To establish a connection with current literature (Laird, 2000) and current applications, the theme and feel (but not the graphics) of the arena were designed to resemble ‘first-person shooter’ multi-player computer games. While previous research (MacInnes, 2001) looked at competition of multiple software agents, these experiments always contained at least one human agent. Human versus software agent matches were implemented on a single computer, while human versus human matches were implemented on two computers over a small network. This allowed players to compete while being in different physical locations.

The remainder of this section will discuss three distinct areas of this experiment’s methods; the construction of the three dimensional arena that was used as the multi agent environment; the creation of the agents used in the experiment and the machine learning algorithms used to guide them; and finally the experiment used to test the software agents and human agents.

Arena

The arena used for all testing in this dissertation was a three-dimensional, graphical representation of a small two-dimensional maze. It was based on OpenRT-3D, a three dimensional graphics engine designed for cognitive testing and was based on the real-time code used in OpenRT (MacInnes, 2001). The engine was written in C++ and uses the OpenGL graphics library for all rendering. Although the base OpenRT was written for multiple platforms, this version of the arena was implemented to maximize performance on a Windows [copyright]

computer. As discussed in MacInnes (2001), measures were taken to maximize real time performance of this application and minimize interruption from the operating system, but due to the special requirements of this experiment, less strict real-time strategies had to be used in some circumstances.

OpenRT was able to create a soft (not guaranteed), real-time environment with millisecond accuracy through a number of system settings and by implementing a custom window class for each environment. In the Windows environment, OpenRT was able to minimize interruption from the operating system by a) avoiding the Windows messaging loop b) increasing the application priority c) disabling the mouse cursor redraw, and d) using specialized high speed libraries for drawing and user input. These methods enabled the application to reduce operating system interrupts to less than one millisecond duration throughout a testing period of nearly one and a half hours (MacInnes, 2001). Standard applications can experience interrupts for one hundred milliseconds or longer.

Since the above strategy involved minimizing operating system influence, it could have negative performance implications on the network connection used in some of the experiments. Priority settings for the application could not be changed, for example, since OpenRT set itself as having a priority that was higher than the operating system itself. The windows messaging loop could also not be bypassed due to the nature of the network connection. A 'Peek Message' loop was implemented to replace both of these methods (see appendix C for sample code). In this system the application 'peeked' at the windows messaging loop to see if any messages were queued and dealt with them if necessary. All remaining computer time was used to ensure the maximum responsiveness of the application. This kept the interrupts to a level that was acceptable

so as to maintain the pre-determined refresh rate of thirty frames per second.

Input and drawing techniques were also used from MacInnes (2001) to enhance precision timing capabilities. Direct Input (copyright, Microsoft) was used for keyboard input and OpenGL was used for high speed rendering of graphics. Both of these software development kits (SDKs) were developed, in part, to allow computer game programmers fast and regulated access to

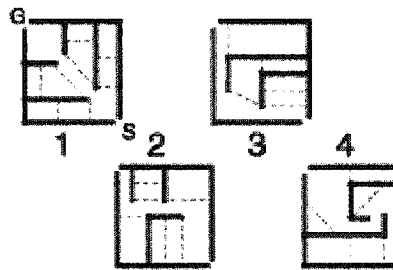


Figure 2 First four of the twelve Hebb/Williams maze, corner alcoves represent the start (S) and goal (G) of the mazes. Figure taken from Shore et al. (2001).

hardware on the computer that would not have been accessed directly in other software.

Although the feel and control of the arena was designed to mimic a typical computer game, the layout itself was taken from the set of twelve Hebb-Williams mazes [Hebb et al., 1946] used to study spatial intelligence in animals. These were chosen due to availability, the simplicity of the layout, and to allow for easy comparisons to studies looking at gender differences [Shore et al., 2001] and computer game experience [MacInnes, 2001b]. Mazes one through four were used to train various computer algorithms in experiments 0a and 0b. Maze two was used for

experiments one and two (see figure 2 for maze layouts, and Appendix B for snapshots) since it was the first maze in the sequence in which a majority of the maze was not visible from a single position.

Textures for walls consisted of a repeating chiseled rock pattern and the floor used brown tiles. Corner alcoves had red and green roofs to provide limited landmarks within the arena. The bots were simple, three-dimensional polygon depictions of a basic walking robot. The legs were cylindric tubes that produced an animated walking motion when the bot was in motion. The body was a simple sphere, chosen to simplify the sphere-based collision detection algorithm. A protruding ‘happy-face’ mask was attached to the front of the bot to allow visual detection of a bot’s current facing.

Measurements were made in arbitrary ‘units’, since their only purpose was for relative calculations. The arena measured 60 units along each side and 5 units in height, with alcove walls in opposite corners having a length of 5 units. Agent’s viewpoint was fixed at 1 unit above the floor, moved at a speed of 0.01 units per millisecond, and turned at 0.05 degrees per millisecond. Bullets traveled 5 times faster than agents at 0.05 units per millisecond. To get a perspective on these numbers, assuming the perception that the bot is moving at a ten kilometer per hour run, then 3.6 units were roughly equal to one meter. It should be emphasized that this conversion is based on a convenient estimate of perceived speed and is not indicative of the physics of the arena.

All code used in these experiments was written for this or previous experiments by the author. A number of open source projects were consulted for the implementation of some portions, and ideas were used as a basis for some portions. Complete code (nearly ten-thousand

lines) for this dissertation has also been released as open source, and can be found at <http://www.cs.dal.ca/~macinnwj> [Online ref, 2003]. Also, see Appendix A For a full Unified Markup Language (UML) class diagram developed during the design of this project.

Agents

The software version used for testing allowed for two autonomous combatants, or agents, in an arena at any time. Although a variety of agent types were implemented (see below), they were all equipped with a common set of abilities and restrictions to ensure fair matches, regardless of opponent. Since the purpose of these experiments was to test human and computer agents on an equal footing, it was important to ensure that neither had an advantage based solely on the information they received, or restrictions on how they could use it. All agents, regardless of type had the same perceptions of the arena (sensors) and the same ability to change the environment (effectors), they only differ in how they use this information (see appendix E for agent summary, and software states). Although human and software agents received this information in different forms (humans received visual feedback while software received mathematical information), it was decided that the fairest comparison was allowing the optimal feedback for each bot and will be discussed in greater detail for each agent. Forcing the software agents to use visual feedback (although not impossible) would be akin to forcing humans to navigate through a purely mathematical representation of the arena.

Agents had access to sensors that allowed them both visual and auditory feedback from the environment. All agents were restricted to a first person perspective, 55° field of view (FOV)

and could not see through walls. Agents could hear gunfire from either agent, a bumping sound when two agents collide and the sound of either agent dying. Representation of all agents comprised of a white sphere with a happy face mask, all on two blue legs. Legs were animated to produce a walking motion to improve the visual illusion of running. Diameter of the agent sphere was 0.5 units, and the bullets, only visible when fired, had a diameter of 0.05 units.

Each agent had a weapon that fired a single bullet at a time, and appeared as a bright yellow sphere. Bullets traveled five times faster than agents (0.05 units per millisecond), and had a radius of 0.05 units that was used in determining collisions. Bullets always traveled straight, continuing in the direction that the agent was facing at the time of firing. Bullets continued their full life span until they collided with a wall or another agent, at which time it disappeared and a new bullet was available for fire. Although agents never run out of bullets, they only had one bullet active at any time. Although real bullets may travel faster than five times the speed of a running person, this projectile was designed to model some common video games often use slower paint balls, rockets, etc. The slower movement also allowed for more defensive strategy on the part of the participants than a more realistic velocity bullet.

Effectors on the world consisted of five simple commands; move forward, move backward, turn left, turn right and fire the gun. These commands could also be given a number at a time for a combined effect (e.g. left and forward moved the agent in a sweeping arc). All

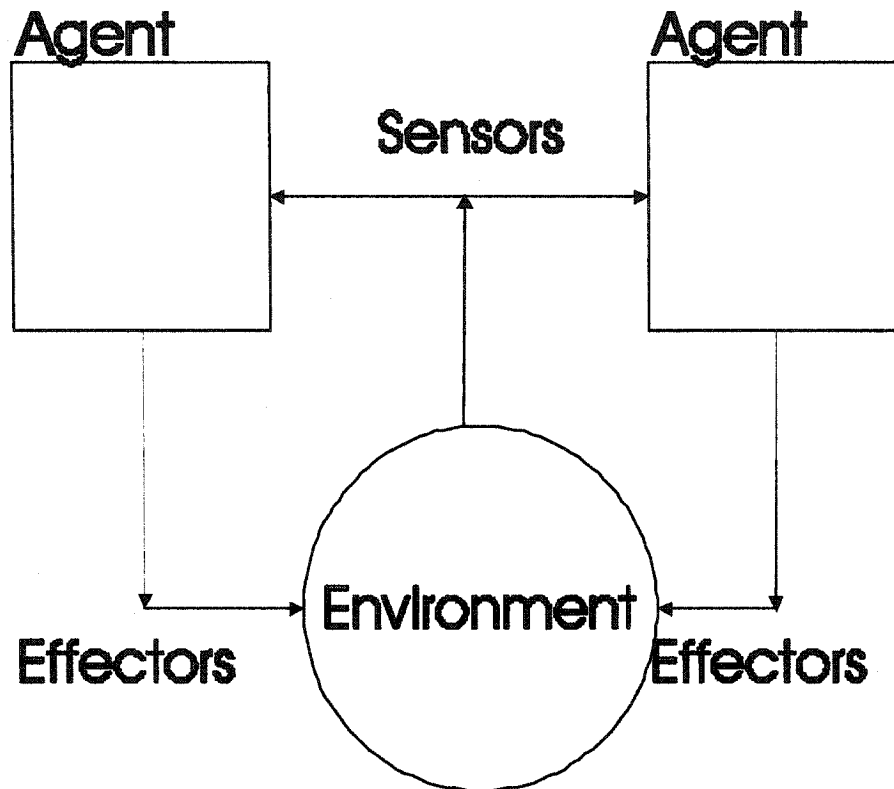


Figure 3 Two agents interacting within the same environment. Note that neither agent can completely control (no predict) the resulting environment state.

commands were acted upon the current state of the agent in a predictable way according to the simple physics of the arena. Velocity was measured according to the time lapse since the previous state and moved the agent in the specified direction. Time based velocity was chosen to ensure consistent movement in the case that thirty frames per second could not be achieved (this turned out not to be a problem due to the real time considerations mentioned above. Frame rate was consistent to thirty frames per second, with a standard deviation of less than 7.0×10^{-8} milliseconds).

As seen in the UML class diagram (Appendix A), every agent has access to a 'brain',

whose sole purpose was to use previous sensor information to produce effector commands for the current state or action. The 'brain' component was designed to allow for easy transition between agent types, and also provide a simple method of incorporating new controlling algorithms in the future using. By taking advantage of inheritance and polymorphism within the C++ language, any brain component could be interchanged with any other as long as all implemented the functionality of the base brain class. At its most generic level, this brain could be thought of as a black box that converted sensor information into output control (see figure 3). At the time of testing, only two agents (Human and DFA) had this brain implemented as its own component, but all had a *conceptual* brain that served the same function. It is important to note that there was no difference in the results of these two approaches, except that the encapsulated brain component would be easier to modify and replace.

Deterministic Finite-State Automata(DFA)

Deterministic Finite-State machines were imported nearly complete from MacInnes (2001) to be used as the base for the simplest of the computer agents. A C++ class was implemented to wrap the previous implementation and allow for easier integration with the current project. Since data was already available for DFA versus DFA matches, it was determined to be a good base line for computer versus human matches. The basic machine used for the DFA monitored the state of the world and actions of the other player. Since game theory states that two players are in constant interaction, the output of this model was the action to be taken by the agent *at that moment* and planning only occurred (implicitly) in the selection of discrete actions. The recursive

component (covered in detail below) added knowledge of the opponent to this decision in an attempt to produce a more successful action (since this and other computer algorithms are based on the perspective of the computer agent, ‘opponent’ for all of these algorithms will hereafter refer to the human agent). A number of DFA’s were tested, all using the same model, but differing only in the level of recursion used to predict its opponent’s next move.

The DFA algorithm is so called because only the current properties of its environment are used in decision making (state machine), properties of environment *uniquely* determine which state the agent is in as well as all transitions between states (deterministic), and there are a limited number of such predefined states (finite). The DFA used, consisted of two major states: Search state (the default state) and Fight State. Search state occurred whenever the agent was unable to determine the actions of its opponent (not in sensor range). In this state, all the actions of the agent were

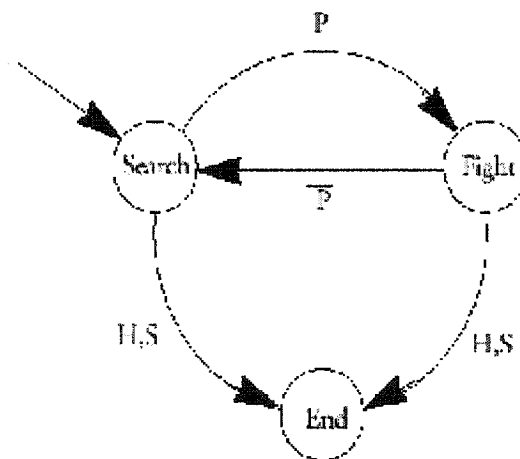


Figure 4 DFA state diagram. *Fight* and *search* are the main states when the opponent is present(P) and not present respectively. End state occurs when either opponent hits (H) or is shot (S).

decided based on the information gathered from the environment. As soon as any sensor provided information about the opponent, the agent entered Fight state. There was also an insignificant End state entered at the end of each match (See figure 4). Since true DFAs only based decisions on the current state of the algorithm, it was decided not to include estimates of opponent's position within the search state. It was left to other algorithms to predict an opponent's current position based on previous information.

The prediction function used in the fight state was at the very heart of the opponent modeling however, it was only possible to estimate where the opponent would be. When opponent modeling occurred, the algorithm would assume that the opponent was a state machine identical to itself. Since the DFA search state was a simple straight line (unless interrupted by an obstacle), triangulation was used to estimate the opponent's position. A triangle was imagined with vertices containing the Agent's current position (A), the opponent's current position (B) and the opponent's anticipated future position. Substituting the known distance (Dist), the known angle (θ) along with the ratio of agent movement to bullet movement(x,5x) to the law of cosines we get:

$$(5X)^2 = X^2 + D^2 - 2XD \cos(\theta) \quad \text{or} \quad (1)$$

$$24X^2 + 2XD \cos(\theta) - D^2 = 0 \quad (2)$$

since X is the only unknown in this polynomial we can solve using the quadratic formula.

$$X = \left(-b \pm \sqrt{b^2 - 4ac} \right) / 2a \quad (3)$$

(Where X is the predicted distance the opponent will travel before bullet contact) When opponent modeling did not occur, the DFA fired at the opponent's current location.

Within these states themselves, there exist a number of rules and transitions based on perceptions of the DFA about its environment and opponent. The default state within the search state was *blind search*, and consisted simply of moving in a straight line until an obstacle moved the agent into the *avoid* state. When an obstacle, such as a wall, blocked the agent's path, the avoid state recommended a turn in a direction based on the minimal angle to avoid the obstacle (whichever direction would bypass the obstacle in the least time). In this way, it was the layout of the arena that determined the search path that the DFA would use.

The fight state was further divided into three sub-states and evaluated in the following order; *shoot*, *evade* and *align*. Shoot fired a bullet as long as the agent was aligned with the position estimate discussed above, evade would attempt to move the agent out of the way when an opponent fired and align (the default) would turn the agent toward the location to which the shoot state preferred.

Neural Network

The neural network agent used the DFA as a base class to define much of its default behavior but overrode some key properties in order to improve its performance against human opponents. The neural network was incorporated into both the search state and the fight state of the agent, and aided in the prediction of the opponent's position in both cases. In the search state

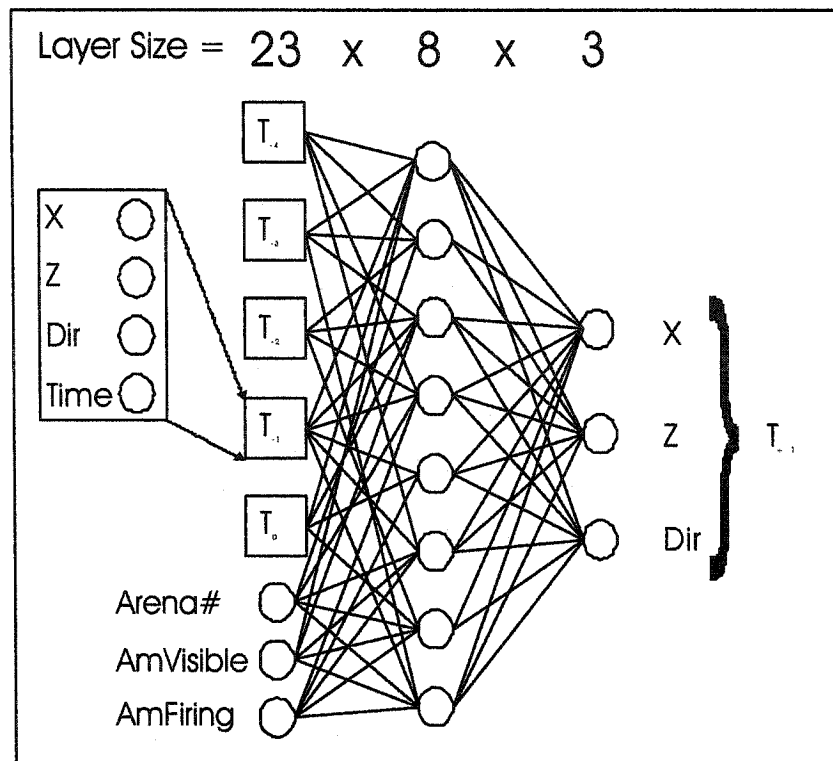


Figure 5 Structure used for both neural net and the experts of the MOE algorithms. Input contained five time samples of previous opponent observations and output was the predicted observation for the next time stamp. Input and output layers were consistent across algorithm, but number of hidden layers and nodes per layer varied to suit the optimal training for that algorithm.

it would suggest possible opponent locations to supplant the DFA's blind search. In the fight state (if modeling was used) the network predicted the opponent's future location to be used for firing calculations. Although the neural network was trained offline (not during match time), the trained network was then fast enough for on-line, real-time prediction.

The neural net was a standard feed-forward, back-propagation network that allowed for continuous variables in both the input and output layers. Two hidden layers, each with eight nodes, used the basic sigmoid activation function

$$f(x) = \frac{1}{1 + e^{-x}}$$

and a learning rate of 0.25 on the back propagation of error. The output layer consisted of three nodes representing the opponent's predicted position in terms of the change in location along two arena axis (ΔX , ΔZ) and the change in facing direction ($\Delta \theta$). The difficulties inherent in true recurrent neural networks (Elman, 1990) were avoided by simulating a time component with the input layers. The five most recently observed, discrete time samples of opponent information were used in the input layer, in addition to global information about arena version and whether the opponent was visible and firing. Each of the five input time samples used Cartesian coordinates for positional information along with nodes for directional facing and change in time (ΔT) since the last sample. Time information was included since velocity (and therefore future position) was calculated based on time, and the neural net could learn any anomalies in the preset thirty frames per second. The input layer, then, consisted of a total twenty-three nodes, four-by-five time sample nodes plus three for global information (see Figure 5). Using a traditional feed forward/back propagation neural network, but including the variables of a static number of recent time samples to the input layer was to allow for some measure of implicit planning in the network. This would simulate the temporal component of a recurrent neural net without getting the convergence difficulties inherent with their use.

The neural network was trained using data obtained from observations of eight participants

involved in multiple matches with the DFA as an opponent (see experiment zero below). This produced over thirty thousand time samples of agent positional data that were used as input to the neural network training algorithm. All data were converted from absolute observation to relative (e.g. $X \rightarrow \Delta X$) and normalized to fall within the observed minimum and maximum for those data before being fed to the network. Unless otherwise specified, all neural networks were trained on

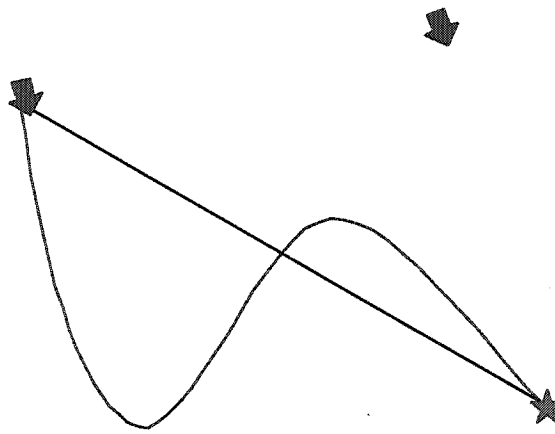


Figure 6 diagram of neural net firing prediction. The initial arrow represents an opponent's starting position and the curved line represents a neural net's predicted future path of the opponent with the star being the end point after X time samples. The triangulation algorithm from the neural net used the straight line as an estimate, but with the agent moving at a proportionately reduced speed.

ninety percent of the data and tested on the remaining ten percent. While other learning rates (α) were tested, 0.25 consistently produced the best results. Networks were trained to the point where over-fitting was observed, or until improvement rate was minimal.

The search state for the neural network agent was divided into two sub-states; *directed*

and *undirected*. The undirected state was identical to the simple DFA 'blind' search, and was the default when lack of information prevented a directed search. Any time that the opponent was not in view, but the agent had some information on the opponent (recent sightings, opponent firing), the neural network would be consulted to provide a best estimate as to an opponent's location, and proceed to that spot in the arena. Directed search would cease when sufficient information was not available, or when the agent reached this estimated location with no further information.

Fight state was also augmented using the neural net by including neural net predictions into the firing angle. When opponent modeling was used (recursion greater than zero), opponent's position was estimated using the results of the Neural net instead of the straight line assumption in the DFA. The law of cosines was still used to calculate the preferred alignment angle, but the opponent's heading was replaced with the direction vector to the neural net's prediction, and the velocity of the opponent was calculated as using the predicted distance of that vector. This allowed for calculations to use straight line trigonometry to predict firing angle even though the neural net's predicted path would likely be erratic over time. The assumption is that a curved path can be simulated with a straight path at a lesser velocity as long as you only care about the point of intersection (see figure 6).

Human

Every match in the current set of studies had at least one of the two agents as human and it provided the basis by which all other agents were measured. The information interface to the human was designed the same way as for computer-controlled agents, except that sensors and

effectors were fed to and from a human participant. The brain component was used to relay this information, but its job was merely to poll the keyboard for user input and send sensor information to speakers and monitor. Input/output issues will be discussed briefly here since they formed the basis of the human agent's effectors and sensors.

Keyboard input was handled through measures suggested by Macinnes (1999). Microsoft's Direct Input (copyright) was used to poll the keyboard for input without needing to use the window's messaging loop (freeing more time for graphics in the 'peek' message loop). Direct access to the current keyboard state also allowed the software to register more than one key press at a time during any given state. The four keyboard arrow keys, typically controlled with the right hand, matched the four movement commands available to all agents, and the space bar was used for the fire command.

Sensors for the human visual component were displayed via a computer monitor (see experiment details below for specs on equipment used in these tests). All information was displayed to a full screen, OpenGL window with the same first-person perspective field of view as the computer agents. Hardware acceleration and double buffering were used to ensure fast, quality updates of the display, and drawing was synchronized with the vertical retrace of the monitor to prevent tearing of the image.

Transfer Control Protocol (TCP)

The TCP player used the transfer control network protocol to communicate with a second computer to forward sensor input and effector output. This agent can actually be thought of as a

type of human agent since the only time it was used was when a human agent was competing with a second human agent on a different computer. The 'brain' of this agent is merely a shell that requests information and provides updates to a human agent on the other machine. Every human versus human match, then, is comprised of two human agents and two TCP agents; one human/TCP combination on each of the two computers.

TCP agents are also the only minor exception to the rule that agent output consists of only five boolean commands. To avoid potential computation differences in the resulting location on different machines, the TCP agent relied on its human counterpart to do calculations on the output keys that would be received by the TCP agent as final maze coordinates. This also prevented having to send time stamp and velocity information over the network which helped reduce network traffic. This is not a problematic difference with other agents since TCP agents are merely a local representative of human player that is using the standard effectors and sensors.

The TCP agent was implemented using a C++ encapsulation of the Windows Winsock libraries (portions based on sample code from[Barron, 2000]). Subjects were randomly assigned to the role of client or server on the network, although this process was completely transparent and irrelevant to performance after the initial connection. Relevant agent information was converted to TCP packets and sent and received on every frame. Winsock connections, like most network connections force a choice between speed of information delivery and chance of packet loss. In order to minimize agent location error, the choice was made to guarantee packet delivery and order, and accept possible time delays. As seen with the frame rate standard deviation (above), this turned out to be a non-issue.

Mixture of Experts

The same normalized data used for training the neural network was also used to train the agent based on the Mixture of Experts (MOE) (see appendix D). As with the neural network, its basic structure was based on the DFA, but certain behavior was overridden when extra information from the MOE was available. Like the neural network, the *search* state and *shooting* sub-state were enhanced using predictions from the MOE. The individual experts also had the benefit of being usable in the overall fight state. As with the neural network, the MOE algorithm was trained off-line, but consulted on-line in real-time.

Like the neural network agent, the MOE went through a number of steps in an off-line training process. First, all data went through a clustering process design to group similar behavior. A hybrid, Kohonen Self Organizing Map (SOM)/ K-means algorithm was used to cluster the data as the first step for creating the experts. While a typical SOM is an excellent way to represent high dimensional data on a two-dimensional map, it often does not produce distinct clusters needed for this procedure. The smoothing procedure, however, produces excellent results and has the added benefit of keeping similar clusters close together. Since a SOM can be implemented as a smooth K-means clusterer, the two were combined, taking properties of each.

A one-dimensional grid of ten nodes was used in order to create well-defined end clusters. Although a growing (self-sizing) SOM may have been more effective, a static size was deemed sufficient to determine the feasibility of the idea itself. Source code was left sufficiently modular to allow for easy substitution if warranted in future work. Smoothing was done in several steps as if the structure were a one-dimensional SOM that produced more natural and gradual movement of

data from one cluster to another. As is often done with both algorithms, a random central point (with the same dimensionality as the sample data) was assigned to each node to seed the start of the process. Distance from this central point was used as the relative measure of fit for each data sample, and data were assigned to the node (I) with the least distance.

$$\min_i (|x - w_i|)$$

Distance was measured as the sum of all the differences between the node's central point (w), and the current data (x). Since data were pre-normalized, equal weight was given to all dimensions of the data. The process of assigning all data to its closest cluster was considered one iteration of the algorithm. After each iteration, each cluster would re-adjust its central point to reflect the central tendency of all samples within that cluster. Each dimension within the central point became the mean of the data for the dimension.

Fifty-thousand iterations were completed on the data, with gradually decreasing size in the number of adjacent nodes used for the smoothing effect. The smoothing factor determined how many clusters' means would be influenced by a particular data point. For example, at the start of the algorithm, a data point would not only influence the new mean of a cluster to which it was assigned, but also the N nearest nodes in either direction of the one-dimensional map. Throughout the fifty-thousand iterations, this smoothing factor started out as five (five nodes on either side, if applicable) and gradually reduced to zero after every ten-thousand iterations. This meant that by the end of the iterations, data would only influence the node to which it was assigned. These clusters were then fed into the neural net training algorithm described above in order to produce

the Mixture of Experts (separate neural nets) instead of a single, large neural network.

As an added benefit to this separation, each network was able to choose parameters different from those that optimized the large, overall network. As with the neural network above, trial and error was used to determine optimal values for parameters such as the learning rate, the number of hidden layers and the number of nodes per hidden layer. Table 1 lists the optimal values for each expert compared to the same values for the main neural network (Expert zero is omitted since the clustering algorithm assigned no data to this node).

The gating mechanism for the mixture of experts was a simple recency algorithm that looked at the opponent's recent actions and chose the expert that most closely resembled that behavior. Fifty frames (one and two-thirds seconds) was chosen due to accuracy results from neural net predictions (see results section). Although an additional neural net is often trained to serve as a gating network with this algorithm, the lack of 'correct' response for training back-propagation made the recency algorithm the more straight-forward alternative.

The MOE was used completely different prediction in the search state and the firing sub-state than did the neural net. Instead of a single network predicting these locations, the MOE brain chose the expert that most closely resembled the opponent's most recent (fifty, see above) actions, and used that as the predictive network. As with all discussed algorithms, only data observed through the agent's sensors was used in these determinations.

In addition to these modifications, the MOE agent also had the unique opportunity to use these experts to choose its own movements. All information was saved and available with these experts including the modal movement for each expert in terms of the effector keys most often

pressed to produce that expert. These keys could then be used as the MOE agent's own choices when they would produce the most desired move. Since directed search was already optimized, and undirected search would benefit little from this idea, It was determined to incorporate this idea into the overall fight state of this agent. Not only was the opponent's position predicted using these experts, but all experts and DFA states were polled prior to choosing movement keys and the solution that most closely resembled the desired new position was chosen as the winning output. It should be noted that this was not possible for the basic neural net agent since there would only be one 'modal' move for the agent since all data were used in the training of that network. Since the modal move was rarely a unanimous winner for any of the experts, an output key sequence was chosen weighted by the frequency that a particular key sequence produced that expert.

	Hidden layers	Nodes per hidden layer	Iterations req'd	Net Accuracy (training)	Net Accuracy (testing)
Neural Net	2	10	10000	92.39%	90.80%
Expert1	2	10	3300	94.07%	92.03%
Expert2	2	15	4100	92.86%	88.98%
Expert3	2	8	2900	95.00%	86.23%
Expert4	2	8	6800	92.52%	89.78%
Expert5	2	8	3000	91.71%	85.71%
Expert6	2	8	8100	90.35%	89.66%
Expert7	2	8	5700	95.10%	92.17%

Expert8	2	8	6100	92.53%	88.34%
Expert9	2	15	9000	91.82%	84.12%

Table 1. Training statistics for individual experts

Recursive Modeling

All intelligent computer agents were capable of simulating some form of opponent modeling, and using that model to determine its output. Although some of these models were developed offline (e.g. neural net, MOE), the creation and use of these models was restricted to information that the agent would have access to in the course of a match (from its sensors). This modeling could assume that the opponent used a decision structure identical to its own (has the same brain), such as the DFA, or it may develop a model based on data that was observed in previous matches and constructed off-line, such as the neural net or MOE.

Each of the three different computer algorithms tested were implemented with three different levels of recursive modeling. The recursive component of the opponent model tried to incorporate different levels of information about the match. Level-zero recursion was the base level and, in effect, did not model the opponent at all. The only information that was used to determine the agent's actions were its own state and the state of the arena. Although the opponent's current position was used in these calculations, its current state (intent) was not. Level-one recursion was the first level at which true opponent modeling took place. The level-one agent considered its own state, the state of the arena, as well as the predicted state of its opponent. It attempted to

incorporate what it thought its opponent would do into its own actions. At level-two recursion, all information from the previous levels was used plus the agent attempted to determine what its opponent thought the agent will do.

A good example of this recursion in action was the align sub-state within the fight state. A level-zero recursive agent would only consider where its opponent was at the moment of trying to align for a shot. A level-one agent would attempt to determine what the opponent is doing and align to where the opponent was likely to be when the bullet arrived. A level-two recursive agent would also shoot where the opponent was likely to be, but it also included how the opponent was likely to react to what the original agent was doing in this calculation. It is important to note, however, that any model created in this recursive process is not itself recursive (agent's assume that their opponent's model is not recursive). This assumption was necessary to prevent infinite recursion from taking place.

Although the recursive process was the same for all agent types, information that the agent was capable of including in this recursion was different across computer algorithm. The DFA, being the simplest of the algorithms was also the simplest recursive model. As with MacInnes (1999), the agent simply assumed that its opponent was a DFA with the exact same rules and states as itself. Recursion simply consisted of literally 'putting itself in its opponent's shoes' and deciding what it would do in its opponent's place. Although this simplistic model was capable of using recursion on any of its states, it was only productive to do so in the fight state. Since the search state did not use opponent information (search was blind), creating a model of that opponent would have served no purpose.

The neural network placed similar restrictions on the states in which recursion was conducted, but for different reasons. The fight state remained recursive, much the same as the DFA, but with the neural net predicting opponent position instead of assuming the opponent was a DFA at the entry recursive level. If the fight state recursed beyond this first level, all further levels (up to the maximum) were assumed to be simple DFAs. Neural net prediction was not used after this point due to a limit on the information available to train a proper neural network. If the neural network agent were to create a neural net recursive model, it would need nearly a full second of observation before being able to produce a single usable estimate. Second-order recursion would become even more difficult, needing a full second of the agent observing its opponent observing the agent. The search state was also denied recursion even though a model of the opponent was now used in the directed search sub-state. Although search was no longer blind in the directed version, the agent could not use an estimate of where its opponent thinks it is when the opponent could not see the agent. The neural net could only legitimately use test data from when it could *see its opponent seeing the agent*, and this would be fight state, not search. The neural net could be set with an educated guess, but a guess of the input would hardly be different than an educated guess of the output of the network.

The MOE algorithm offered the most flexibility as far as recursive modeling. Since the experts themselves were very comparable to the DFA states, they could often be used and interchanged. Since the experts and the DFA states had well defined effectors associated with them, the MOE could often poll both DFA states and MOE experts to determine which one produced the closest to the desired result. This also allowed the MOE the use of these stereotypes

as different strategies for its own moves, which was not possible in other algorithms. In order to prevent the agent from dramatically flipping between states and experts thirty times per second, a constantly increasing likelihood of shifting was adopted. An agent would only consider changing its expert prediction at a one percent cumulative chance every frame (i.e, thirty percent after one full second).

Experiment Design

Four experiments in total will be discussed, although only data from the last two will be of primary importance. The first experiment was a review of DFA agent research discussed in detail in MacInnes (1999). A review is included here since it provided a foundation for some of the following experiments and provided a baseline performance for both human and computer agents. The second experiment was the first involving human agents, but in addition to providing data for analysis (mostly on SOM effectiveness), its main purpose was to provide training data for the algorithms used in other experiments. The final experiments tested all combinations of human and computer agents and provided a majority of the user modeling data discussed in this dissertation. Single sessions with a variety of subject pairs as well as an in-depth multi-session test with a single pair of subjects were conducted. Experiments will hereafter be referred to as E0-a (MacInnes, 1999), E0-b (training data), E1 (single session) and E2 (multi-session).

Experiment zero (a) consisted of duels between two agents (DFAs) in random virtual environments of the same dimensions as the Hebb-Williams mazes. DFAs differed only in the level with which they recursively modeled their opponent, varying from zero to three. Matches were

fought as the best out of fifteen wins, with all other details identical to current experiments.

Experiment zero (b), although tested against human agents, was not designed to gather user modeling data. Since two of the three algorithms used in this research required a large pool of observations with which the algorithms were to be trained, the previous version of the arena was modified as a data gathering tool. Human/DFA match capability was added, and the resulting code was released on the web available for download. A request was attached to the site and forwarded to various departments at the University for volunteers to download the program, and return their output files. A score board was attached to the web site to try to increase enjoyment of the experiment and to try increase the data return rate. Although win/loss performance data were collected by the program, it was decided not to include it in the data from experiments one and two due to the informal and varied, data collection methods.

Mazes one through four of the twelve Hebb-Williams mazes were used as the arenas. Each subject fought once in each arena in a match that was the best out of fifteen kills. Since performance data were not important, only the recursive level zero DFA was used as the computer opponent in all matches. Observation and positional data from both the DFA and the human agent were saved throughout the duration of every match and each arena. The data were output to a special directory, which the participants were asked to return to the experimenter (via email, or some other such method). Only data that were returned complete and intact were included as observation data for training algorithms. Eight subjects contributed data over the period of one and a half weeks.

Twenty volunteers (eleven men and nine women) from the university and general population

participated in Experiment one. Two computers were placed in separate but adjoining rooms, to prevent potential participant interaction outside of the virtual arena. Computers were networked together over a standard one-hundred megabit (100Mb) TCP connection creating an independent and isolated, Local Area Network (LAN). Neither machine nor the LAN were connected to the university network or the internet. Computers were identical Pentium IV machines with the following specifications: 2.4 GHz Pentium IV, one gigabyte of double-data rate random access memory (DDR RAM), ATI Radeon™ 9000 pro video card with 128 megabytes of DDR video RAM, 10/100 ethernet card, and a seventeen inch Viewsonic™ (VG700) monitor running 1024x768 resolution. Components were chosen to maximize the likelihood of maintaining a consistent thirty frames per second on both machines for the duration of the experiment. The network card was the highest speed available at the university at the time of testing, the graphic card was tested to be capable of rendering the arena at one thousand frames per second (when not synchronized to the monitor refresh), and the monitor had a maximum frame rate of eighty-five Hertz at the specified resolution.

Each participant played in a total of ten matches, where the objective of a match was to maximize the number of times you killed your opponent while minimizing the number of times you were killed. Each match began by placing the two combatants in random locations in the arena. Combatants were instructed to search out and shoot their opponent as best they could using the control keys (effectors) mentioned above. A match ended when the total kills for the two combatants reached five. The ten matches in the experiment contained a single match against each of the nine possible computer agents (three algorithms by three recursive levels) and one match

against the human opponent in the other room. Participants were unaware of the percentage of matches which would be human and computer. Matches were pseudo-randomized in a way that mixed the sequence of matches, but still allowed the same sequence for the two concurrent combatants (they both needed a human opponent at the same time). At the start of the experiment, each subject was asked to choose a number between one and one-hundred. These two numbers were then multiplied together, and the result was entered into an input file and used as the seed for the C++ random number generator found in 'math.h'. Given the same initial seed, the C++ random number generator will produce the same random sequence for the two participants regardless of machine or system clock. The ten matches lasted between twenty minutes and one-half hour, depending on the speed of the combatants.

All matches were synchronized between the two computers, so they would start at the same time regardless of whether the opponent was a local computer agent or a TCP agent. This was to ensure that delay between matches would not give information about the coming TCP opponent. This did introduce information about the match previously played, but this was less problematic than information of future opponents. A long delay after a match tended to signify that the previous match was not a human opponent since they likely continued playing after the local match was ended. This information was not completely accurate, however, since players did spend differing amounts of time answering between match questions, and since each player had to confirm that they were ready to continue with a new match, it was possible that the previous human opponent was merely taking a longer break.

Subjects were asked additional information about themselves, the individual matches and

the overall experiment in addition to the random number mentioned above. Since participants were volunteers, and often volunteered in pairs, each pair of participants were asked at the start of the experiment whether they knew each other and whether they had played any type of network computer game together. No subjects were excluded due to this criteria but it was saved to be included as part of the analysis. Other subject information included their initials, as well as a self report on their relative computer game experience. Each participant was asked to rate this experience on a scale of one to nine with one being never played computer games before and nine being play computer games daily. During an intermission between each match, participants were also asked to decide whether they believed their opponent was controlled by a computer, or the human in the other room. A confidence score from one to five was also requested with five being the most confident and one being the least. At the end of the entire experiment, participants were asked to give their subjective impressions about the experiment. For example, the traits that they used to distinguish between humans and computers, strategies they used to fight in the matches, and how many opponents did they think were human and computer. Participants were told on the last question that it did not necessarily have to match the total of response numbers given in the post-match questions if their opinions changed. Most questions were asked and recorded by the computer through the course of the experiment. The random number and the post-experiment discussion were the only questions asked and recorded by the experimenter.

Experiment two was identical to experiment one except for the participants and the duration. In order to test the computer algorithms under expert scrutiny, two participants were recruited to play in an extended version. Participants were selected who were familiar with the

controls of the arena, (but not these particular computer agents), who knew each other and had played network games together, and who (in the experimenter's opinion) were likely to rate themselves as seven or higher out of nine on the computer game experience scale. The two participants chosen were known to the experimenter, had both play-tested previous versions of the arena, had played each other numerous times in computer games and rated themselves at eight and nine on the nine point gaming scale. Although this selection process introduced obvious experimenter bias into this experiment, it was deemed necessary in order to procure combatants that would test the agents' effectiveness and believability under these conditions. These participants were tested in the exact same process as mentioned for experiment one, except the experiment was repeated three times over a period of two hours.

Chapter 5: Results and Discussion

Overview and Descriptive Statistics

A total of twenty subjects (ten male, ten female) participated two at a time in the experiment. Two subjects (both male), participated in an extended version (E2) which looked at patterns over three sequential runs of the experiment. The mean self reported experience level for all participants was 4.11 on the one to nine scale. As expected, this differed across gender with men having a mean of 5.75 and women at 2.80. Alpha (significance value) was chosen at 0.05, although marginal values will be reported to indicate potential for future research.

Looking at each of the subject pairs, only two reported that they did not know their opponent and fourteen reported never having played network games (*LAN*) with their opponent. Both knowledge and LAN background was a requirement for the pair in Experiment two. The high degree of familiarity between subjects was likely due to a number of factors. Subjects were selected from two science faculties (Computer Science and Psychology) in order to maximize the possibility of getting the best distribution of computer and gaming experience. Since experience and gender were the focus of the study from the human perspective, knowledge was a less critical variable. Since no range was given for prior knowledge of an opponent, this likely covered possibilities from friend, to mere passing familiarity. Finally, the need for two subjects to volunteer simultaneously, may have placed scheduling restrictions which increased the likelihood of familiarity in the subjects. The potential bias is deemed to be slight however, due to the likely range of prior knowledge. This increased knowledge also reflects the usual applications of these multi-agent systems where potential opponents are usually more than passingly familiar.

Accuracy

Before the three dependent variables were tested, it was decided to determine the benefits, if any, of the Kohonen SOM that fed into the mixture of experts. Since the purpose of this step was to create individual experts that were to model user stereotypes, it would only be effective if the newly created sub-groups were better models than the overall data (these SOM graphs are descriptive results most closely associated with E0b - training data). All data referred to within the results section represents data that has been normalized to values ranging from zero to one in real values. This was performed to allow for easier transition between the many algorithms used at each stage of data processing and machine learning (The SOM in particular required normalization for

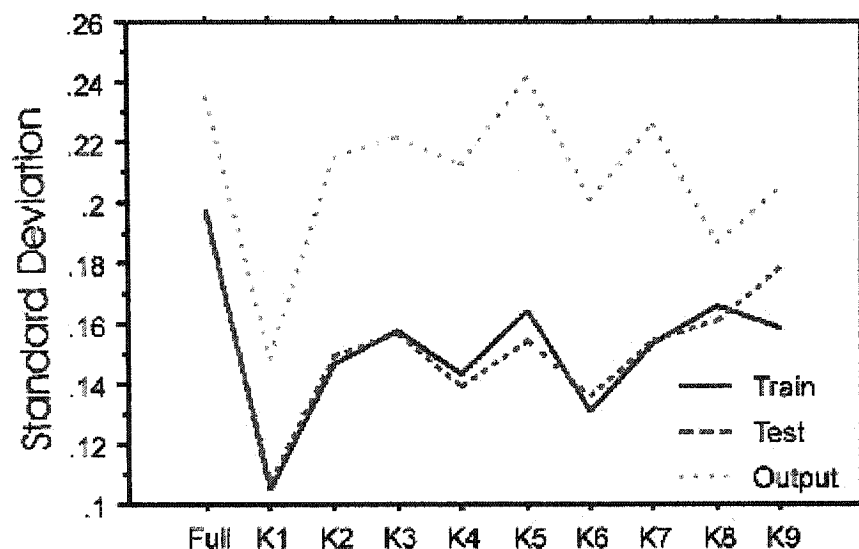


Figure 7 standard deviations of samples within the overall data (Full) compared to the Kohonen clusters (K1-9). Graphs show that the clustering was effective on the experiment 0b training data (train), as well as testing those clusters on the previously unseen data from E1 (test). Since the clusters were created only using the data from the input layer of the neural net, it was decided to test on the output layer as well.

the distance measure to equate across attributes).

To test this effectiveness, the average standard deviation of each resulting group was calculated and compared to the overall data. The standard deviation was first calculated for each of the attributes of the data, then averaged across all attributes that were to be used as input for the neural networks. Figure 7 shows the results of these calculations for the training data, the output layer of the training data as well as the resulting data from experiments one and two.

First, as seen in the standard deviations of the data used to train the Kohonen SOM, clustering the data had a definite effect on the expert groups. As expected with a clustering algorithm that minimizes distance as its similarity measure, the standard deviations of the groups are consistently less than the standard deviations of the whole. If nothing else, this is evidence supporting the fact that the hybrid Kohonen SOM/ K-Means was effective as a clustering algorithm.

Since the final neural networks would use this input data to predict the output data, it was decided to see if the clustering (which was performed on the input attributes only) would have any effect on the standard deviations of the output layer. If the clustering on the input layer alone had a similar positive effect on the standard deviations of the output layer, it would be possible to show early evidence of a predictive link between what would be the input and output layers of the neural networks. Using the same clusters that were created with the input, Figure 7 shows that there is also some improvement on the output data as well. Although the overall standard deviation and the improvement of the clusters is less than for the input layer, it is still less than the standard deviation of random samples (.28) within this normalized range (0.0 to 1.0).

The end result of this clustering of course, would be to use these clusters to test new data with the MOE agent. The Kohonen Mean of each cluster would be used to sort the new data to determine which expert (stereotype) would be the closest fit. In an effort to determine the benefits of this strategy, the data from experiments one and two were run through the SOM using only the means from the training clusters (The clusters and means were not retrained with the new data, only tested). The same standard deviations were then calculated for the resulting clusters and the results are shown in Figure 7 as the test data. These new clusters did show improvements in the stereotypes equal to those seen in the training data, suggesting that these clusters were just as accurate in grouping new data from as yet unseen people. This did not prove that all strategies and stereotypes were accounted for within these groups, but it did suggest the strategies that were represented were more closely approximated with the experts than with the overall data.

Immediately following the clustering process, feed forward, back propagation neural nets were trained for the neural net and MOE algorithms. The data, in its entirety, was used to train and test the neural net algorithm and each of the SOM clusters were used to train and test neural networks to be used as an expert for the MOE. As with other implementations of MOE algorithms (Jacobs, 1991), it was not necessarily expected that the MOE would outperform the neural network. Since, in theory, a sufficiently complex neural network can model any pattern, dividing the problem into simpler patterns should only serve to reduce the complexity of the neural network needed.

Table 1 (page 60) shows the training accuracies of the neural network and MOE experts. The data from the overall network as well as each of the SOM clusters were divided up into two

groups - ninety percent of the samples were used to train the neural net, while the remaining ten percent were used to test the resulting network. Resulting accuracies for these two sets are found in Table 1 in the training and testing columns respectively. All neural networks were trained until there was a reversal in the test accuracy or no significant improvement in the training data over five hundred training iterations. The rule for reversals was included to prevent overtraining of the network which tends to improve the accuracy in the training data at the cost of accuracy on future, untested data. These results show that while the smaller, specialized experts are not more accurate than the overall network, many require the same or less complex internal structures and fewer training iterations needed to reach optimal performance. Two-thirds of the experts required fewer nodes per hidden layer than the full neural net to reach optimal accuracy while only two required more. Every expert also required fewer training iterations to reach peak accuracy, and this result was compounded when one considers that the experts had, on average, one-ninth of the training samples per iteration. The column labeled 'total training samples' reflects the number of samples used in training that expert (from the SOM) times the number of iterations used to train the network.

Accuracy looked at each of the algorithm's ability to predict an opponent's next location in the arena given its five most recent locations(except for the DFA, which by definition only uses current information). Since variables such as *gender* and subject *experience* were not available in experiment 0a and 0b (training experiments), only data from experiments one and two were used for this analysis. Data were analyzed using a ten (*time*) by three (*algorithm*) repeated measures ANalysis Of VAriance (ANOVA) with *gender* and *experience* as between subject variables .

Time refers to how far in the future the algorithm was tested, and was counted as how many samples of one-sixth of a second from the current time. Experience was the self rated measure of how often a subject played video games and was condensed from the original scores as follows: one to three = low; four to six = medium; and seven to nine = high. Algorithm was the prediction method including DFA, neural net and MOE. Gender, of course, was the sex of the opponent being modeled. Analysis which only look at the MOE were also able to look at the differences between the nine experts as a variable.

The ANOVA stated above resulted in every variable and interaction being highly significant ($P < 0.001$) including the four-way interaction between *experience*, *gender*, *time* and *algorithm*. Although large data sets themselves cannot create significant results, the over two-hundred-thousand data points were more than sufficient to show every significant pattern within the data.

ANOVA Table for Time

	DF	Sum of Squares	Mean Square	F-Value	P-Value	Lambda	Power
Gender(F)	1	249.393	249.393	41.530	<.0001	41.530	1.000
NewExp	2	248.081	124.041	20.656	<.0001	41.312	1.000
Alg	2	1229412.402	614706.201	102364.275	<.0001	204728.551	1.000
Gender(F) * NewExp	2	416.270	208.135	34.660	<.0001	69.320	1.000
Gender(F) * Alg	2	2431.918	1215.959	202.488	<.0001	404.976	1.000
NewExp * Alg	4	473.333	118.333	19.706	<.0001	78.822	1.000
Gender(F) * NewExp * Alg	4	753.308	188.327	31.361	<.0001	125.445	1.000
Subject(Group)	217935	1308718.257	6.005				
Time	9	3336968.203	370774.245	952916.410	<.0001	8576247.692	1.000
Time * Gender(F)	9	173.432	19.270	49.526	<.0001	445.733	1.000
Time * NewExp	18	176.777	9.821	25.241	<.0001	454.329	1.000
Time * Alg	18	513828.844	28546.047	73365.389	<.0001	1320576.993	1.000
Time * Gender(F) * NewExp	18	233.216	12.956	33.299	<.0001	599.382	1.000
Time * Gender(F) * Alg	18	1093.689	60.761	156.159	<.0001	2810.860	1.000
Time * NewExp * Alg	36	352.306	9.786	25.151	<.0001	905.452	1.000
Time * Gender(F) * NewExp * Alg	36	468.702	13.020	33.461	<.0001	1204.599	1.000
Time * Subject(Group)	1961415	763175.193	.389				

Figure 8 ANOVA of accuracy results for *time* (prediction), *algorithm*, *gender* and *experience*.

Due to the number of significant effects, the entire ANOVA table is included as figure 8 instead of listing all possible interactions. Graphs and descriptions of the base effects as well as some of the more interesting interactions are discussed below.

Figure 9 shows the result of *accuracy* (Y axis) by *time* (x axis) split by both *algorithm* and *gender*. As to be expected, the accuracy decreased for all conditions the further in time that the algorithm tried to predict. This effect of time was robust and consistent, and will only be mentioned further in how it interacts with other variables. Overall, the neural network algorithm was superior to the MOE and DFA in its ability to predict future opponent positions with the MOE coming in a clear second.

The gender differences seen on the graph reflect the interaction of time, algorithm and

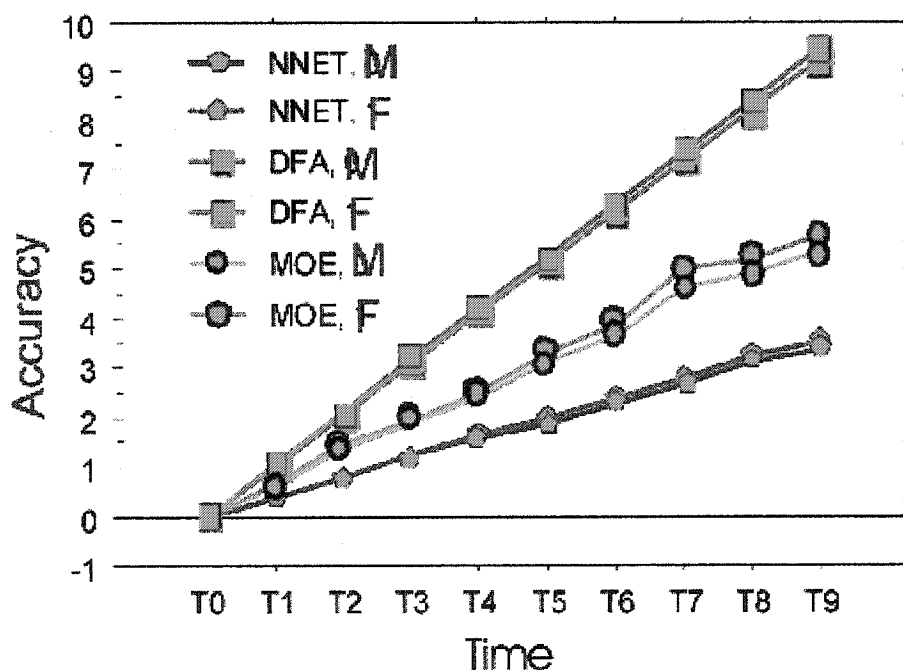


Figure 9 Accuracies (as Euclidean error) of the three different algorithms across ten prediction samples. Data further split by gender to demonstrate the algorithms' differing abilities to predict the strategies of the two genders.

gender. Although not as large as the effects between algorithms, the gender effects were significant and showed that different algorithms were better at predicting future location for different genders. While the MOE and the neural net were more accurate for the men, DFA was more accurate for the women. This data, when seen from the perspective of theories on navigation strategies, offers converging evidence of different algorithms being more effective against the different strategies that are attributed to men and women (landmark vs. spatial strategy).

Further evidence for this distinction was found within the MOE itself. Figure 10 shows the time sequence data for the MOE only, divided by individual expert. The experts were fairly evenly split between accurate (three, five, seven and eight), inaccurate (two, four, six and nine) and one erratic (one) that may have been caused by predicting direction reversals. Looking at the distributions of gender within the original SOM clusters (table 2), it seems that the clustering

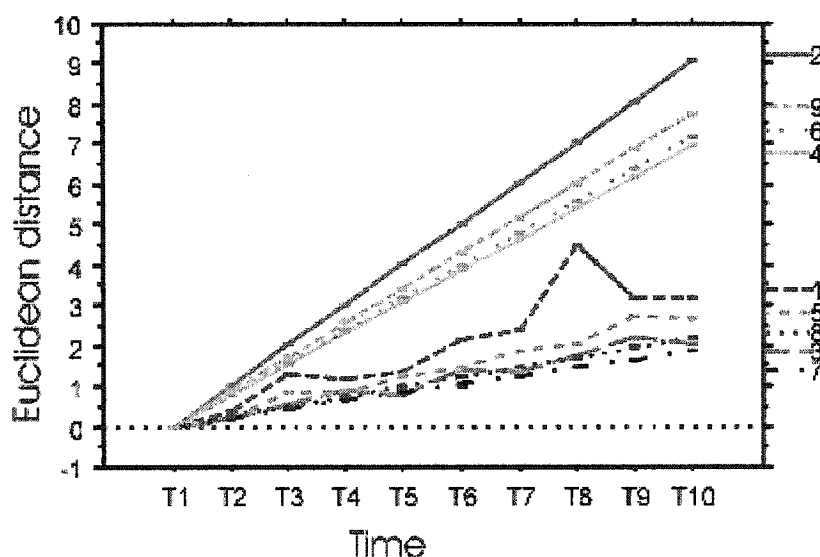


Figure 10 Accuracy of prediction for the nine MOE experts (stereotypes). Note that only the MOE allowed for this type of split analysis

algorithm has allocated the samples based in part on gender even though it was not one of the factors

Samples	M	F	M%	F%
725890	444190	281700	61%	39%
193330	110040	83290	57%	43%
168890	97560	71330	58%	42%
54550	39830	14720	73%	27%
44370	32940	11430	74%	26%
35690	28460	7230	80%	20%
149200	75180	74020	50%	50%
61510	45290	16220	74%	26%
6180	4630	1550	75%	25%
12170	10260	1910	84%	16%

Table 2 Number of training samples split by gender. The top row represents the split in the overall data, while the remaining rows are for the MOE experts.

involved in training. Clusters one, two and six had higher portions of samples from women than that of the average, while the others have less. Since the MOE is less accurate at modeling women, it is no surprise that the experts from these three clusters were two of the lowest accuracies and the one with the most variability.

While the same pattern existed for the neural net and the opposite held true for the DFA (better at predicting women's location), neither algorithm was open to the same type of divided analysis as shown with the experts of the MOE. Since the stereotypes represented by the MOE were designed to reflect different strategies, it could be beneficial for future versions of this modeling algorithm to include gender in the clustering and training of the experts when the information is available.

Looking at the accuracy data again, but this time with *gender*, *experience*, and *algorithm* as the variables (figure 11), we begin to see some of the interaction between gender and experience. As seen in the ANOVA table, experience was a significant factor in accuracy prediction, but the effect varied across all conditions. Once again, the neural net and the DFA were nearly mirror images of each other. The DFA was better at predicting women's performance and this result got worse with more experienced opponents. The neural net however, performed better against the men, and this result got better against the more experienced opponents. For these

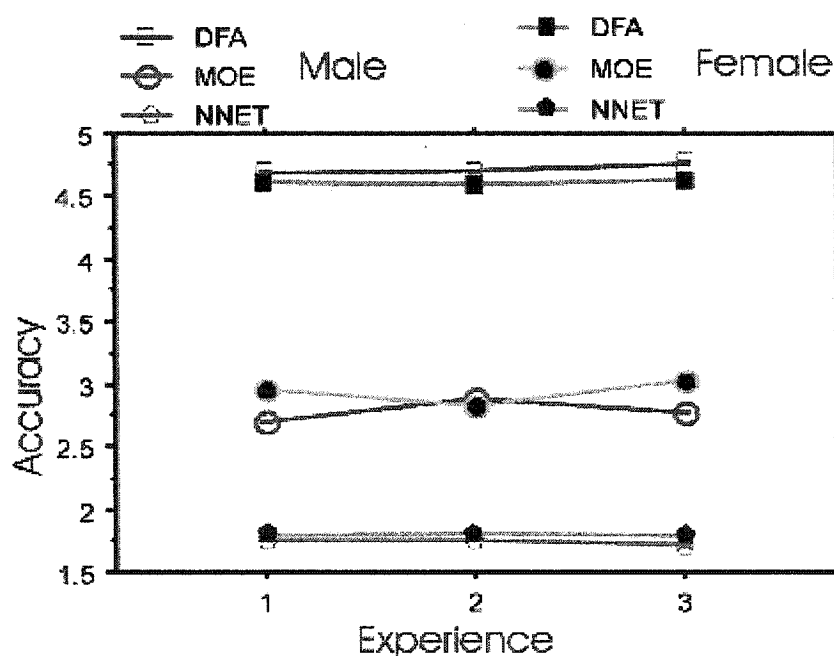


Figure 11 accuracy score by the experience of the participant. Split by both gender and algorithm.

two algorithms at least, the effect of experience seemed to be consistent across gender -i.e. no gender versus experience interaction within these two algorithms.

The MOE, however, showed very mixed results and clear interaction between gender and

experience (The fact that only one algorithm showed this interaction was indicative of the three-way *gender x experience x algorithm* interaction). Overall, the MOE was better at predicting patterns with male opponents and was also better against those with higher experience. The exception in both cases was a reversal with participants who claimed mid-range game experience.

It is unclear what to make of this mid-range difference. Since the statistics were highly significant, we will accept the results as accurately reflecting the truth, but any explanation will likely require further testing. One possibility is that subjects who rated themselves in the medium range of game experience, were more varied than those on the extremes. It is important to remember that experience was a self report variable and could have been influenced by the fact that it was subjective. Although data was anonymous, frequent players may have seen the need to downplay their computer time (possibly out of habit), and low end gamers may have seen the need to

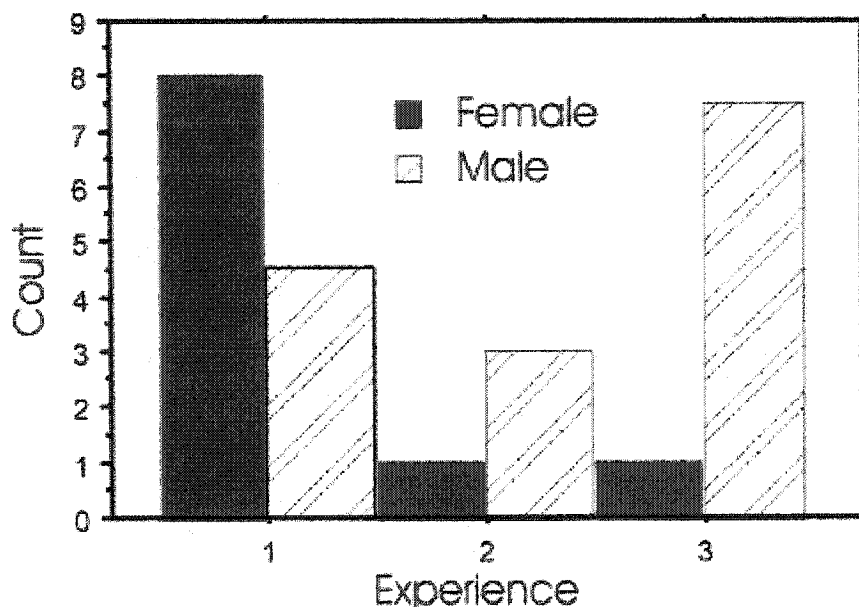


Figure 12 Histogram of gender by experience. Although experience is later transformed to a more appropriate bi-level variable, it is left here as three to show the non-normal distribution.

exaggerate. Anecdotal evidence supports this. The case of two participants, for example, who rated themselves as low experience, but also reported having played each other in competitive network matches on a number of occasions. If problems in self report were causing bias, we would have expected to see a higher variance in the mid group than at the extremes, but we in fact saw the opposite trend: lower variance in the mid group.

One assumption that may not have been valid, was that experience was a linear scale with a normal distribution. Taking a look at histograms for the distribution in men and women (figure 12). It was apparent that this bore further scrutiny. Male experience was clearly bimodal, and women's scores were skewed toward low. Mid-range scores in fact, accounted for less than fifteen percent of all participants tested. The data shown were likely to be a accurate sample of the real world distribution, since it matched data taken from Shore et al. (2001) and MacInnes (2001). Differences include fewer men with low gaming experience on both studies (skewed toward high gaming experience), and the Shore et al. study had no female participants with high gaming experience at all (The MacInnes study, as with this research, had one).

Through the lack of a bias, and similarities with previous studies (Shore, 2001), it would seem that this data was an accurate representation of the population distribution and reflected the established *preference* for spatial tasks (games) among men (the relation between this preference and spatial ability will be discussed later) The unusual data among opponents with mid-range experience, then, was likely an artifact of two populations with very different gaming background before the experiment and will add to the justification for transforming experience into a binary variable.

Effectiveness Score

As mentioned previously the effectiveness of an agent was defined by the number of times it was able to defeat its opponent in any best-out-of-five match and the value ranged from zero to five. Although the accuracy of an algorithm influenced this score (since a more accurate predictor should be more effective), all three algorithms were able to use this information in different ways. For example the DFA, by definition, was unable to use information from previous states in its current decisions while both other algorithms could. Furthermore, the MOE had subdivided stereotypes that could be (and were) used for its own moves (in addition to prediction), which the neural net did not. It is worth noting in the following analysis that a 'good' score depended on the perspective of a particular analysis. If we were determining which algorithm was better, a lower score represented an improvement since that was the score against that agent. While analyzing experience however, a higher score was better since the analysis switched to the human perspective against that agent.

Experiment 1

The analysis of the effectiveness score was split into three stages. First, the between subject variables were subjected to a factorial ANOVA; this was followed by two repeated measure ANOVAs of the within subject (and relevant between subject data) variables and finally a look at a few graphs of interactions between these groups. This division was necessitated by the makeup of the data. Although the repeated measures ANOVA was the appropriate final analysis

for this experimental design, it precluded the inclusion of a few between subject variables. Since the repeated analysis treated each subject as a single data point, variables that dealt with the individual matches were lost.

Due to the large number of between subject variables and the scarcity of some of the combinations, a full analysis of all variables was not possible. Even though most variables were balanced at the base level, certain combinations of between subject data were not found. Usually this would be a case for collecting more data to add to the study, but this would be unlikely to help. One case in example, was women who rated themselves as 'high' on computer gaming experience. This combination was only found with one subject in all four of these experiments. A review of other similar study's using this same graphics engine however (Shore et al., 2001; MacInnes (1999); plus other unpublished data) combine to over one hundred participants but only two in this category, so even a large number of extra (random) subjects would be unlikely to increase this category. To minimize this problem a number of variables were collapsed to fewer, but equally relevant factors.

The first consideration for the factorial analysis was experience and was actually represented by a number of different factors. The prior computer gaming *Experience* was reported by each participant and was rated for the analysis as high or low (one-four, and five-nine on the self report scale respectively). This transform kept a strong reflection of the original variable as tested by regression ($R=0.90$). Since participant's performance may improve as they duel with the ten different opponents, the order that they experience the different algorithms was included (hereafter referred to as 'Practice'). Finally, since experiment two looked at longer term effects of

running in the experiment many times against the same human opponent, the ‘Sequence’ of the experiments was also analyzed. (Note that ‘Practice’ refers to matches within an experiment and ‘Sequence’ refers to between experiments).

The time to complete a match (‘Duration’) measured the length of time in milliseconds that it took for a subject to fight each opponent (all five kills). This variable required further transformations to be used in a factorial ANOVA; to convert it to a discrete variable, and also to

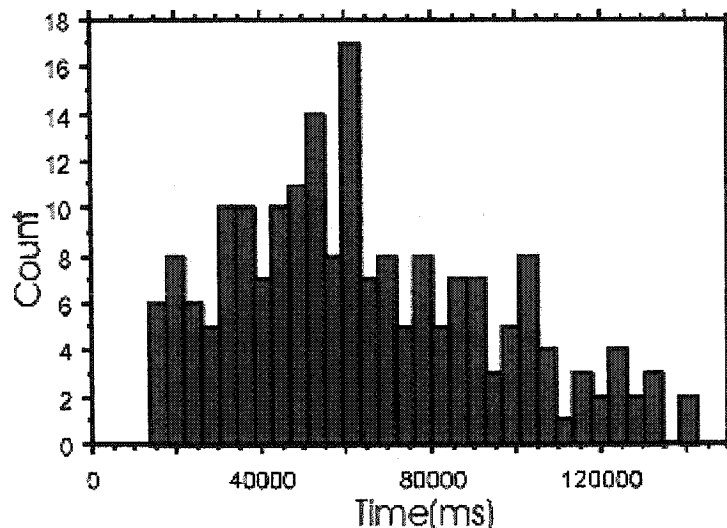


Figure 13 histogram of match completion time.

deal with the skewed distribution (see figure 13). Time was first transformed into a Z score reflecting the number of standard deviations a particular time was distant from the mean. This was only partially effective since a few scores were more than two standard deviations greater than the mean, but a Z of negative two was not possible due to the minimum boundary of zero. It was decided to convert the time to a binary variable split evenly about the mean time to ensure fair distribution and minimize the likelihood of empty cells for interactions. To ensure that this variable

was still an adequate representation of the original variable, a regression of the two measures was performed. The new variable still held a strong reflection of the original with $R = 0.798$.

The effectiveness (factorial) ANOVA of *gender x experience x practice x duration* showed a significant effect of gender x duration ($F(1,249)=4.98, p<0.03$) and a marginal effect of gender x experience ($F(1,249)=3.01, p<0.083$). Again, limited group sizes did not allow for testing interactions on the full set but interactions of up to depth two were tested. Gender and experience were both legitimate in the repeated measures analysis, so they will be discussed in that more appropriate forum.

The effect of gender by duration, shown in figure 14, demonstrates the fact that men tended to do better at longer matches, while women tended to do better at shorter matches. These differences were also reflected in the histograms for the two genders (figure 15). While men were evenly distributed, women tended to finish much faster than the men. This could easily have been

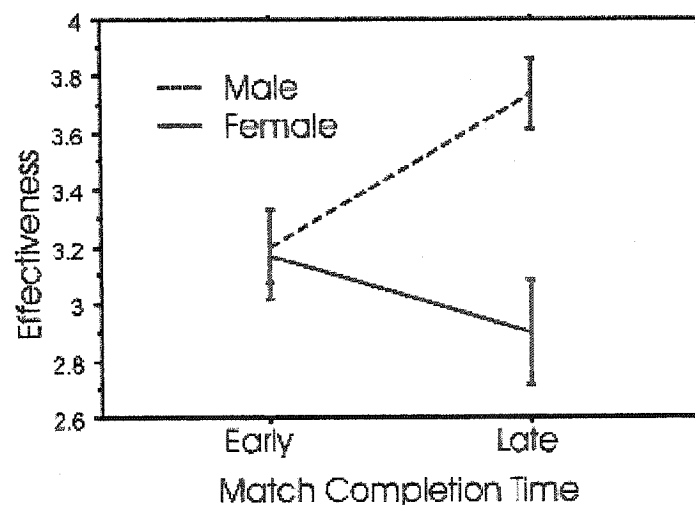


Figure 14 Effectiveness of participants based on how early a match was finished. Data split by gender shows women doing better the earlier they complete a match while men tend to do better in longer matches.

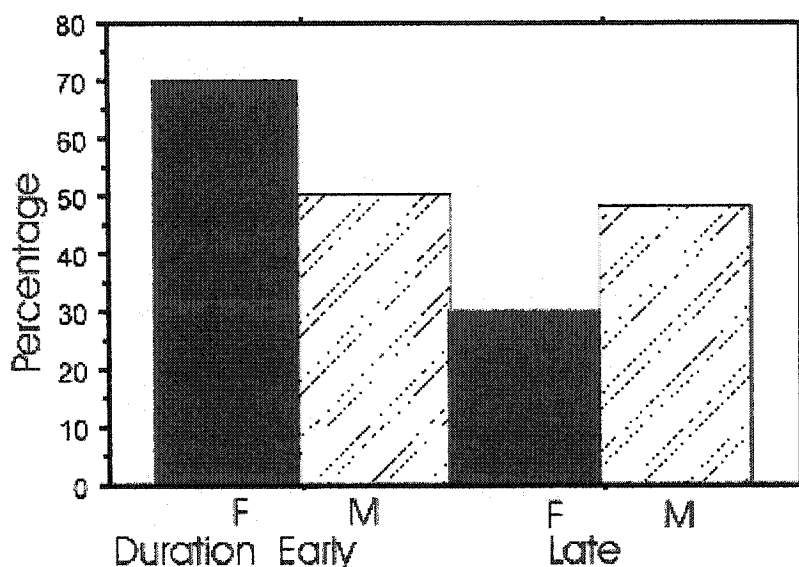


Figure 15 percentage of matches that finished early and late split by gender. Since men and women did equally well, this is further evidence for strategy differences.

indicative that differing strategies were being employed by men and women in this arena.

Effectiveness against opponent was analyzed with a repeated measures ANOVA as both a ten factor (opponent) and a three x three (algorithm x recursion). The first analysis treated each of the nine computer opponents as its own factor and allowed for comparison with human opponents, and the second treated algorithm and recursion as separate variables, but did not allow comparison with humans, since recursive modeling could not be measured (if it is being done) in human opponents. It was with this second analysis that the full between/within ANOVA would be shown.

In the ten factor analysis, including all computer and human opponents, the type of opponent was marginally significant ($F(171,9)=1.877$, $P<0.06$) as seen in figure 16. The score for human opponents, as expected, was 2.5 and stemmed from the fact that human/human matches

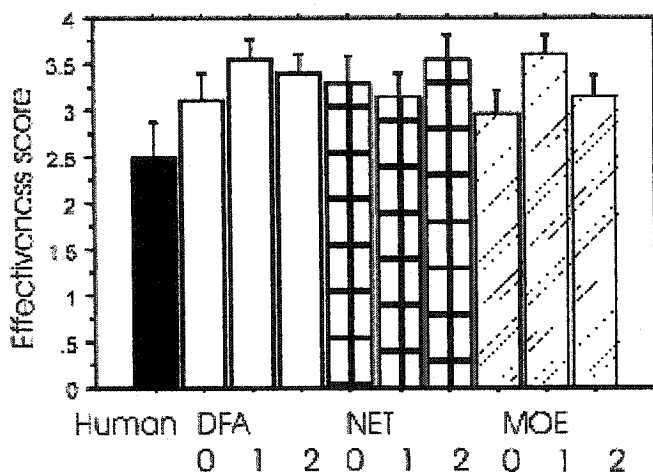


Figure 16 Effectiveness of agent for human, and the 3 software agents with their recursive levels. Each agent is treated on par with humans in this analysis.

were always paired as one winner and loser with a five total score. Although none of the computer agents reached this threshold of half wins, the non-recursive MOE and the non-recursive DFA came close with just under and just over three wins respectively.

The human score was then removed and the remaining algorithms were analyzed in a three (*algorithm*) x three (*recursive level*) repeated ANOVA along with *gender*, *experience*, whether the opponents had played Local Area Network games previously (*LAN*), and which opponent had a higher rated experience (*HigherEXP*). The first four factors are self explanatory, but the latter two bear mentioning. Although both of these variables were originally meant for the believability ratings, it was decided that they may be relevant with analysis as well. The possibility of information regarding the human in the next room affecting software's believability was straightforward, but affecting how you play against the software opponents was less so. However, if knowledge could affect the opinion of an opponent, it could also affect the style and strategy against that opponent and therefore the final score. Including all four variables for the effectiveness and believability ANOVAs also provides a greater continuity between analysis.

Software perspective

The main effect of *recursion* was significant ($F(2,30)=6.517, p<0.005$), and interacted with *LAN* ($F(2,30)=5.44, p<0.01$), *HigherEXP* ($F(2,30)=5.07, p<0.02$) and *algorithm* ($F(4,60)=2.728, p<0.04$) (Figure 17, A,B,C,D). Overall, *R0*, or bots which did not recursively model their opponents had the fewest games against them than those which did recurse, regardless of how deep the recursion (figure 17a). This pattern remained fairly consistent across all conditions except for a few instances shown in the interactions above. Participants who did not play network games (LAN) with their opponent prior to the experiment seemed to show no effect of recursion at all. Those that had prior LAN experience showed most of the differences including a further increase against agents at recursive level two (figure 17b). Splitting the data by whether there was

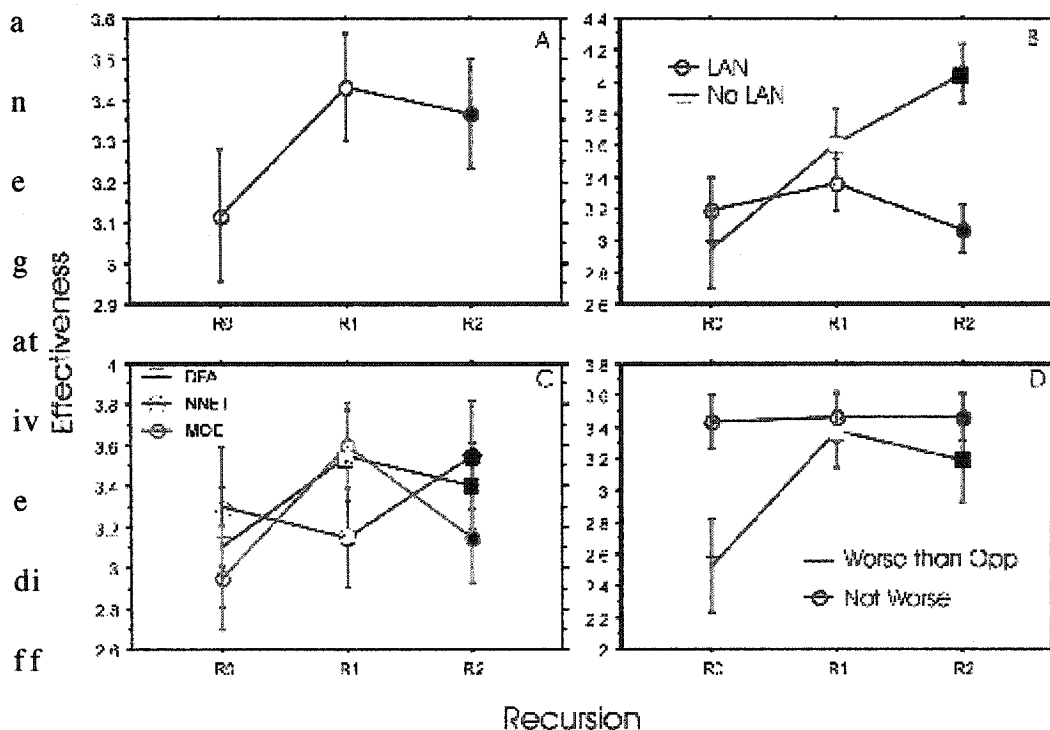


Figure 17 shows the various graphs for effects and interactions for effectiveness involving recursion (R0-R2). From left to right, top to bottom, they are the base effect (a), split by LAN (b), algorithm (c), and whether the participant had less experience than their opponent (d).

nce in experience levels between the two opponents (better than ones opponent), it was shown that the lesser experienced of the two opponents was responsible for the recursive zero effect, with all other conditions being nearly equal (figure 17d). Although it is difficult to say exactly what these trends mean for agent design (and recursion in particular), it is clear that human player knowledge, experience, and expectations have a large impact on results.

Although zero level (no) recursion was best overall (figure 17a) by nearly four-tenths of a game, it was only the best for two out of the three algorithms (figure 17c). Looking back at the MacInnes (1999) DFA results, we see nearly opposite results, likely due to the opponents changing from being a DFA in MacInnes (1999), to a human opponent in this study. This made sense due to the greater difficulty in predicting human patterns than that of a finite state machine. MacInnes postulated that the error for less accurate predictors would have an additive effect with increasing levels of recursion. The optimal level for recursive modeling would therefore be a function of the accuracy and we would expect different algorithms to peak at different levels of recursion. Looking at the recursive levels for each algorithm, we see that while the MOE and DFA were optimal at level zero, the neural net peaked at recursive level one. This would seem to agree with the prediction since the neural net was the most accurate of the three algorithms. The slight increase in effectiveness at recursion two as well as the lack of a significant algorithm by recursive level interaction in this study, however, suggested that there may have been more to the story than this simple explanation. Although accuracy may have been an influence in algorithm effectiveness, other factors seemed to have been at work.

A second explanation for the lack of recursion effect in the neural net and MOE was that

recursive modeling was already included implicitly *in the networks themselves*. Since the neural networks were trained on observations of human matches, these algorithms may have learned recursion as it was (theoretically) used by the human agents. Any further attempts at explicit recursion for these algorithms was ineffective due to the implicit recursion of the networks.

Algorithm itself was not significant, except in interaction with recursive level (discussed above), and with Gender ($F(2,30)=3.455, p<0.05$). Since the interaction with gender tells us more about our gender question than it does about our algorithms, it will be discussed from the human perspective.

Human perspective

The ANOVA showed no significant effect of gender ($F<1.0$), and the only marginal

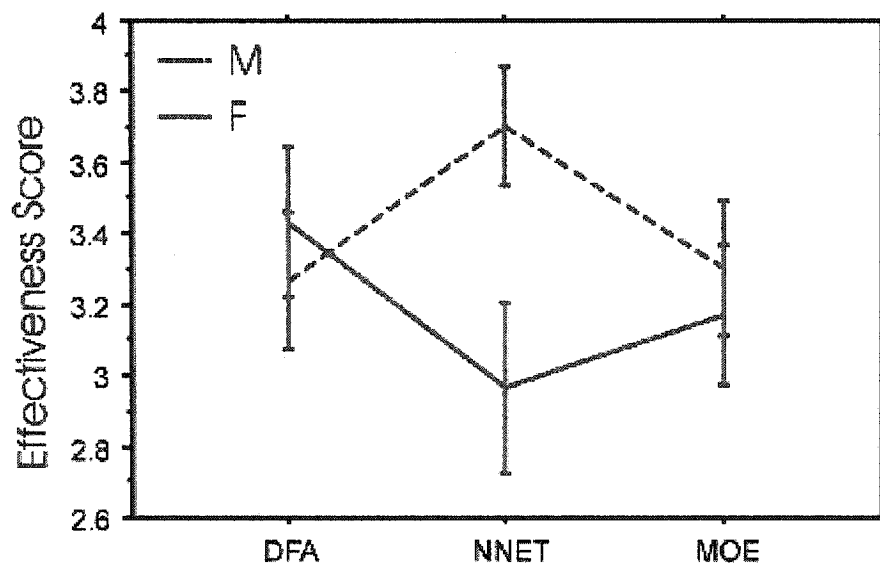


Figure 18 Effectiveness of algorithm against both male (M) and Female (F).

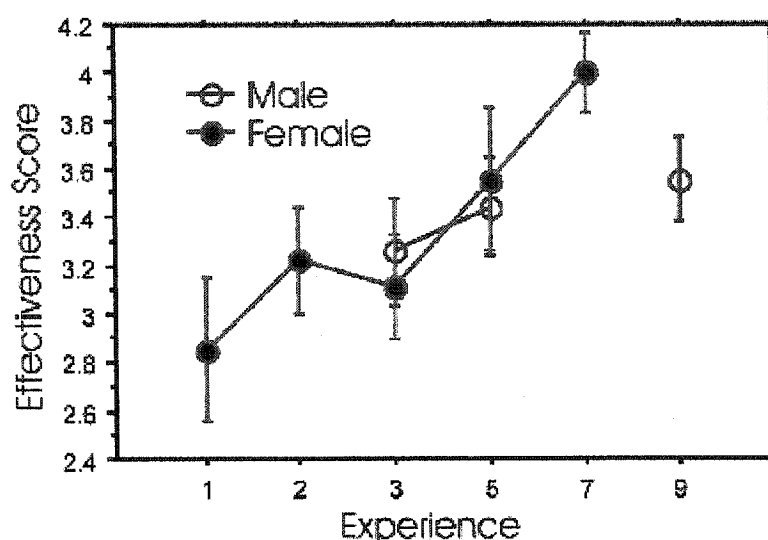


Figure 19 Effectiveness of participants based on their self-reported computer game experience, also split by gender.

interaction with gender was with the software opponent in the match. As seen in figure 18, there was little difference between men and women when fighting a DFA and MOE, but a significant difference (0.7 of a game in favor of the men) when fighting the neural net. Strangely, these were not the patterns we would have expected based on the accuracy data that showed that DFA's were better at predicting the location of women while the other two were better at predicting the patterns of men. It is obvious that what an agent did with a prediction was at least as important as the accuracy of the prediction itself.

It is interesting to note however that the MOE tended to perform better than the neural net even though the net was clearly the more accurate of the two. The key difference may have been that the accuracy score was averaged across a person's complete match. The effectiveness score, however, only needed accuracy in the brief time before a killing shot was fired.

One important non-result in the effectiveness data was the lack of significance ($F < 1.0$) of

gender. Although there was a slight difference in scores between men (mean 3.42) and women (mean 3.19) the statistics demonstrated that this was better explained by differences other than gender. One difference that was predicted that may have explained this was experience, but this also was non-significant ($F < 1.0$). It may be hypothesized that the reason for this lack of significance was due to increased variance due to the complexity of the task, or due to the lack of data in certain key sub groups. The same may be said for the lack of a Gender effect, but looking at the interaction of gender and experience (full scale) in figure 19, we clearly see that the driving force in effectiveness was experience, and that if anything, women of high experience tended to do better than their male counter parts. Again, this should only be seen as a trend towards experience, and not a significant interaction. Although with these results it was highly unlikely that, given more data, experienced female gamers would end up performing worse than their male counterparts. The most likely explanations were that the limited size of the sample in this elite group were showing women to be better when in fact they are the same, or if the difference was real, that the self-selected nature of this group was driving the improvement

Since this experiment could have been seen as a dual task, it was decided to look for evidence of a trade-off between effectiveness and believability. Subjects may have been dividing limited resources in to the two tasks differently and may have been affecting the experience (or gender) results with the tasks as separate measures. A regression of effectiveness by believability (as compound variables) in fact did show a small ($R=0.15$) but marginal effect ($F(1,160)=3.75$, $p<0.06$), but it was in the positive direction (Figure 20). Participants who scored well against their

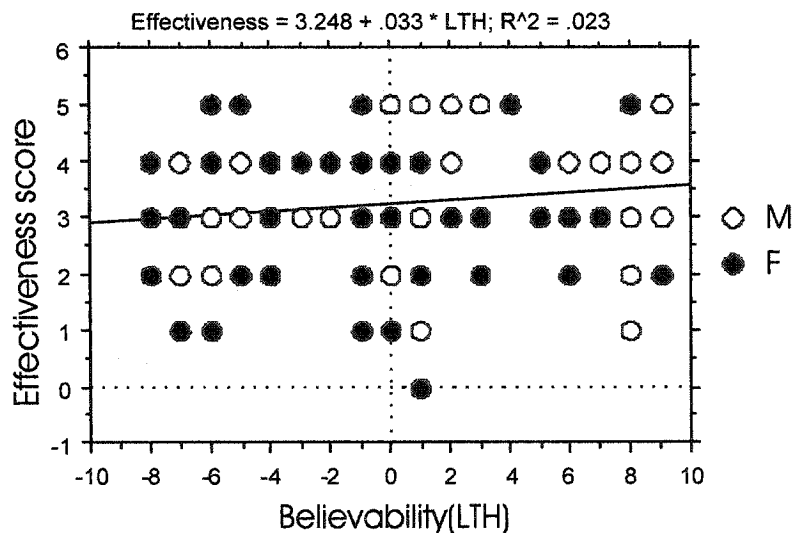


Figure 20 scatter plot of effectiveness and believability, along with the regression. Data further split by gender. Since believability is represented by 'Less Than Human' (LTH), the higher the score, the better the human participant did in guessing the software opponents.

computer opponents, tended to make more accurate judgements on the believability measure as well. This pattern also seemed similar for both men and women. Further results and discussion of effectiveness and believability relationships will be discussed more fully after the section on

believability by looking at a combined performance score.

The fact that women tended to finish matches earlier than men along with the different ability for algorithms (and even experts) to predict strategy (as seen in the accuracy data), was strong evidence for men and women using different strategies in these competitive environments. Perhaps even more important was the fact that these strategies, although different, *were just as effective against these opponents*. This result was consistent with research on gender strategy differences in spatial tasks as well as research on the importance of experience over gender (MacInnes, 2001). Although Shore et al. (2001) offered a counterexample of this effect, it was likely due to increased variance in gamers with high experience caused by significant differences in the environment with that of a typical computer game (collision detection of the Shore et al. mazes was different enough from typical first-person-shooters that performance was likely hindered for experienced gamers).

Experiment 2

Experiment 2 looked at a single pair of participants with very high gaming experience, who were familiar with the Hebb-Williams mazes (although not this experiment), and had played LAN games together prior to testing. Participants were tested in same experiment three times in sequence with only the random order of opponents changed each time. Since Experiment one produced what was likely a generous estimate of the abilities of these three algorithms, it was decided that a more thorough test against highly experienced participants was needed to fully test the limits of the algorithms.

Since both participants were male, gender was no longer an issue for the analysis. The remaining factors; *recursive level*, *opponent* type, the length of *time* of a match, the *order* a match was played, and the new factor - the *sequence* of the experiment itself was subjected to a factorial ANOVA. Due to the limited number of subjects (two) for Experiment two, it was not surprising that there was a lack of significant effects, even limiting the interactions to level two.

Ideally, more subjects would have been added to this experiment to give power to the non-significant trends. Unfortunately, the participants reflect a highly specialized group, in addition to being required to show in pairs. Future studies could be organized around 'LAN tournaments' in which just such game players arrive from all over to compete in network computer game matches. Given the caliber of players that usually attend these events, it should be relatively easy to find volunteers for an experiment of this nature.

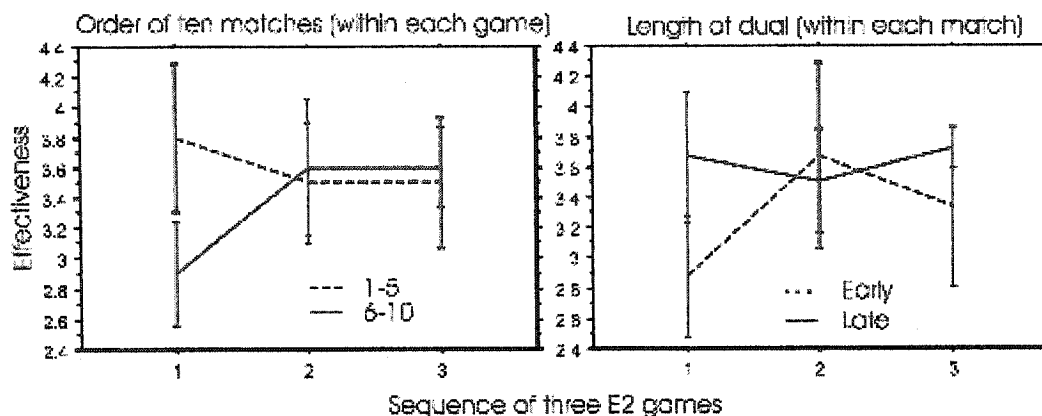


Figure 22 Effectiveness across the three games in E2. Results are shown split for order (1-5, 1-10) and length of dual (early/late). The first game in both cases show the majority of the differences.

Some of the other non-significant 'trends' will be discussed in some detail, in particular those that confirm or deny results from E1, or those that speak to the purpose of this study. The effect of match time mentioned above, does have a qualifier. As would be expected, this effect is

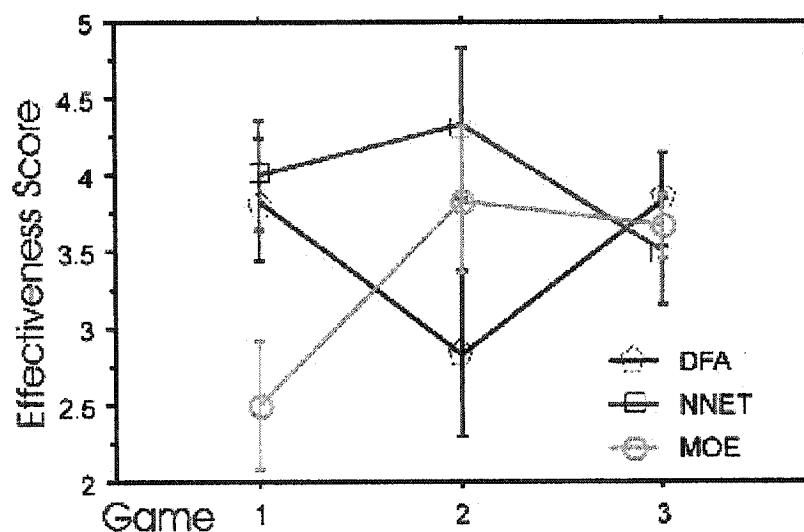


Figure 21 Effectiveness score for the three games in E2 split by software opponent.

tempered as players progress through the three experiments in sequence. As seen in figure 21b, almost the entire effect of time existed in the first of the three alone. Although fast matches hindered these participants in the first, they adjusted for the second and third. The same held true for the order of the matches, an overall non-significant difference and interaction, but the graph (figure 21a) reflects a trend in that the first experiment order appeared to be a factor. Both of these trends, while not significant, did demonstrate the speed at which these participants adapted to the experiment. By the second run of the experiment, both results were nearly identical.

The (non-significant) order by algorithm interaction from E1 also continues the trend in E2. Figure (22) shows a very similar trend in many details. The MOE performs better than the neural net regardless of order (from one to ten within a game). The DFA as well, is the same whether it appears early in the experiment or later. Both network based algorithms, however, start as being as good or better than the DFA, but tend to perform worse in later trials than in early ones.

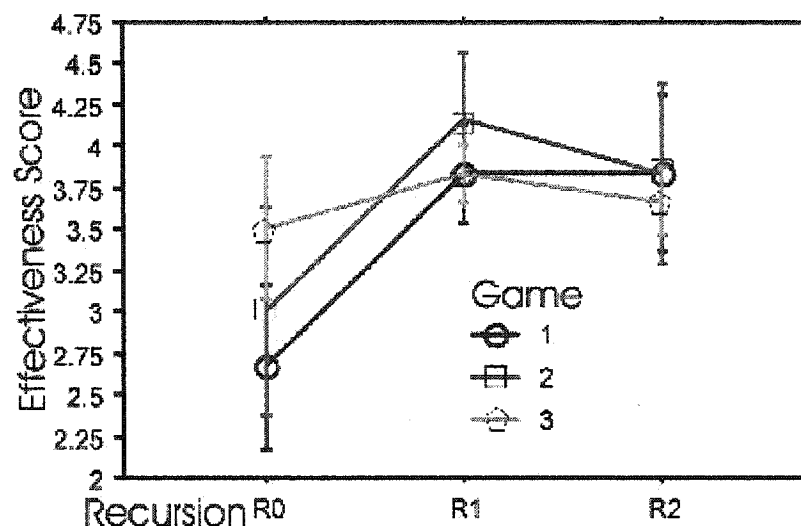


Figure 23 Effectiveness of highly experienced E2 participants across recursive opponent and split by order of the three games played.

Although the MOE has a tendency to win more than the other algorithms in the first run of the experiment, there seems to be little difference by the end. It may take different amounts of time to develop strategies against these algorithms, but these experienced players do quite well against all of them by the end.

As with experiment 1, the data was also subjected to a three (algorithm) by three (recursion) repeated ANOVA for proper analysis. Only recursive level was significant and, as with previous results, had no recursion as optimal. Although an analysis with sequence of matches was not possible (due to a singularity in the ANOVA matrix), it did follow a now familiar trend with these experienced players showing no differences by the third competition (figure 23).

Believability Rating

Experiment 1

The believability of an agent, as mentioned previously, was how capable the algorithm was in fooling the human opponent into believing that the agent was also human and was gathered as a limited scope Turing test. Since much of this analysis looks at believability, an initial analysis was conducted on how believable the human opponents were as humans. The first point of interest was that people do not seem to be very good at spotting human opponents. Although the median guess was a respectable positive three, the mean was actually +0.75, just barely on the human side of the Turing scale and actually less than the minimum correct guess (+1.0). An ANOVA (limited to base effects and no interactions due to only one data point per subject) of this human data showed no significant effect of any of the independent variables (All F 's < 1.0).

Participants in E1 had a better showing, however, with a mean guess of positive 3.3. This result seemed entirely due to the first match with further matches averaging between four and five. Although this interaction was not significant ($F < 1.0$), it was likely due to the variance in the first match alone. Highly experienced gamers from this study had little difficulty in identifying human opponents after minimal exposure to the experiment.

Since we had ratings of how human an agent was for every opponent, there were many possible ways to combine these ‘Turing’ scores to measure an agent’s believability. The true rating itself, as recorded by the experiment was a number ranging from one to five and reflected least to most confident. A positive or negative sign on that number reflected a guess of human or computer respectively. Possibilities for a believability score included the raw Turing score, an error score that showed how far the score was from a perfect guess, and finally a comparative that reflected how close the agent came to the believability of actual humans. Each of the three possibilities will be discussed below.

The raw *Turing* score was not appropriate for this analysis since it held no relation to the human score of any particular subject, and also did not allow for comparison of software and human agents due to the different base point. The Absolute Guess Error (AGE), calculated as five minus score for human agents and negative five minus score for software agents, was slightly better. It allowed for comparison of all agents by comparing each to its ‘optimal’ guess as a baseline, but did not include reference to that subject’s best (albeit variable) guess for humans. It was decided that the best choice for dependent variable in the believability analysis would be human-centric measure in spite of the difficulties participants had in spotting other humans. It was calculated by

subtracting each agent's Turing score from the Turing score that the participant rated their human opponent. This resulted in a relative rating of how much 'Less than Human' (LTH) an agent was with smaller numbers reflecting more believable agents and negative numbers suggesting agents that were more 'human' than the humans. The AGE score, however, will be mentioned briefly for thoroughness, in particular where it differs from the LTH score.

As mentioned with the effectiveness analysis, whether someone had played network games with an opponent was one of the between subject variables. One other piece information that was gathered that was not used was whether opponents even knew each other. The two variables could not possibly be tested since it was impossible for two people to have played each other in a LAN tournament and not know each other and were therefore confounded by necessity. Although they could not interact, the different pattern of data between these two variables warrants mentioning. As seen in figure 24 (A&B), these IVs show very different results. Figure 24A shows an increase in error when participants knew their opponent. This means that *computer* opponents

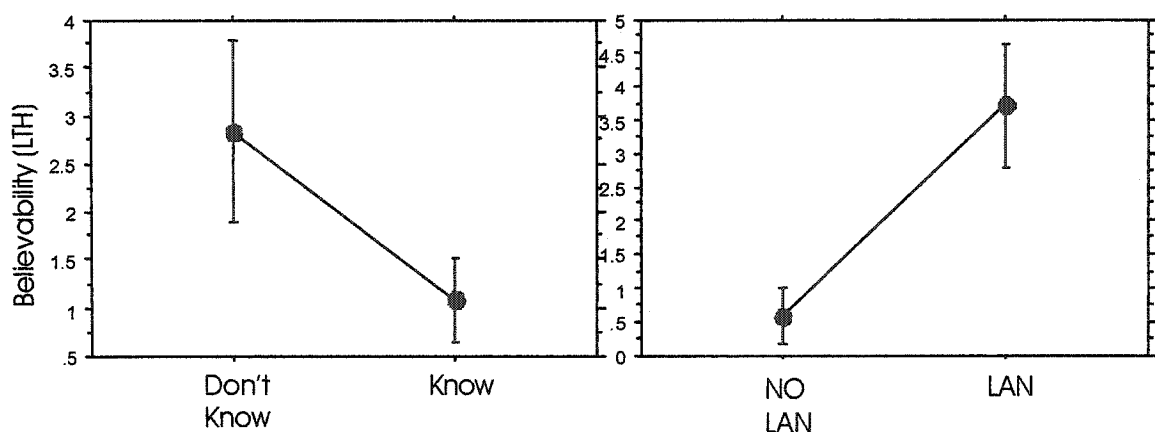


Figure 24 Believability rating for both prior knowledge of human opponent, and prior LAN experience with human opponent.

seem to have been more believable when participants knew the human opponent in the other room. This may seem unusual unless the trait of anthropomorphizing is considered. People tended to give human traits to inanimate objects (computer agents), and it seems reasonable to assume that this may have increased when they were looking for traits of someone they were familiar with. This was supported by subjects' reflections after the experiment which included comments such as "...Ah, I see what TS was planning...". This effect, however was reversed when participants had also played each other in competitive LAN games. This seemed to give participants extra information about the opponent's playing style which allowed them to more accurately determine the non-human opponents.

As with effectiveness, believability was initially subjected to a ten factor ANOVA with all opponents contributing equally. Since one of the assumptions of an ANOVA analysis is the uniformity of variance, it was decided to use the AGE score instead of the LTH score for this initial stage. Since the human opponent was the baseline for the LTH score, the resulting score for that opponent would have a zero mean and zero variance that would have called into question the validity of the analysis. The AGE score however uses the 'ideal' guess as the baseline, and still allowed comparison across algorithm. There was no significant effect of algorithm ($F < 1.0$) in this base analysis.

Algorithm along with *recursion* were then combined with *gender*, *experience*, *LAN*, and *HigherExp* in a three x three (within), two x two x two x two Repeated ANOVA with Less-Than-Human(LTH) as the dependent variable, and again with limited interactions for the between subject

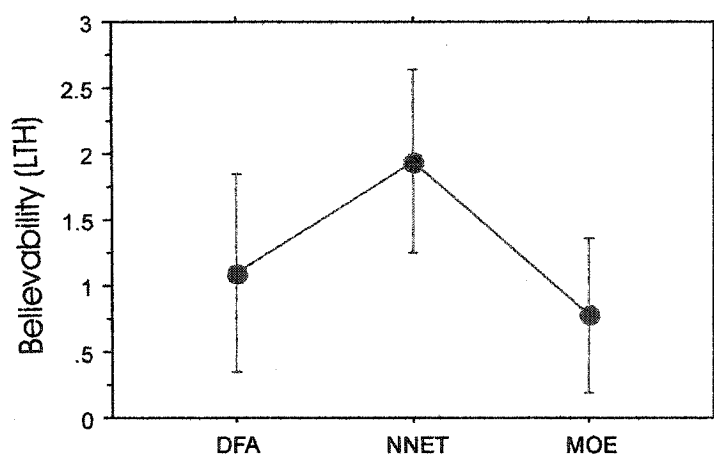


Figure 25 believability for the three different software algorithms.

marginal significance ($F(2,30)=2.833, P<0.08$). As we see in figure 25, performance exceeded the others, with an average difference from the human score of less

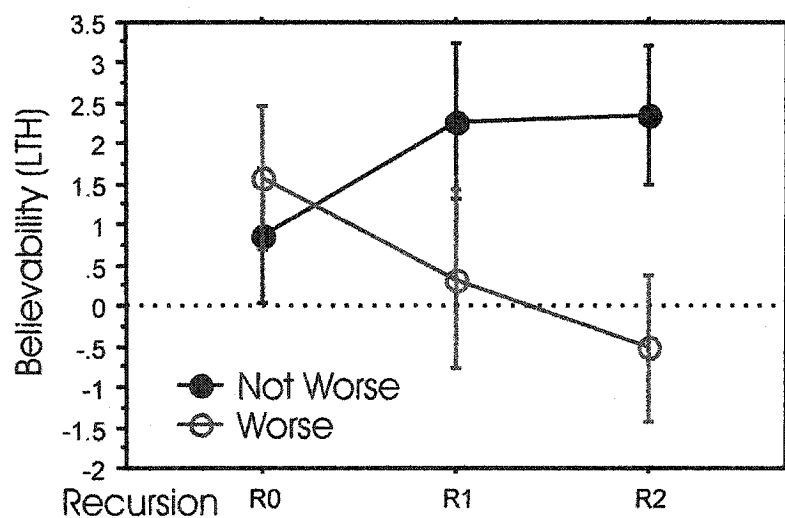


Figure 26 Believability of software agents by their recursive level, split by whether the human opponent had a lower experience rating.

than one rating point. The DFA was second with a mean of just over one and the neural net was the least believable with a mean difference of over two rating points.

significance was due to the lack of power (0.362) as a between subject variable or do to the duel nature of the task.

Significant interactions included recursion with LAN and recursion with HigherExp. Figure 26 shows that although zero level recursion performs better overall, this was not the case for all groups. Participants who had played their human opponent in LAN games found the process of recursion to create much less believable agents, while those without such experience found recursive level two to be most believable. In fact they were found to be as believable as the human opponents. This could have reflected the additional knowledge of such participants, but those who answered 'no' may have played these network games, just not with their current opponent. It is more likely that these LAN pairs had knowledge about their each other's style and that they were more capable of spotting different agents.

Similarly, when an opponent's experience rating was higher than their own, the optimal

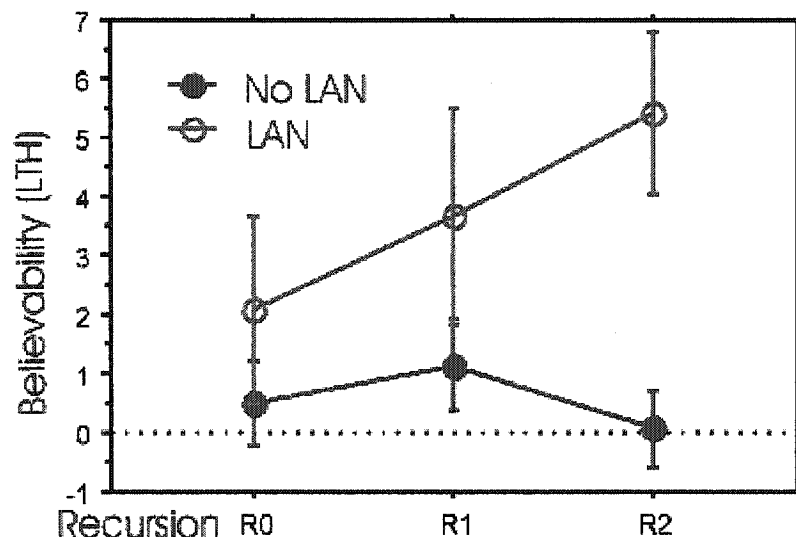


Figure 27 Believability rating of each agent's recursive level split by whether the human opponents had played LAN games together. LTH scores more believable agents as lower.

recursive level again shifted to level two which actually becomes more human than the humans themselves (figure 27). It is likely that this was a reflection of the fact that many participants may have had performance expectations of their opponents based on knowledge of whether their opponent was better than they were. This may have shown through with a three way interaction with recursion and LAN, but level two interactions were not possible.

Experiment 2

The only extra information gained from E2 with regard to the believability results was with learning as the three experiments were run. The expert participants from this experiment had similar

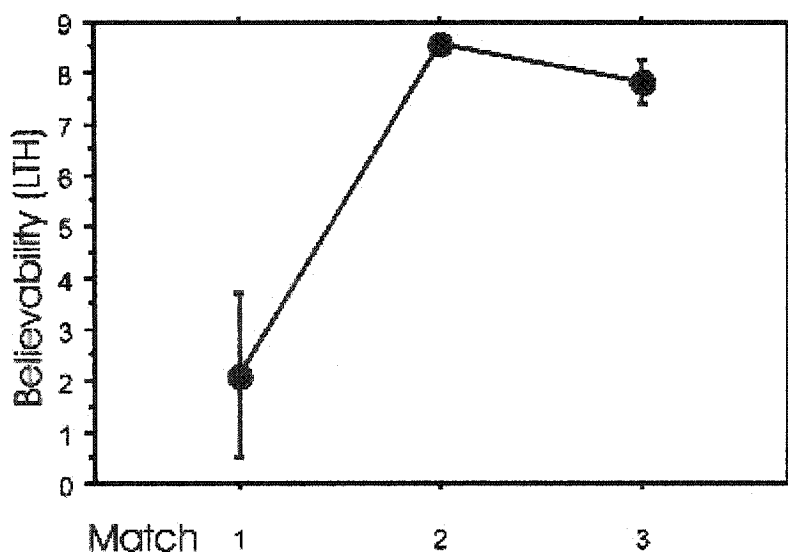


Figure 28 Believability of agents in E2 across the three consecutive runs (Match). Note that with the LTH scale, a 10 reflects a perfect Turing guess on the human opponent as well as a perfect guess on the software opponent. After the first match, these expert players are not fooled by the agents.

troubles to other subjects spotting the computer agents in the first experiment, but had little difficulty in the final two ($F(2,41)=4.92, p<.02$). In fact, as seen in figure 28, subjects are near zero error for both of the final two runs of the experiment.

Combined Performance

Due the lack (or near lack) of some expected results (experience) along with the significant regression between effectiveness and believability, it was decided to try the analysis with a combined 'Performance' score. Since participants knew that they were expected not only to defeat their opponents, but also determine which ones were human, the experiment was essentially a dual task design. To combine effectiveness and believability, each was first converted into a Z score by calculating how many standard deviations an individual was from the mean for that score. These Z scores were then combined to create a new 'Performance' score. The resulting score was then subjected to a three x three (within); two x two x two x two (between) ANOVA as were the originals. As with the original scores, interpretation of the result depended on the perspective of the analysis, a higher number was better as was seen from the human participant, but a lower score

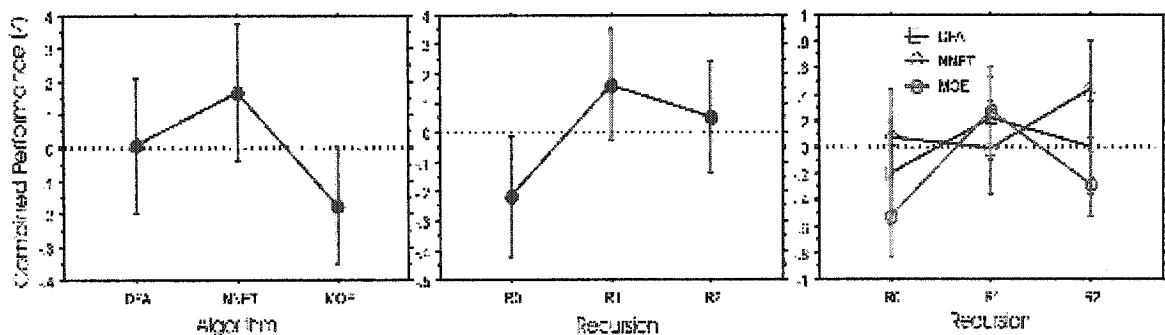


Figure 29 Graphs of the combined performance for algorithm, recursion as well as the interaction.

was better when considering the performance of the agent (low score *against* that agent).

There was a marginal effect of recursion ($F(2,30)=3.27, P<0.06$), a significant effect of algorithm ($F(2,30)=3.53, P<0.05$) as well as an interaction between the two ($F(4,60)=3.22, P<0.02$). The trends (Figure 29) were also the same as those found in the individual analysis: Recursive level zero was the best overall result; Mixture Of Experts (MOE) out performed the other algorithms; and the neural network had a different pattern than the other two algorithms in that recursive level one was optimal with a sharp drop at recursive level two. It is important to note that the original analysis are still important in light of this combined score. While this result suggests that both algorithm and recursion were important for an agent's performance, the recursive level was more important for the effectiveness component while the algorithm had more influence on believability.

From the human perspective, the first result (marginally significant) was the main effect of

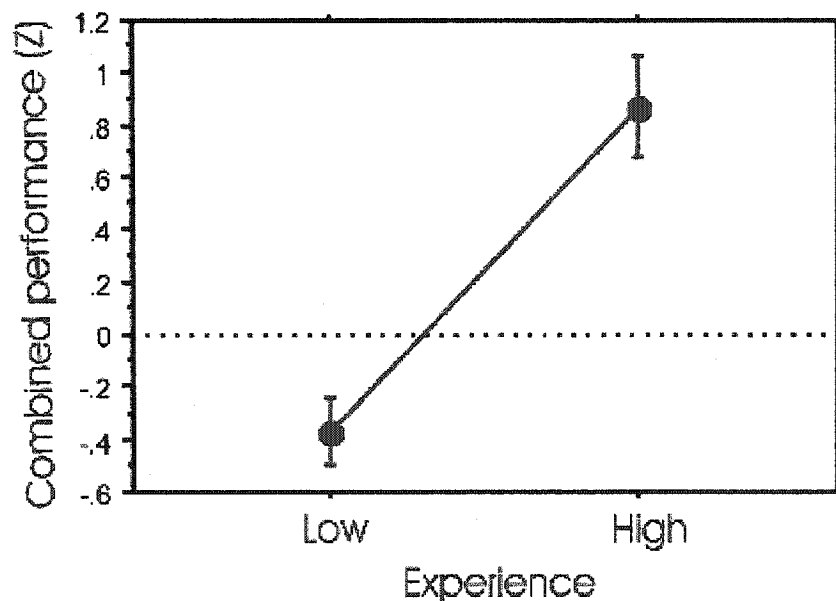


Figure 30 Participants who rated themselves as higher experience performed far better than those who self-rated low.

experience ($F(1,15)=4.18, P<0.06$). As expected, those with high experience tended to perform better than those with lower experience (Figure 30). The fact that it was not (or barely) significant with the individual analysis was indicative of the way that different participants allocated resources for the two tasks. The regression (effectiveness \times believability) was positive, suggesting a complementary relationship, but the variance added by different strategies and emphasis could easily have caused the result described above. Another important side note was that despite a 0.6 difference in mean standard deviation (Z score), there was still no significant difference between the genders in terms of performance ($F<1.0$). While this difference may have been significant without considering other variables the observed difference could be explained better by other variables, such as experience.

As with earlier analysis, the interaction of LAN and recursion remained significant

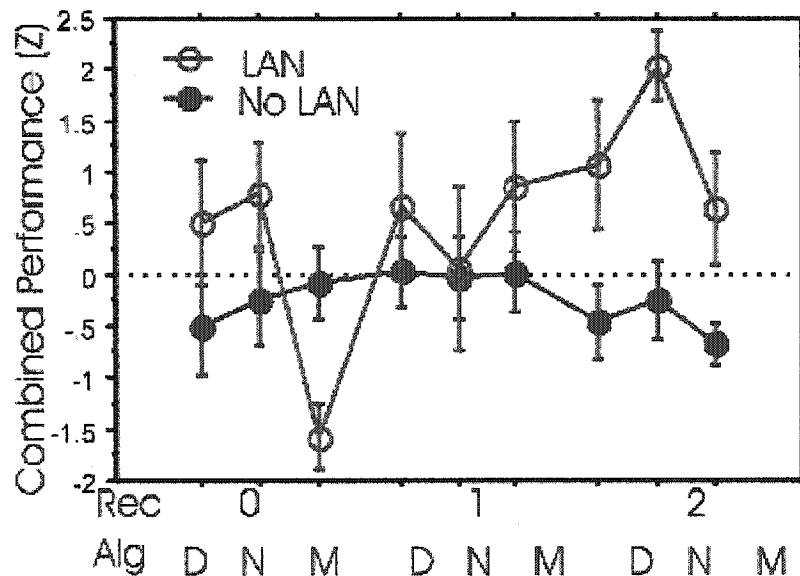


Figure 31 Combined performance score for each software agent (algorithm and recursion) split by whether the participants had played network games before.

($F(2,30)=7.94$, $P<0.002$) as did the three-way interaction of Opponent x Recursion x LAN ($F(4,60)=2.72$, $P<0.04$). Figure 31 demonstrates that recursive level two behaved differently depending on prior LAN experience of the two opponents (as with the believability score in figure 26). The three-way interaction, however, was mostly driven by the MOE at recursive level 0, which seemed to cause more trouble for the LAN gamers than any other algorithm and more so than for the non-LAN participants.

Chapter 6: Conclusion

Experiments

One of the true benefits of Human-Computer Interaction research is the multiple perspectives from which the data may be considered. Not only do the computer and software tell us about human traits and preferences, but the human performance can tell us a lot about the software. In a true interaction, information flows both ways as this research has clearly shown. Results are summarized below from both the human and the software agent's perspective

Software Perspective

The results reported from the training for the machine learning algorithms as well as their use in various experiments have shown that all three algorithms were very effective under different circumstances. DFAs, MOEs and neural nets were shown to have different strengths depending on the task, the opponent as well as the external circumstances of its use.

The accuracy data, which most closely resembles a true machine learning problem, tested the three algorithms' ability to predict an opponent's position based only on current and/or previous state information. Although all three algorithms were able to do the task, the neural network was the superior algorithm. The MOE performed the second best at this task with the DFA placing third. The neural net was so effective, in fact that the error for predicting ten time samples into the future was the same as for the DFA at only four time samples in the future.

Although the stereotypes based on the MOE solution were not as accurate overall as the single neural net, it still contained training properties that made it a valuable solution. As would be

expected with a clustered solution the resulting stereotypes showed far less variance within the groups than in the overall data. The experts trained from these groups were also able to train in much less time than the overall neural net and with less complex network structures. Although it was originally hypothesized that these stereotypes would be less susceptible to overtraining due to the specialized nature of the data within the groups, many of the experts seemed to exhibit more properties of overtraining on the test data. This may have been caused by not finding the optimal network structure of these neural nets, the small amount of data in some of the smaller clusters, or perhaps even the nature of the clustering process itself. In any event, this question would be ideal for future research.

The most significant main effect on the effectiveness of the computer agents is from the recursive level that agent used in modeling the opponent. Despite earlier work that showed an advantage for recursive modeling between two DFAs (MacInnes 2001), this study found that no recursion was optimal. Since it was hypothesized in MacInnes (2001) that the optimal level (and drop off for higher level recursion) was due to compounding error as the recursion progressed, it is reasonable to assume that this effect was due to the high error rates for these algorithms. Partial support for this suggestion was found in the better than average recursive modeling of the highly accurate neural network.

One of the major hypotheses of this paper, that a Mixture of Experts based on dynamic stereotypes, would outperform other modeling algorithms met with success, but only in some of the measures. As mentioned previously, the MOE did well in the accuracy tests, but not quite as well as the single neural network. The MOE was able to train more efficiently (fewer hidden nodes,

fewer training iterations) than the neural network, and a few of the expert/stereotypes did perform as well or better than the neural net even though the algorithm as a whole did not. The clustering of the experts, however, did offer a number of insights into strategy that would be useful in future research.

The effectiveness and believability data (when supported by the combined performance measure) for the MOE was much more clear. The Mixture Of Experts was superior in both measures and significantly superior in the combined performance measure. Of the other two, the roles reversed with the neural net being the second most effective and the DFA the second most believable. This trend was even more pronounced for participants who rated high on gaming experience. Where inexperienced players were fooled (nearly) as often by the DFA, the experts showed no problem identifying that they were playing the simpler algorithm. Experienced players had nearly a one and a half fold increase in error for the MOE over the other two algorithms.

Although the MOE was not the most accurate opponent, it consistently rated highest in performance for both effectiveness and believability across all experiments. With the neural net being the most accurate, it was interesting that it was not also the most effective. However, the algorithms, by their nature, differed in how they were able to use the accuracy information and the stereotype separation of performance seemed to work very well for the MOE. It is also worth noting that the accuracy scores were the average performance over the entire match. Effectiveness, however, was most heavily influenced by the accuracy in the last second of a match (firing on the opponent) and may have been slightly different than the average. Whatever the reason, the MOE was a superior performer in both higher level performance tests.

Human Perspective

The MOE displayed a number of features that connected to the literature on gender differences with spatial ability. While testing the basic accuracy of each algorithm, it was observed that different algorithms performed differently depending on the gender of the opponent. Where the MOE and neural net were better at predicting locations of male opponents, the DFA was better at predicting women. This effect interacted with experience in that DFAs were better at predicting less experienced opponents and the neural net was best against those with more experience. The MOE, interacted with both of these variables, predicting differently for each combination of gender and experience.

Since the movements each participant make are a reflection of the strategy they use against these agents, we can look at these results as varying against the strategies that these different opponents were using. In this light, the gender differences that we see in prediction, reflected the different strategies that are hypothesized to be used by men and women. Further evidence for these different strategies was found in the clustering data itself. The Kohonen SOM/ K-Means hybrid seemed to differentiate data based, in part, by gender even though this was not a factor in the distance measure used by the algorithm. Since the overall MOE did less well in predicting women, it was no surprise that clusters with higher than average percentages of samples from women, tended to be the least accurate and more variable than the others.

In spite of significant gender differences in accuracy, however, there were no significant gender differences in how effective these algorithms were in defeating opponents. Men and women used different strategies, the three algorithms predicted these strategies to different degrees of

accuracy, but men and women performed equally well against all of these algorithms. Although participants with different experience levels showed the expected effect, men and women *of equal experience* did just as well against the computer agents. It is often more difficult to find women in this category, but female computer game players seem to do just as well as their male counterparts. Even though there are gender differences, there are no gender advantages in this study.

The most obvious message of the believability data is that most people are not very accurate at spotting human opponents in this environment. The mean across all groups in the first experiment was barely better than the midway point between a guess of human and a guess of computer. Highly experienced participants from the final experiment did much better after some exposure to the experiment - after the first run through the experiment.

An interesting result of the believability score was one that bore no relation to how well the computer agent was programmed at all. Results have shown that participants were more likely to think the computer opponent was human if they knew the 'potential' human opponent in the other room. Just the thought that they may have been competing against their friend increased the amount that they anthropomorphized the computer opponent. This trend was reversed, however, if the two friends had also played computer LAN games together in the past. It seemed that knowledge of each other's behavior in similar games translated into better estimates of the friend in this new environment.

Contributions

The first of the contributions of this research fall under the heading of tool development for cognitive research. Over seven-thousand lines of code went into the creation of a virtual arena and the agents that inhabit it. The environment was designed to be flexible enough to run a variety of cognitive experiments and mimic many standardized settings including the Hebb/Williams mazes (including the original version as well as this modified task). The timing of the code was researched and optimized to be accurate to millisecond precision, even in a multi-tasking operating system such as Windows. The modularity of the C++ code allowed many possibilities for agent algorithms, only a fraction of which had been implemented for these studies. In addition, all of this code had been released to the scientific community as 'Open Source' in a bid to foster science and ease of replication.

The algorithms used to create the 'brains' of the computer agents have contributed to the literature for both machine learning and user modeling. A relatively new approach was used to provide dynamic, clustered input to the neural networks used in the Mixture of Experts. Although this solution was not the optimal algorithm in all tests, it showed a great deal of promise, as well as shedding some light on strategy differences within the participants.

Finally, this research gave preliminary data on which factors were important in modeling users within these environments and under what conditions. The within-subject variables such as algorithms and recursive level could be modified and improved depending on the desired result of effectiveness and/or believability, but the between-subject variables usually cannot. Developers of environments can control the program, but factors such as experience and gender cannot be optimized, only understood. By working with the effects that these factors have on a system,

developers can better design environments, agents, competitors and collaborators that best suit any particular combination.

Future Work

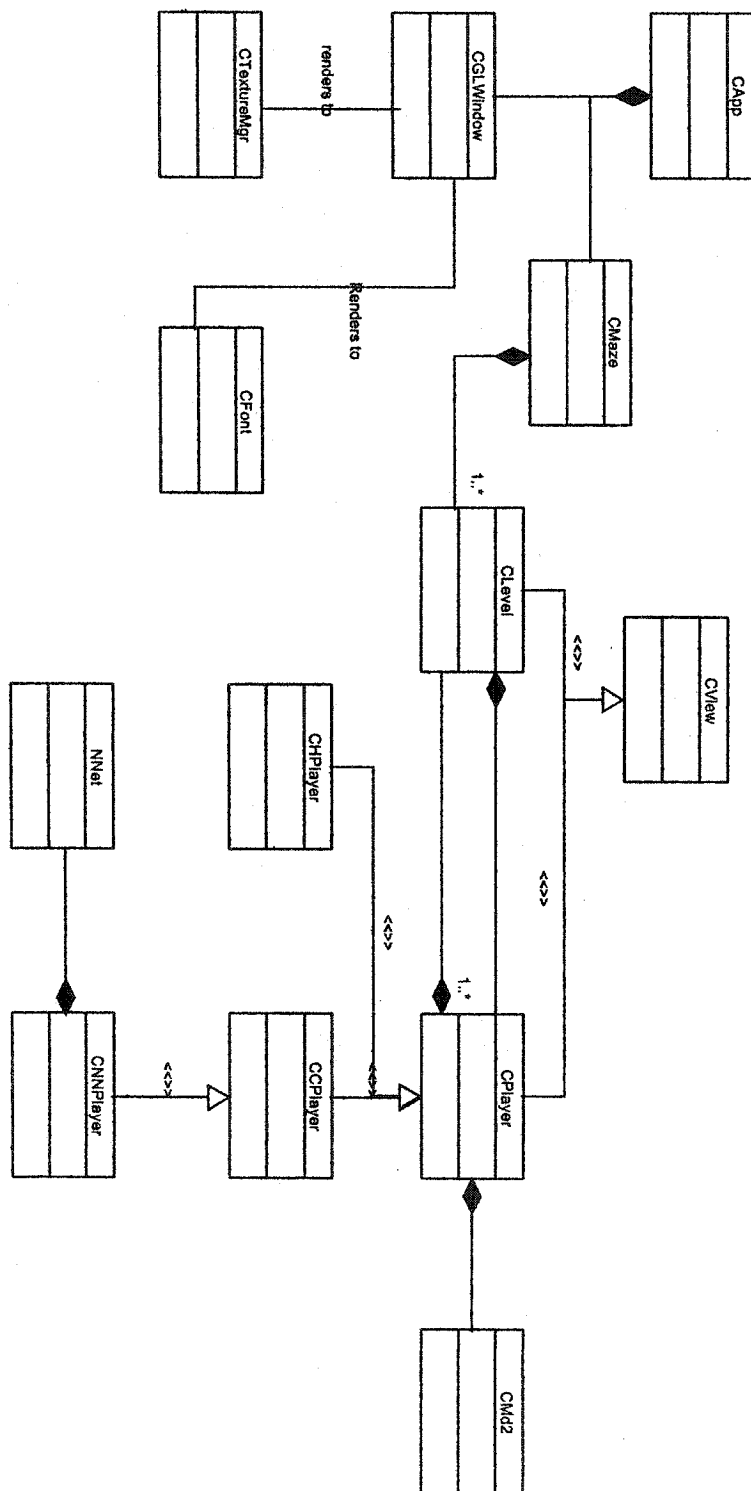
The four primary avenues for research stemming from this dissertation would be extending the connection to the research on gender differences, replicating this research in other important multi-agent environments, looking more closely at experienced and inexperienced users as the two distinct populations that they appear to be and improving the algorithms used for user modeling (possibly hierarchical clustering and MOE).

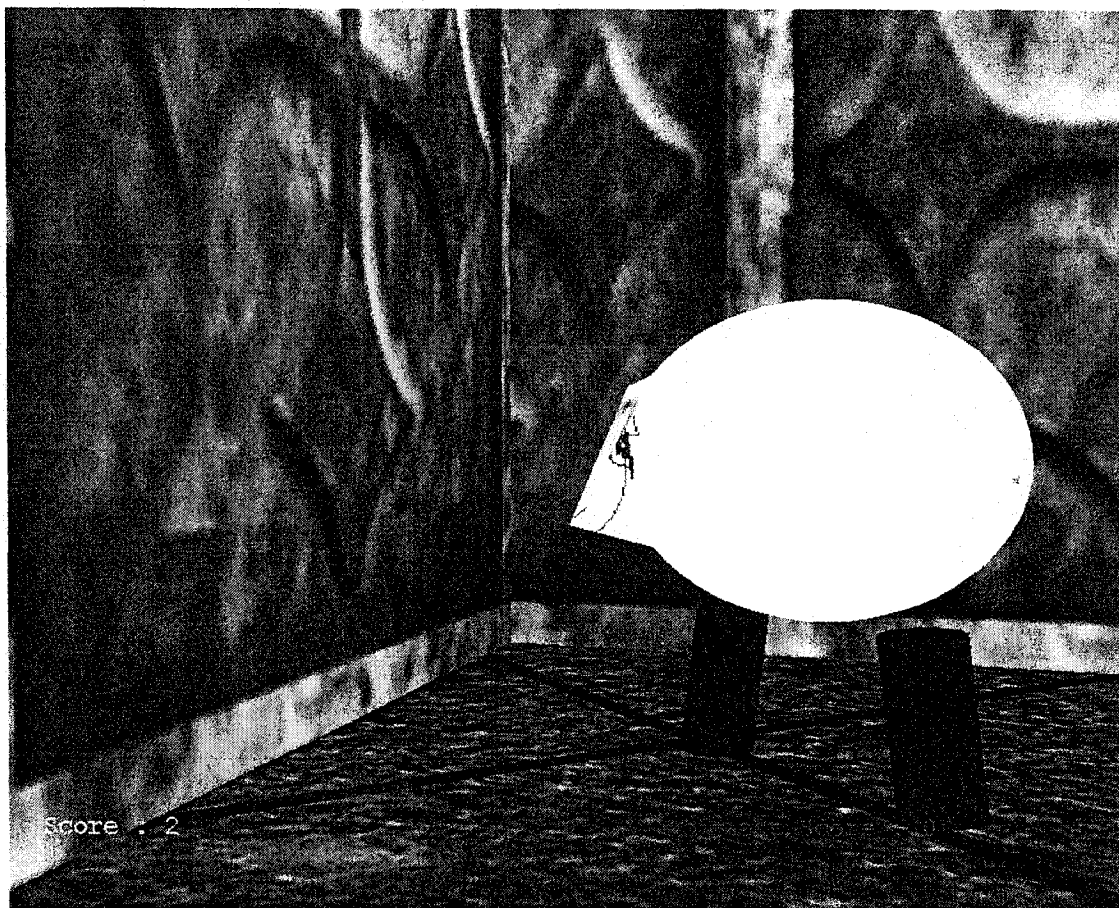
With the host of new research being done on gender differences in spatial ability (in addition to this dissertation), it would be beneficial for research to attempt to link the various biological and social connections for these differences. With results linking testosterone with gender differences, the success of researchers training away these differences, and the current results showing the importance of experience, an interdisciplinary study looking at whether spatial tasks (such as video games) increase testosterone levels may be warranted.

As mentioned earlier in this document, other competitive multi-agent systems exist that could benefit from this type of research. Although these results should be indicative of other environments, tests with training simulators and on-line trading environments could be an important replication of these results.

Finally, due to the importance of experience in most of the results in these experiments, studies that look specifically at each group may be warranted. Both subjective and objective results

of this research suggest that participants with differing experience levels look for very different things when competing in these environments and will likely be better served by different computer solutions. In addition to this, the goals themselves may differ across experience levels when looking at these multi-agent environments in the real world. Whether it be for simulation, commerce or game, it is likely that inexperienced participants would be looking for a 'training' opponent that would help them build the skills needed for that environment. Expert competitors, however, would likely be looking for an opponent that could give a proper challenge to their abilities. Competitive algorithms are needed it whether it be a 'Quake-bot' that plays like an expert human, or a simulated pilot that can participate in an accurate dogfight.





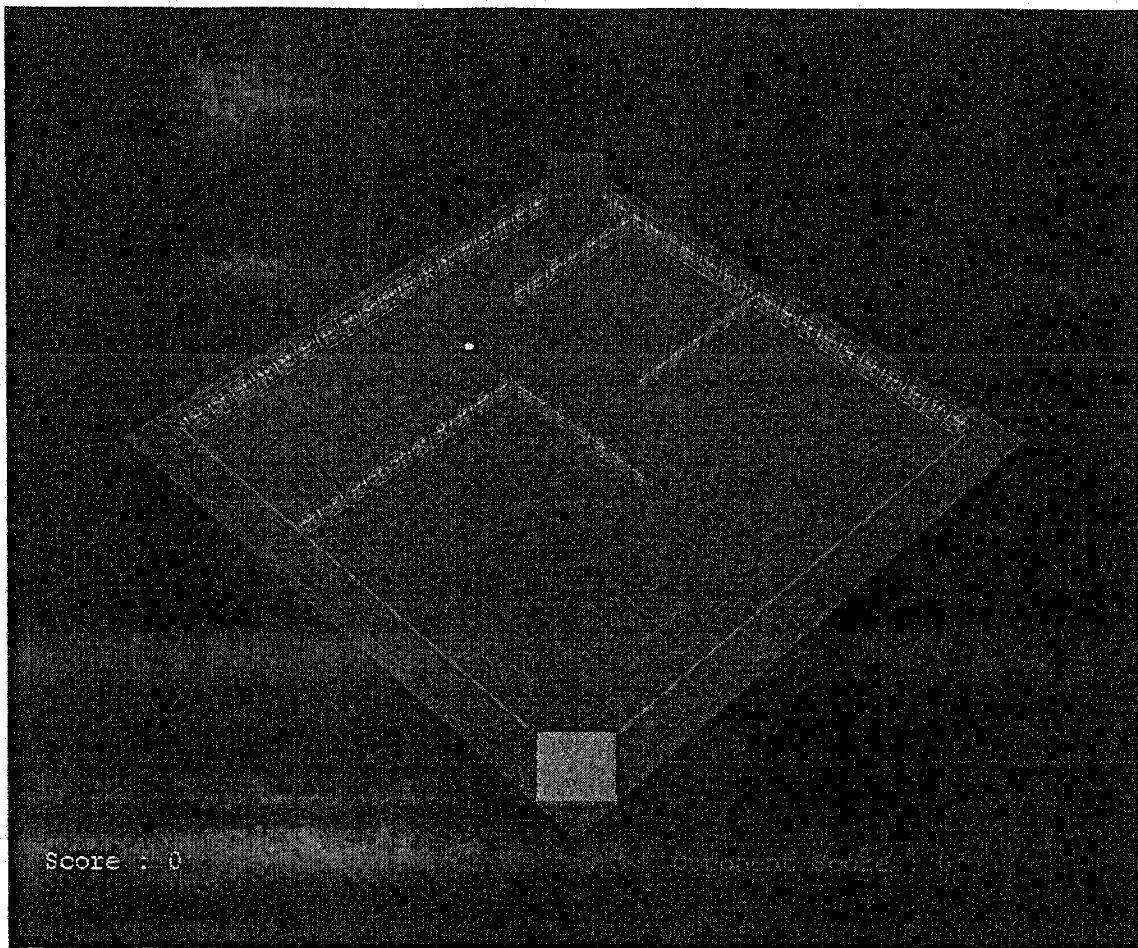
Appendix B.1 Side view of opponent from participant's perspective



Appendix B.2 View of opponent's 'face' from participant's perspective. The face and leg positions were cues to opponent's current facing.



Appendix B.3 Distance view of opponent from participant's perspective.

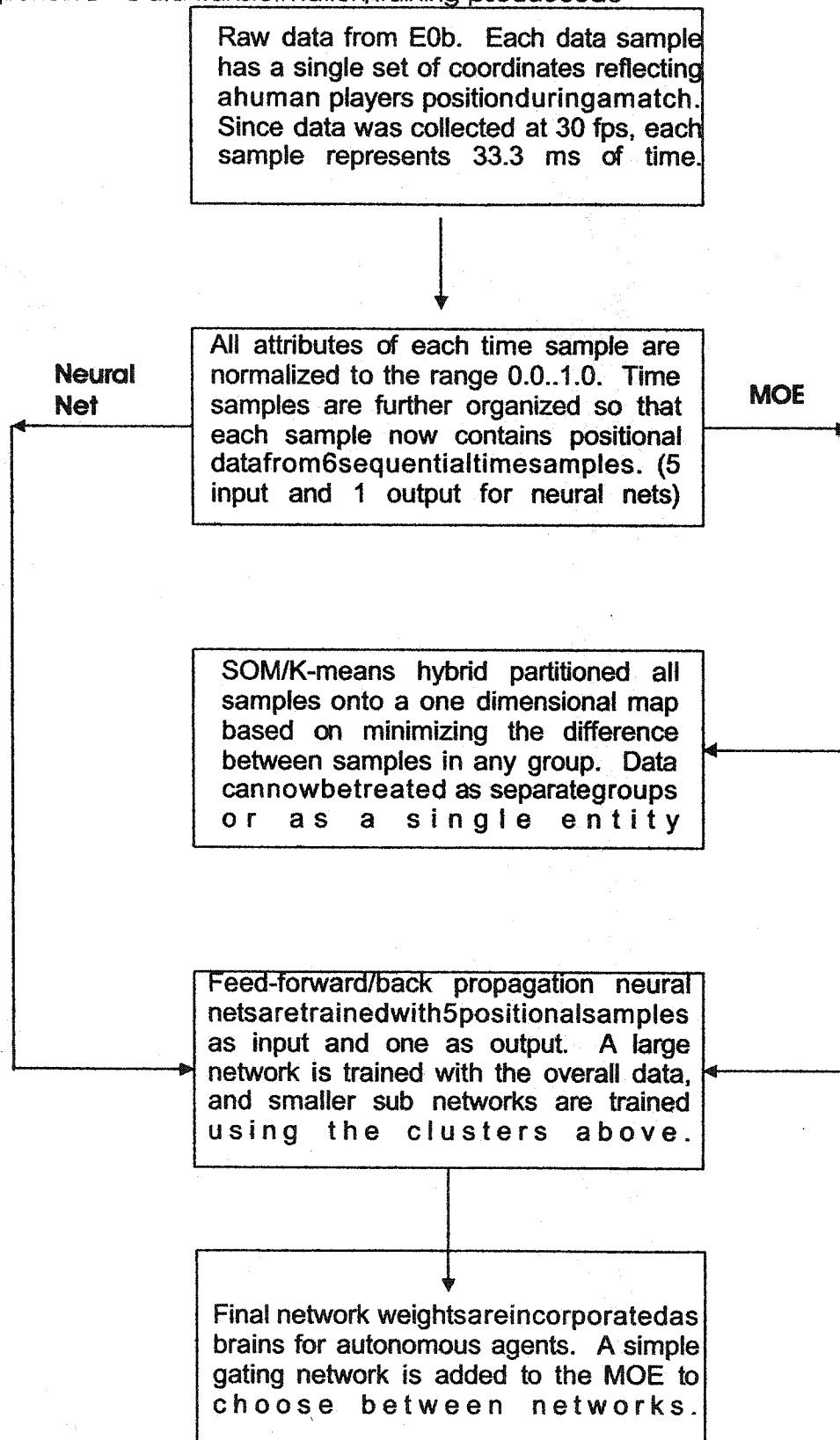


Appendix B.4 Overhead view of the fourth Arena (the one used in E1 and E2). Note that this perspective is for demonstration purposes only, and was not available to human or software agents during matches.

```

int WINAPI WinMain( HINSTANCE hInst //Win32 entry-point routine
                   HINSTANCE hPreInst,
                   LPSTR lpszCmdLine,
                   int nCmdShow )
{
    MSG msg; // Windows message structure
    int Terminate=0;
    CApp *MyApp; //one instance of application
    //create instance of application class
    MyApp = new CApp(hInst);
    //set up new maze
    if(!MyApp->NewMaze()==-1) //problem creating new maze
    {
        ::PostQuitMessage(0);
    }
    //now allow message loop
    while(!Terminate)//!CLOSEGLOBAL && GetMessage(&msg, NULL, 0, 0))
    {
        if(PeekMessage(&msg,0,0,0,PM_REMOVE))
        {
            switch(msg.message)
            {
                case WM_QUIT:
                    if (MyApp!=NULL)
                        delete MyApp;
                    Terminate = 1;
                    break;
                case WM_SIZE :
                    //don't break, go right to paint
                case WM_PAINT :
                    break;
            }
            //process local message
            TranslateMessage(&msg);
            DispatchMessage(&msg);
        }
        else
        {
            if(!CGIWindow::DXGetKey(DIK_ESCAPE)&&(MyApp->MazeFinished()<0))//-1 is continue
            {
                //update and render the maze
                if(!MyApp->Update())//problem rendering
                {
                    ::PostQuitMessage(0);
                }
            }
            else
            {
                //delete old and start new maze
                if(MyApp->NewMaze()==-1) //problem creating new maze
                {
                    ::PostQuitMessage(0);
                }
            }
        }
    }
    return msg.wParam;
}

```



Appendix E Agent Summary

Interaction	Human	DFA	NNET	MOE
Sensors	visual	mathematical		
-FOV	55°			
-view distance	Unlimited within the bounds of the arena. No visual information provided if behind an intervening object (wall).			
-input	-Visual information 30 times per second through computer monitor. Distance and facing information must be interpreted from (non stereoscopic) depth cues. -Auditory information through non-directional speakers.	- visual information as absolute mathematical location information (only if within the FOV above). Distance to walls, opponent facing and location all provided as coordinate and vector values. -Auditory information provided as non-direction boolean flags.		
Effectors	4 arrow keys plus space bar for firing. Keys may be pressed in any combination. Received by the environment as boolean signals.	4 directional plus one firing signal. May be sent in any combination. Not sent through the keyboard, but received by the environment as boolean signals.		

States (software only)	DFA	NNET	MOE
Search	-active when opponent not in FOV		
-‘directed’ search	NA	‘Directed’ search after any recent sensor information. Move directly to opponent’s estimated location until arrival or new sensor information	as NNET, except gating mechanism chooses optimal estimate
-‘blind’ search	‘Blind’ search uses only visible environment information. Active as soon as opponent no longer visible.	‘Blind’ search resumes when estimate proven invalid	as NNET
Fight	-active when opponent in FOV		
-fire	-command to shoot is given when bullet available and currently aligned with preferred target location (align = true)		
-align	-changes direction of agent to face estimated location of opponent. Estimate is current opponent location for no recursion, and triangulated location for one+ recursion	-as DFA, but estimate is based on neural net predicted location.	-as DFA, but estimate is based on MOE predicted location.
-avoid	-turn to avoid bullet if not aligned and opponent firing		
End	-active when either agent is killed. Losing opponent is transported to random location in arena.		

Appendix F Definitions

Agent - An agent is an autonomous entity that interacts in its environment through its sensors (input) and effectors (output).

Recurrent Neural Network - A neural network that is sensitive to patterns over time.

Bayesian Network - A probabilistic learning method that combines prior knowledge with observed data.

Information Gain - The amount of knowledge to be gained by knowing the value of a given attribute. The more random the attribute, the higher the information gain.

Overfitting - Creating a model of the training data that fits so closely that it is less capable of modeling other data.

Markov Model - A description of a process as a series of states. The probability of transition from one state to any other depends only on the current state (no memory).

Deterministic Finite State Automaton (DFA) - An state machine (agent) in that transitions are determined only by the current state (no probabilities).

References

- Astur, R. S., Ortiz, M. L., & Sutherland, R. J. (1998). A characterization of performance by men and women in a virtual Morris water task: A large and reliable sex difference. *Behavioural Brain Research*, 93, 185-190.
- Barron, T. (2000). MultiPlayer Game programming. Prima Publishing, Roseville, California.
- Burns, B & Vollmeyer, R. (1998). Modeling the Adversary and Success in Competition. *Journal of Personality and Social Psychology*. (75 No. 3) 711-718.
- Carmel D. and Markovitch S. (1996). Opponent modeling in a multi-agent systems. In G. Weiss and S. Sen, editors, Lecture note in AI, 1042: Adaptation and Learning in Multi-agent Systems, Lecture Notes in Artificial Intelligence. Springer-Verlag.
- Choi, J. & Silverman, I. (2002). The relationship between testosterone and route learning strategies in humans. *Brain and Cognition*, 50, 116-120.
- Dabbs, J. M., Chang, E. L., Strong, R. A., & Milun, R. (1998). Spatial ability, navigation strategy, and geographic knowledge among men and women. *Evolution of Human Behavior*, 19, 89-98.
- Elman, J. L. (1990). Finding structure in time. *Cognitive Science*, 14:179--211.
- Fischer, G. (2001) User Modeling in Human-Computer Interaction. *User Modeling and User-Adapted Interaction*, 11, 65-86.
- Galea, L. A., & Kimura, D. (1993). Sex differences in route-learning. *Personality & Individual Differences*, 14, 53-65.
- Ghahramani, Z. (2001). An Introduction to Hidden Markov Models and Bayesian Networks. *International journal of Pattern Recognition and Artificial Intelligence*. 15 (1), 9-42.
- Grön, G., Wunderlich, A., Spitzer, M., Tomczak, R. & Riepe, M. (2000). Brain activation during human navigation: gender-different neural networks as a substrate of performance. *Nature Neuroscience*, 3:4, 404-408.
- Hamilton, D.L. (1979). A cognitive-attributional analysis of stereotyping. In L. Berkowitz (ed.) *Advances in Experimental Psychology*.

Harris, L. J. (1978). Sex differences in spatial ability: Possible environmental, genetic, and neurological factors. In M. Kinsbourne (Ed.), *Asymmetrical function of the brain* (pp. 405-522). Cambridge: Cambridge University Press.

Hebb, D. O., & Williams, K. A. (1946). A method of rating animal intelligence. *Journal of General Psychology*, 34, 59-65.

Jacobs, R. & Nowlan, S. (1991). Adaptive Mixtures of Local Experts. *Neural Computation*, (3) 79-87.

Kohonen, T. (1982). Self-organizing formation of topologically correct feature maps, *Biological Cybernetics* 43 (1), 59-69.

Laird, J. (2000). It Knows What You're Going To Do: Adding Anticipation to a Quakebot. Presented at the AAAI Spring Symposium on Artificial Intelligence and Interactive Entertainment, March.

Lawton, C. A. (1994). Gender differences in way-finding strategies: Relationship to spatial ability and spatial anxiety. *Sex Roles*, 30, 765-779.

MacInnes, J., Banyasad, O. & Upal, A. (2001). Watching Me, Watching You. Recursive modeling of autonomous agents. *Abstracts of the Canadian Conference on AI 2001*, Ottawa, Ontario. p 361-364.

MacInnes, J. & McCabe, J. (2001). The Rising Cognitive cost of Automation. *Proceeds of SELF-ACE*, Montreal, Quebec. Oct.

MacInnes, W.J & Taylor, T. (2001). Millisecond Timing on PCs and Macs. *Behavior, Research Methods, Instruments & Computers*, 33 (2), 174-178.

Minsky, M. (1967). *Computation: Finite and Infinite Machines*. Prentice-Hall, Inc., Englewood Cliffs, N.J., 32-66 (Chap. 3).

Moukas, A. (1997). User Modeling in a MultiAgent Evolving System. *Proceedings, workshop on Machine Learning for User Modeling, 6th International Conference on User Modeling*, Chia Laguna, Sardinia, .

Mozer, M. (1993). Neural net architectures for temporal sequence processing. To appear in: A. Weigend & N. Gershenfeld (Eds.), *Predicting the future and understanding the past*. Redwood City, CA: Addison-Wesley Publishing, 243-264.

O'Keefe, J and Nadel, L. (1978). *The Hippocampus as a Cognitive Map*. Oxford: Oxford University Press.

Orwant, Jon. (1995). Heterogeneous Learning in the Doppelganger User Modeling System. *User Modeling and User Adapted Interaction*, 4(2):107-130.

Rich, E. (1979). User Modeling via Stereotypes. *Cognitive Science* (3), 329-354.

Russel, S and Norvig P. (1995). *Artificial Intelligence: A Modern Approach*. Prentice-Hall publishing, 598-600.

Sandstrom, N. J., Kaufman, J., & Huettel, S. A. (1998). Males and females use different distal cues in a virtual environment navigation task. *Cognitive Brain Research*, 6, 351-360.

Shore D., Stanford L. , MacInnes J. , Klein R. & Brown R. (2001). Of Mice and Men: Using Virtual Hebb-Williams mazes to compare learning across gender and species. *Cognitive, Affective and Behavioral Neuroscience*, 1(1), 83-89.

Thagard, P. (1992). Adversarial Problem Solving: Modeling and Opponent Using Explanatory Coherence. *Cognitive Science*. (16) 123-149.

Turing, A. (1950). Computing Machinery and Intelligence. *Mind*, 236, P433.

Voyer, D., Voyer, S., & Bryden, M. P. (1995). Magnitude of sex differences in spatial abilities: A meta-analysis and consideration of critical variables. *Psychological Bulletin*, 117, 250-270.

Voyer, D., Nolan, C., & Voyer, S. (2000). The relation between experience and spatial performance in men and women. *Sex Roles*, 43, 891-915.

Webb, G, Pazzani, M & Billsus, D. (2001). Machine Learning For User Modeling. *User Modeling and User-Adapted Interaction*. (11) 19-29.

Widmer, G & Kubat, (1996). M. Learning in the presence of concept drift and hidden concepts. *Machine Learning*. 29, 69-101.