Proceedings of CGAMES'2006

8th International Conference on Computer Games: Artificial Intelligence and Mobile Systems

24-27 July 2006

Hosted by

Galt House Hotel
Louisville, Kentucky, USA

Organised by

University of Wolverhampton

in association with

University of Louisville

and

IEEE Computer Society
Society for Modelling and Simulation (SCS-Europe)
Institution of Electrical Engineers (IEE)
British Computer Society (BCS)
Digital Games Research Association (DiGRA)
International Journal of Intelligent Games and Simulation (IJIGS)

Edited by:

Quasim Mehdi

Guest Editor:

Adel Elmaghraby

Published by The University of Wolverhampton
School of Computing and Information Technology
Printed in Wolverhampton, UK

Published by The University of Wolverhampton, School of Computing and Information Technology

ISBN 0-9549016-1-4

Contents

Programme Committee

Preface

The Programme Committee is very pleased to welcome all delegates to the 8th International Conference on Computer Games, CGAMES 06 USA. CGAMES is part of Game-On® International Conference of Wolverhampton University, UK. We have the privilege to bring the conference for the second time to the Louisville, Kentucky. The 7th event was held in CNBDI-Angouleme, France which was a huge success and well participated. Our 9th International conference: Artificial Intelligence, Animation, Mobile Systems, Educational and Serious Games, will be hosted by the Dublin Institute of Technology, Dublin, Ireland, 22nd-24th November 2006. The theme has been chosen to reflect the major changes in the way in which games are developed and played.

We are very proud to announce that the conference has successfully maintained its links with its sponsors: *the IEEE Computer Society*, *British Computer Society*, *Digital Games Research Association (DiGRA)*, and the *Society for Modelling and Simulation (SCS)* who where the first sponsors of the conference, we are thankful to them for their help and support.

The conference will endeavour to help the new researchers and MSc/MPhil/PhD research students to present their work to their peers and experts in the field in order to improve the quality of their work in this exciting subject area. The quality of submitted papers has been maintained at a high standard by having them reviewed by our reviewers who have been delighted with the work produced by the authors. Our special thanks go to the reviewers who have been most diligent in their task by providing detailed and useful feedback to authors. The best papers will be reviewed for possible inclusion in the *International Journal of Intelligent Games & Simulation* (IJIGS).

This conference has flourished by the hard work put in by our colleagues in Louisville. Our big thanks and appreciation go to their generosity and time and effort to help organise this conference and provide valuable support. We particularly wish to thank the General Programme Chair, Professor Adel Elmaghraby; University of Louisville and colleagues at the Intellas Corporation, Don Anderson and Dennis Jacobi.

We would also like to thank the School of Computing & Information Technology, University of Wolverhampton, UK, especially the Dean Professor Rob Moreton for his endless support.

We trust that you will all enjoy your stay in Louisville and benefit from this conference by making new contacts for future mutual collaboration.

Quasim Mehdi, General Conference Chair
University of Wolverhampton, July 2006

Proceedings

Session 2
Keynote Presentation

GAMING IN CONTEXT: THE SOCIAL, CULTURAL AND EDUCATIONAL RELEVANCE OF COMPUTER GAMING

Don Anderson
Intellas Group, LLC, 15424 Beckley Hills Drive, Louisville, KY 40245 USA
(502) 254-1601
danderson@intellas.com

Keywords

Games, war games, educational games, multi-player games, role-playing.

Abstract

Myopia is a common affliction among those involved in technology. Simply put it is failing to see the forest because of the trees. As technologists and creators of computer games and computer gaming platforms, it is easy to become too narrowly focused on specific technical issues that apply to our work. This is not to say that these issues are not important and need to be addressed in order for the quality and capabilities of computer games to increase, but rather that we can not afford to lose the perspective of the context in which they exist and operate.

Computer games and game technology are profoundly influencing the world around us in both obvious and subtle ways. Modern cultures and societies are faced with the most rapid rate of change in the course of human history. Change in itself is neither positive nor negative. Rather it is how we frame and direct this change that adds the element of whether or not a particular change or element of change is positive or negative. This paper will attempt to take a snapshot of computer games from this larger context and demonstrate how they are becoming an important and integral part of the world around us.

Background

Gaming had remained relatively unchanged throughout most of the course of human history. Games have been with us since ancient and possibly even prehistoric times. The majority of these games involved two players pitting their skills in a variety of games of strategy. In the Ancient Near East these included such games as Marcala, Seega, Senet and Quirkat. In India the ancient predecessor of Chess was born. In China such games as Go and Mah Jongg appeared. In Ancient Rome in addition to some of the aforementioned Near Eastern games, there were Tali, Tesserae (a dice game), Duodecum Scripta (Twelve Lines), Tabula (a backgammon predecessor), and Latrunculi.

Most of these games were two dimensional and involved strategy of mind on mind between two players. A few functioned as group gambling games, but were still essentially two dimensional. This gaming methodology continued for several thousand years with no essential changes until the modern era.

In a previous study for a United States Air Force project our team identified three generations of war games, which can also be applied to games as a whole. In fact a fourth generation could be added to the original three to define the emergence of

multi-dimensional games over the course of the last ten years. See Figure 1.

GENERATION	FOCUS	SCOPE
First	Strategy	Mind on mind
Second	Attrition	Force on force
Third	Effects	System on system
Fourth ?	Interaction /integration ?	Multi-dimensional?

Figure1. Game generations

After several thousand years of first generation games, the second generation was born through the medium of a new generation of board war games that more accurately modeled conditions on the battlefield through the depiction of military units as individual game counters with attributes of attack and defense strength, movement and unit type. Although the first vestiges of this system appeared during the 17th century using miniatures to depict the combat units, the real origins of these second generation games came in the early 19th century in Prussia when civilians and members of the Prussian Army developed the first detailed and realistic war games. Civilian war gaming in the United States began in 1953, with the development of a game called "Tactics" by Charles Roberts in Baltimore, Maryland. This resulted in the creation of the Avalon Hill Company in 1958 and the release of such games as Tactics II, Gettysburg, and Dispatcher. By 1962 Avalon Hill was selling 200,000 games per year and added such titles as U-Boat, Chancellorsville, D-Day, Civil War, Waterloo, Bismarck and Stalingrad.
The games developed during this time period and still being developed in the present involved "sides" and force on force. Victory came through attrition and the achieving of specific objectives or goals in the game scenario.

In the mid 1970's another phenomena occurred that would impact the course of gaming and the advent of the computer gaming industry. Gary Gygax and Rob Kuntz introduced the first set of rules for a new fantasy role-playing system, "Dungeons & Dragons," and founded TSR Games. Role-playing introduced a system in which any reasonable number of players could enter the game. Play could be set for short periods of time all the way up to campaigns that could go on for days, months or even years. Success or victory was redefined in contrast to the typical board game in which one player emerged victorious. In Dungeons & Dragons success or victory is judged on a more individual level, but also by a particular group to which the player has affiliated him/herself.

Soon after the introduction of the role-playing paradigm, the affordable micro computer made its appearance, and the computer game became a reality. Initially computer games were simply conversions of their board game or war game counterpart. The advent of a variety of different role-playing games along with later advances in game theory applied to traditional two dimensional board games brought the inception of the third generation of games that placed increasing emphasis on the relationship of effects in the outcome of the game.

Although some initial efforts to create text based Massively Multiplayer Online Role-playing Games (MMORG) took place as early as 1978, the real introduction of this phenomena started in the early 1990's and has taken off in the new millennia. The increased accessibility provided by the Internet and broadband connections has

made this evolution a reality as gaming enters a fourth multidimensional generation.

The Computer Gaming Explosion – Relevance on Steroids

Previous game generations and individual styles and paradigms within those generations typically had small, but limited followings of gamers. For example the world of the hard core war gamer is a relatively small and currently shrinking world. Prior to the transference of role-playing to the computer, Dungeons & Dragons also had a limited and loyal following. Parlor board and card games have always had a more general and widespread acceptance, but even these only affected a very narrow slice of people's lives.

Today computer and electronic gaming has grown into an industry that now surpasses the movie industry in drawing people's entertainment dollars. The sales of video games in 2003 increased by a dramatic 25% to $23 billion world-wide (USA Today 2005). Part of the draw is the switch from passive to interactive entertainment, entertainment that now pervades every area and facet of our lives. The former viewer now becomes an active participant in the action – and we haven't seen anything yet! Instead of passively sitting and watching a movie, we can become one of the actors, affecting even the outcome of the story, based on our response to the story world.

The spread of computer/video games to a variety of media formats has been accelerating. Now games are not only available on PC's and dedicated game consoles, but also on a wide range of portable devices including cell phones, PDAs and small mobile game devices such as Game Boy. The introduction and use of mobile devices has been noted by many in their research [Mansour et al. 2005]. However the most noteworthy example of the affect of this explosion on our society is to see my young nephew totally absorbed in his Game Boy.

Social, cultural and educational relevance are evident in the changes taking place in our global society as a result of this explosion. Researchers have observed that the new "game generation" has evolved a new way of thinking which includes multiprocessing, a short attention span, and learning through exploration and discovery [Kaufman et al. 2005, Asakawa and Gilbert 2003, Gee 2003, Prensky 2001] Based on these assumptions today's games provide the ideal environment for learning within this new generation of thinkers.

The Social and Cultural Integration of Computer Games

Social and cultural integration of computer games is part of an even bigger paradigm shift – the Information Age. The revolution and evolution of computer games would not be possible without the advanced platforms that have been developed as a result of the introduction of the microchip and the dawning of the Computer and Information Ages. Also the Internet has made possible the concept of sharing information as needed over vast distances. This technology has become part of every facet of our lives and has become both pervasive and ubiquitous.

Several researchers in their studies have been examining using a concept called Mixed Reality [Cheok et al. 2005]. Mixed reality moves computing away from the traditional keyboard and mouse into the real environment to support more social and

interactive behavior. Mixed Reality combines virtual pieces with the real environment and is in entertainment an example of ubiquitous human media [Dourish 2001].

Magic Land [Ngyuen et al. 2005] [Prince et al. 2002] [Farbiz et al. 2005] is one example of the implementation of a mixed reality environment in which 3D captured avatars of real humans and 3D computer generated virtual animations can play and interact with each other. Users in real space can interact with the virtual world.

The same research group in Singapore also has created a Human Pacman game that uses a mixed reality environment to embed a virtual playground in the natural physical world using such technologies as mobile computing, Wireless LAN and ubiquitous computing [Cheok et al. 2003, 2004].

One of the more interesting applications developed by this group is the Poultry Internet [Lee et al. 2004] which provides a means of human-computer-pet interaction that transfers human petting over the Internet to the pet chicken wearing a special dress to transmit the sensations.

Another use of computing has been the integration of a number of recent television shows with an on-line game component. A specific example is the new reality show "Treasure Hunter" running on NBC in the United States. Each episode integrates a contest to select the correct answer "Treasure Chest" to be entered into a drawing of $10,000 for the evening. Entries are possible either by a cell phone code or over the Internet at the show's web site. Additionally there is a larger on-line treasure hunt in which the on-line participant can attempt to solve the clues to find the ultimate treasure. I am sure that we can expect more of the same.

These are just a few recent examples of the increasing integration of computer games and computer gaming technologies in every facet of our daily lives. These games are becoming a vital part of our society and culture as a whole.

The Educational Impact of Computer Games

New ways of thinking lead to new learning models, as observed by researchers in Canada exploring the use of multi-lingual computer based educational systems for medical training and education [Kaufman et al. 2005]. An interesting observation from this study is the concept of the development of a new learning model that is assisted by simulations and computer games to transform the role of the student from a passive to an active mode via peer collaboration. The student becomes a teacher, both of himself and of others.

Another valuable observation made by the same group is that simulations and computer games also provide the potential for integrating theory, experience and best practice. This was further supported in some of our own interaction with the United States Air Force to propose development of a new training system using a combination of existing game theory along with the concepts of system on system; effects based structures for the training modules.

Another study emphasized the importance of the visual and behavioral fidelity of the avatars or components of collaborative learning-based games [Mansour et al. 2005]. The ability of the graphics in such a system to correctly mimic true human interaction through body positioning, facial expressions, and other typical human behaviors, was observed to greatly influence the effectiveness of such systems.

Even standard off the shelf and on-line computer games have been observed to have influence on students' learning patterns, reading habits and even television viewing habits as observed by a Swedish study [Wiklund 2005]. The study found that students who read regularly prior to gaming still did so, but the surprise was that the ratio of fiction decreased and non-fiction/technical reading increased. Television viewing decreased in importance.

To increase the effectiveness of gaming in education one group in the Netherlands has created a game design tool to create enhanced educationally oriented games [Overmars 2004]. The tool "Game Maker" makes use of object-oriented design concepts to create high quality educational games targeted to specific purposes. The interesting fact is that not only are the products of the system useful in creating games more targeted to educational purposes, but also the process of using the tool is educational in itself to students exposed to it.
In another recent study [Masuch and Nacke 2004], the enormous potential of computer games in education both in teaching and research was found to:

- Motivate students
- Teach interdisciplinary teamwork
- Provide hands on experience in multimedia integration

Other Issues

In a previous study [Anderson et al. 2004] the importance of a change in development methodologies was emphasized to facilitate the interdisciplinary character of current game development. The typical creation of a new video or computer gaming title is more like a Hollywood production than the previous concept of a group of computer geeks developing a program in the basement. Obviously the revenues being generated in this industry warrant this approach.

Which brings up another issue, a powerful argument for the relevance of computer games in our society is simply the enormous financial impact that these games are having world wide. At the onset of computer gaming, most of us couldn't have imagined this outcome in our fondest dreams.

Conclusions and Future Work

This paper has summarized the history of gaming and computer gaming to demonstrate the rapid evolution in the information age from a paradigm, or a group of paradigms, that had limited impact on society and culture as a whole, to the explosion of computer/video gaming into a multi-billion dollar industry impacting nearly ever facet of modern society. Examples were provided to demonstrate both the effects and the direction of this explosion and its integration into the fabric of culture and society as a whole. The educational impact and opportunities are seen to be significant.

Since this change is ongoing and dynamic, it is only speculation to indicate to specific directions. However, it appears obvious that the increasing relevance of computer games will continue to be influenced by and influence the society around us. Subtle variations will exist in the various sub-cultures in our world as a whole. The opportunity is afforded for all of us to celebrate the diversity on our planet and to discover new ways to achieve peace and understanding through shared knowledge.

Future work is virtually boundless, but in the instance of this specific topic can be constrained to observing the direction and opportunities for increasing the relevance of computer games in our society. Granted this does not mean we have to give up the sheer joy of entertainment in gaming, but rather that it affords us to learn as we play – learn to understand one another in a more profound and meaningful way.

References

Anderson, D., Arnold S., Jacobi, D., Mehdi, Q., Gough, N. "Turning a Corner: Games and the Social Content," in Proceedings of 5th International Conference on Computer Games: Artificial Intelligence, Design and Education, Reading, UK, November 2004, pp.301- 305

Asakawa, T. and Gilbert, N. 2003, "Synthesizing Experiences: Lessons to be Learned from Internet-Mediated Simulation Games." *Simulation & Gaming* 34, no.1: pp.10 - 22

Cheok, A., Fong, S., Goh, K., Yang, X., Liu, W., Farbiz, F. 2003. "Human Pacman: A sensing-based mobile entertainment system with ubiquitous computing and tangible interaction." In *NetGames Second Workshop on Network and System Support for Games*, pp. 106 – 117.

Cheok, A., Xu, K., Liu, W., Teo, H., Teo, S., Lee, S., Katai, O., Kawakami, H., Notsu, A. 2004. "Mixed Reality Media for Social and Physical interaction." In Proceedings of International Conference on Artifical Reality and Telexistence, pp. 79 – 84.

Cheok, A., Li, C., Nguyen, T., Qui, T., Lee, S., Liu, W., Teh, K., Diaz, D., Boj, C. "Social and Physical Interactive Paradigms for Mixed Reality Entertainment," in Proceedings of 7th International Conference on Computer Games, Angouleme, France, November 2005, pp. 8 – 15.

Dourish, P. 2001. "Where the Action is: The Foundations of Embodied Interaction." In *MIT Press*.

Farbiz, F., Cheok, A., Wei, L., ZhiYing, Z., Ke, X., Prince, S., Billinghurst, M., Kato, H. 2005. "Live Three-Dimensional Content for Augmented Reality." In *IEEE Transactions on Multimedia*, vol.7, pp. 514 – 523.

Gee, J.P. 2003. *What Video Games Have To Teach Us About Learning and Literacy.* Palgrave, New York

Kaufman, D., Sauve, L., Ireland, A., "Simulation and Advanced Gaming Environments: Exploring Their Learning Impacts," in Proceedings of 7th International Conference on Computer Games, Angouleme, France, November 2005, pp. 16 - 25.

Lee, S., Farbiz, F., Cheok, A. 2004. "A Human-Pet Interactive Entertainment System Over the Internet."

Mansour, S., Rude-Parkins, C., El-Said, M., " The Relationship Between the Avatar's Behavioral Fidelity and Social Interaction in 3D Collaborative Learning-Based Games," in Proceedings of 7th International Conference on Computer Games, Angouleme, France, November 2005, pp. 60 - 65.

Masuch, M., Nacke, L., "Power and Peril of Teaching Game Programming," in Proceedings of 5th Game-On International Conference on Computer Games: Artificial Intelligence, Design and Education, Reading, UK, November 2004, pp.347 – 351

Nguyen, T., Qui, T., Xu, K., Cheok, A., Teo, S., Zhou, Z., Mallawaarachchi, A., Lee, S., Liu, W., Teo, H., Thang, L., Li, Y., Kato, H. 2005. "Real Time 3D Human Capture System for Mixed-Reality Art and Entertainment." *Visualization and Computer Graphics, IEEE Transactions on 11* (November – December), pp. 706 – 721.

Overmars, M., "Game Design in Education," in Proceedings of 5th Game-On International Conference on Computer Games: Artificial Intelligence, Design and Education, Reading, UK, November 2004, pp.14 – 18

Prensky, M. 2001. *Digital Game-Based Learning*. McGraw-Hill, New York.

Prince, S., Cheok, A., Farbiz, F., Williamson, T., Johnson, N., Billinghurst, M., Kato, H. 2002. "3D Live: Real Time Captured Content for Mixed Reality." In *International Symposium on Mixed and Augmented Reality (ISMAR)*, pp.7 – 13.

Wiklund, M., "Behavioural Changes in Students Participating in an Upper Secondary Education Program Using Unmodified Computer Games as the Primary Teaching Tool," in Proceedings of 7th International Conference on Computer Games, Angouleme, France, November 2005, pp. 66 – 73.

.

Session 3

TOWARDS ONLINE ADAPTATION IN ACTION GAMES: CASE STORAGE AND RETRIEVAL

Thomas Hartley and Quasim Mehdi
School of Computing and Information Technology
University Of Wolverhampton, UK, WV1 1EL
E-mail: T.Hartley2@wlv.ac.uk

KEYWORDS

Online adaptation, case storage and retrieval, k-d trees.

ABSTRACT

In this paper we present our work towards the development of an online learning and adaptation architecture for non-player characters (NPCs) (agents) in first person shooter (FPS) computer games. We will outline the development of our case storage and retrieval method, which uses an adaptive k-d tree based approach and discuss the issues related to employing this technique for online storage and retrieval of cases. We conclude by evaluating the performance of the developed data structures and discussing results.

INTRODUCTION

Action games traditionally take place in a 3D environment, in which the player interacts through the control of an avatar. These games usually involve running around and using deadly force against an enemy (Laird and Lent 2001). This typically means a player can shoot an opponent using a variety of weapons, which they pick up from the game world. Players can also pick up other items such as health packs, ammo and bonuses (e.g. shield, mega damage etc). The player observes the game world from a first person (the head position of the avatar) or third person perspective (behind the shoulder of avatar). Examples of action games include Half Life, Quake, Unreal and Tomb Raider.

Currently opponent behaviour in commercial action games tends to be achieved through pre-programmed scripts, finite state machines (FSMs) and the A* path finding algorithm. The use of these techniques has yielded impressive results. However by their nature scripts and FSMs are rigid, predictable and cannot adapt / generalise to circumstances that were not anticipated by the artificial intelligence (AI) programmer (Fairclough *et al.* 2003). In future, game developers may need to employ more sophisticated academic AI techniques, which enable them to make NPCs in their computer games more human-like and responsive to the game player. In Hartley *et al.* (2005) an online adaptation architecture for interactive simulations (specifically FPS computer games) has been proposed as one AI approach that can be used in games development. Online learning through interactions with a human user is one of the key research directions for intelligent agents. Traditionally computer game AI has not made use of unsupervised online learning (Spronck *et al*. 2003); however there has been a certain amount of research in the area (Dinerstein and Egbert 2005, Spronck *et al*. 2003). All these works produce more believable behaviour for NPCs and as Spronck *et al*. (2003) comments; they demonstrate the potential of this approach for improving the entertainment value of commercial computer games.

In this paper we concentrate on the storage and retrieval approach used in our online action prediction system (Hartley *et al.* 2005). We develop an adaptive k-d tree based technique, which employs a hybrid split strategy to build trees online as cases are observed. An optimisation technique is also developed, which rebuilds trees as they grow out of balance. The remainder of this paper is organised as follows: In the next section we discuss related work and the proposed action and behaviour prediction architecture. After that we outline the development of our k-d tree based storage and retrieval method. We introduce and discuss our hybrid split strategy and our optimisation approach. We conclude by discussing our initial results and future work.

ONLINE ADAPTATION SYSTEM OVERVIEW

The architecture outlined in Hartley *et al.* (2005) expands on existing work in the area of online behaviour adaptation. Work in this field has made use of a variety of approaches, such as prediction, user modelling, anticipation, reinforcement learning and plan recognition (Hartley *et al.* 2005). This work builds upon existing incremental case based approaches to modelling an observed entity in order to predict their behaviour (Dinerstein and Egbert, 2005, Fagan and Cunningham, 2003 and Kerkez and Cox, 2001), and makes a number of novel contributions to improve the capabilities of the system. Specifically our architecture offers a more comprehensive state representation, behaviour prediction, and a more robust case maintenance approach, including case storage and retrieval. One of the most interesting approaches to prediction / anticipation in computer games that has

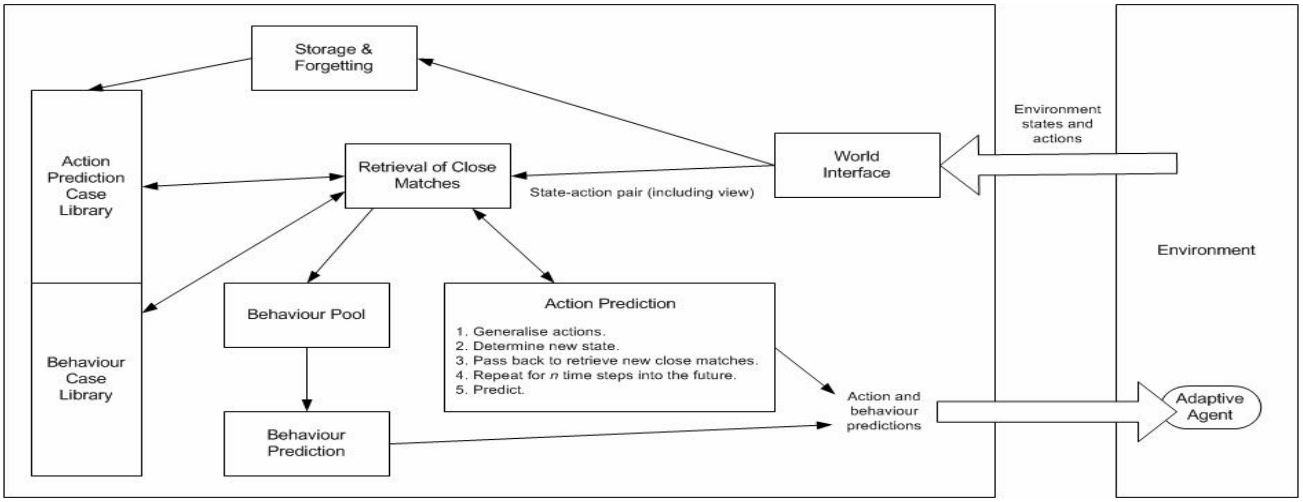


Figure 1: High-level overview of the proposed system architecture

inspired our work can be found in Dinerstein and Egbert (2005). Their system has been designed specifically for use in real-time interactive simulations, it provides the ability to model an observed entity and predict their actions.

Figure 1 illustrates a high-level overview of the proposed action and behaviour prediction layer of our adaptation architecture. The input for the system comes from observed environment events and the output is action and behaviour predictions. The world interface module converts environment input into state-action pairs. Once the input has been properly formatted the system retrieves close matches to the input from the case libraries. The action prediction library returns cases, which are close to the query state-action pair. The behaviour case library returns cases, which contain similar states to the query states. These are then passed to the behaviour pool and actions prediction modules, which are independent processes and will run in parallel (i.e. different threads) or sequentially. The action prediction module generalises actions and determines a new predicted state based on the generalised action. The predicted state is then passed back to the retrieval module in order to determine close matches to it. The process, between the retrieval and action prediction modules, is repeated for *n* time steps into the future, the predicted state is then passed to an agent upon the completion of this process. Once predictions have been made the storage and forgetting module updates the case libraries using a case maintenance policy.

Case Maintenance

In order to guarantee the performance of our action prediction case library, a case maintenance policy is required. Too many cases in the library can significantly slow down the system and lead to poor prediction and generalisation (Kolodner 1993). In addition deleting cases (or "forgetting") is very important when learning something as non-stationary as human behaviour (Dinerstein *et al*. 2005). In Dinersteins' work the number of cases in a region of the state space is fixed and cases are selected for replacement based on their age and unimportance (i.e. their similarity to a new case being added). This approach is satisfactory, however limiting the number of cases in a region of the state space can prevent the system from learning behaviours and as stated above having too many cases will reduce the prediction quality and speed of the system.

In order to keep the action prediction case library at optimal performance we will use the following procedure:

- If the closest library case to a query case exceeds a predefined upper threshold the query case will be added.
- If the closest library case to a query case is below a lower threshold, the query case's view (see next section) will be compared to the library cases view(s). If their views are dissimilar the associated view and action of the query case is attached to the library case, if they are similar the query case is discarded.
- If the closest library case to a query case is in between the lower and upper threshold (as illustrated in Figure 2), all cases in this area will be evaluated using a metric. The library case with the lowest returned metric value will be removed and the query case added. We formally define the case replacement policy as follows:

$$\text{if } M(\mathrm{s},\mathrm{a}) < \mathrm{r} \text{ then replace all } (\mathrm{s},\mathrm{a}) \text{ with } (\mathrm{s}_t,\mathrm{a}_t).$$

Where (s,a) is a case in the case library, $(\mathrm{s}_t,\mathrm{a}_t)$ is the query case and r is a predefined threshold value. The metric *M* is defined as follows:

$$M(\mathrm{s},\mathrm{a}) = -\alpha \times \mathrm{age} + \beta \times ||(\mathrm{s},\mathrm{a}),(\mathrm{s}_t,\mathrm{a}_t)|| + \delta \times \mathrm{useful}$$

This replacement metric is based on Dinerstein *et al*. (2005) approach; however we have also incorporated "usefulness", which defines how often the case has been used in prediction. A case which is used more often will more likely be retained in the case library. For example the simple usefulness metric: $\text{useful} = \text{age} \div \text{times used}$ could be implemented. The time since the case was last used in prediction could also be taken into account.

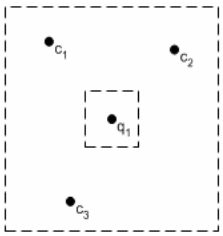


Figure 2: A lower and upper threshold for a 2 dimensional case base. q_1 is the query case and c_1 to c_3 are library cases

CASE STORAGE AND RETRIEVAL

Incremental case-based techniques are used in our proposed system (Hartley *et al.* 2005) to model an observed entity. State-action pairs are treated as cases and stored in a case library. States and actions are made up of *n*-dimensional feature vectors. We have proposed a dual state representation, consisting of primary and secondary state definitions. The primary state space is relatively compact and is used for the main searching and indexing of the system. Additional (or secondary) state information is stored in environment views that correspond to specific primary states and is used to provide a more comprehensive match of states stored in the case library to query states. As cases are observed a case maintenance policy (See previous section) is used to determine if a case will be stored in the library.

The case library is used in the action adaptation layer to predict an observed entity. Prediction is achieved through the nearest neighbour (Mitchell 1997) algorithm, which determines close matching library cases to a query case. A states associated action is then used to predict an observed entity's action. A naive nearest neighbour algorithm could be used for prediction, where every instance in the case-base is examined. However this approach is only suitable for small case bases as it has a time complexity of *O(N)*, where *N* is the size of the case base. Spatial access methods are used to structure the case base more efficiently, in order to avoid examining every case in the library and improve retrieval times.

ADAPTIVE K-D TREES

One of the most well-known main memory *d*-dimensional data structures is the k-d tree (Bentley 1975). The idea behind k-d trees is to partition the state space into disjoint regions by means of iso-oriented hyperplanes that pass through at least one data point. The direction of the hyperplane alternates between dimensions (attributes) at each level of the hierarchy and acts as a discriminator. The data points represent nodes in the tree (Gaede and G¨unther 1998).

k-d trees have a number of disadvantages in that they are unbalanced, sensitive to the order in which points are added and data is scattered throughout the tree (Gaede and G¨unther 1998). The adaptive k-d tree (Bentley and Friedman 1979) overcomes these deficiencies by choosing a split point that divides data into about equally populated partitions. The splitting hyperplanes are still parallel to axes, but they need not contain a data point and their directions do not have to strictly alternate (Gaede and G¨unther 1998). Interior nodes of the tree contain the dimension and the position of the split. Data is stored in terminal / leaf nodes. A leaf can contain a list of points, usually up to a fixed number. Figure 3 illustrates an example adaptive k-d tree for a 2-dimensional state space, with a maximum leaf node size of 1.

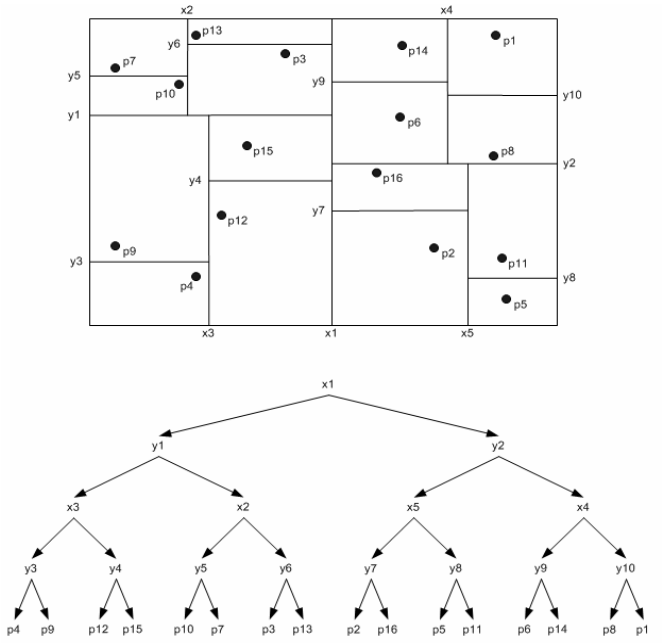


Figure 3: Adaptive k-d Tree

The adaptive k-d tree is a rather static structure, which is difficult to keep balanced when faced with regular insertions and deletions (Gaede and G¨unther 1998). However there are a number of techniques, which can be used to mitigate / overcome these limitations.

BASIC CONCEPTS OF OUR APPROACH

The idea behind our solution for fast case retrieval is a tree structure that partitions the state space into disjoint cells with each leaf encompassing a region of the

space. Leafs (or buckets) are of fixed size and store cases that are contained within its region. Decomposition will be achieved through point based adaptive k-d trees, which are built online as cases are stored. The split dimension of the tree will be rotated (i.e. x, y, z, x…) and the split position can be chosen locally optimal (i.e. optimal to the cases in the bucket to be split) or independently of the data.

This type of data structure is called semi-dynamic as it allows weak insertions and/or deletions. Weak updates can be viewed as updates that do disturb the balance of the tree, but not too drastically (Overmars 1981). Semi-dynamic data structures can be transformed into dynamic ones by building a new structure as the old one degrades. This process is referred to as dynamization (Overmars and Leeuwen 1981). The idea is that neither insertions nor deletions actually need to restore the "balance" of a data structure immediately, as long as the structure remains "in reasonable shape" (Overmars and Leeuwen 1981). Since our case maintenance policy replaces cases within a region of the state space, rather than making arbitrary deletions our system fits well with this idea.

CONSTRUCTION OF THE K-D TREE

In this section we discuss our tree approach in more detail and explain the process involved in inserting a new case. Since our adaptive k-d tree is built online and faces regular updates we need a fast approach that keeps the tree relatively balanced. We opted to build the tree as cases are added, using a construction technique that is akin to LSD-Trees (Henrich *et al*. 1989). Inserting a case involves traversing the tree to find the bucket, which corresponds to the region of the state space that encompasses the new case. The case is then added to that bucket. After a number of insertions a bucket reaches its limit and a new insertion results in it being split, which divides its region into two sub regions. Initially one bucket corresponds to the entire state space. When a split occurs a bucket changes into an internal node and generates two child buckets that store its cases, according to a split strategy. The new internal node contains the dimension and position of the split.

The split strategy involves selecting a split position and location. In general two categories of split strategies can be distinguished (Henrich *et al*. 1989). Data dependant split strategies depend on the case stored in the structure, for example using the median cut along a dimension with the widest distribution of cases. Distribution dependant split strategies choose the split position independently of data. For example a uniform case distribution could be assumed and a buckets region could be split into equal sizes (Henrich *et al*. 1989).

To reduce processing requirements we use the traditional approach of rotating around the split dimension (i.e. x, y, z, x, …), however selecting a split location is more complicated. Data dependant strategies can result in a skewed tree if data is inserted in pre-sorted order (Henrich 1996). Whereas distribution dependant split strategies require a hypothesis of the data distribution to be formulated prior to the tree being built (Henrich 1996). Since cases in our proposed system are observed online, based on an agents relative position to an observed entity, it is difficult to accurately determine data distribution before hand. In addition as cases in our proposed system are sequences of relative positions they will appear in a geometric order, rather than randomly. As a result we have used a hybrid split strategy, which attempts to combine the best of both approaches by adapting the choice of split location to avoid degradation of the data structure.

Limited Redistribution

We implemented a hybrid split strategy called limited redistribution (Henrich 1996), which works as follows. If a bucket cannot accept another case we attempt to redistribute a case from the bucket to its sibling (I.e. the

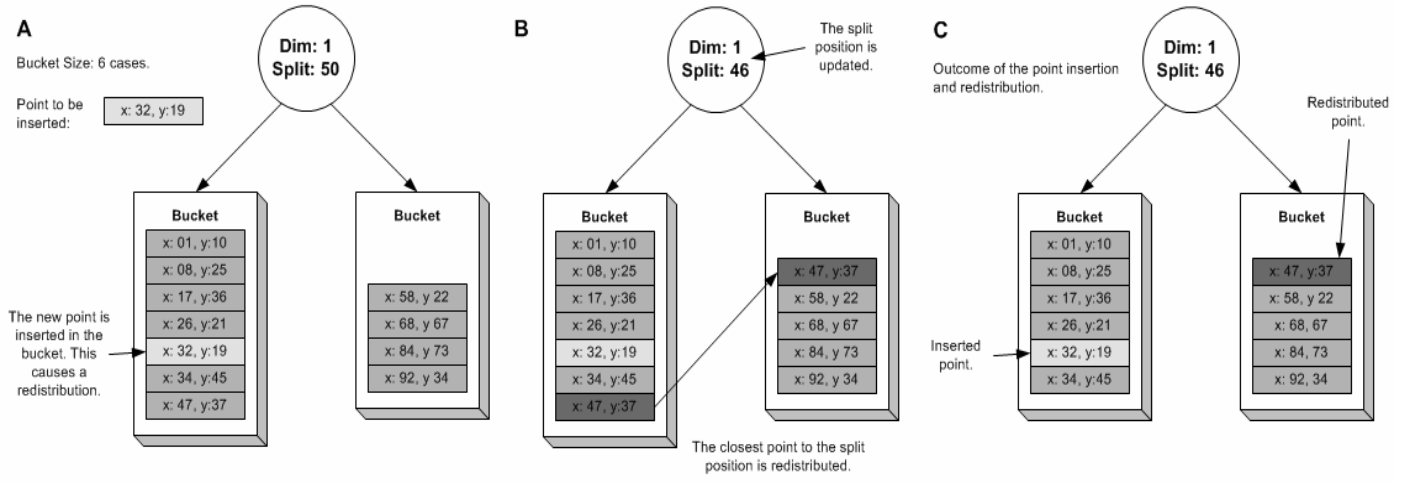


Figure 4: Case Redistribution

other child of the bucket's parent node), rather than splitting the bucket. If the sibling bucket can accommodate the new case without splitting, a local redistribution is performed, which shifts the case that is closest to the split line from the bucket into its sibling and adjusts the split line in its parent node (Henrich 1996). Figure 4 illustrates this process.

By using this approach the originally set split line can only be considered preliminary, since it is adjusted as cases are added. A consequence of this is that the data structure becomes sensitive to the insertion of presorted cases (Henrich 1996). Therefore Henrich (1996) suggests stopping redistribution for those paths that are in danger of degrading due to presorted insertions. He formally defines this process as follows. Let β denote the number of buckets in the k-d tree and let l denote the length of the path from the root node to the bucket. Limited redistribution is only performed if $3 < l - \log_2 \beta$. Redistribution is therefore only performed when the path from the root node to the bucket being split is short enough (i.e. considered to be balanced, based on the number of buckets in the structure).

OPTIMAL TREES: A FULLY DYNAMIC K-D TREE

As discussed above the drawback to our construction method is that it only supports weak updates. In order to stop the tree degrading drastically we developed an optimisation routine, which rebuilds a balanced tree online. To this end, we based our design on dynamization work by Overmars (1981) and Overmars and Leeuwen (1981). Overmars (1981) developed a dynamization scheme that transformed semi-dynamic data structures into dynamic ones by maintaining multiple static structures. They demonstrate that a new structure can be constructed and take over from the old one in a set number of updates, if enough processing time is available.

Our k-d tree dynamization process works as follows. The case base at most consists of 2 structures, OLD-MAIN and MAIN, however normally only MAIN exists (Overmars 1981). As updates occur the tree slowly becomes imbalanced. When the number of insertions and deletions become half the structures initial size we check if $l - \log_2 \beta > 3$, where l is the longest path length in the tree. When this becomes true, MAIN is made into OLD-MAIN and a new MAIN is started. At this point we assume MAIN had n_0 cases and that the new MAIN can take over from OLD-MAIN in λn_0 updates, where λ is some factor, which in this case should be less than half the size of the initial case-base ($\lambda < ½\ n_0$). Otherwise the structure may need to be rebalanced before it is rebuilt. In addition we have added the check $l - \log_2 \beta > 3$, so that the tree is only rebuilt when the current structure is actually out of balance. Overmars (1981) assumes the tree will need rebuilding when the number of updates reaches half its initial size, however this may not always be the case.

Building the new MAIN consists of two steps, building the tree and inserting buffered updates. For this process we use the following notation, where S is a structure containing n points (Overmars 1981). $I_S(n)$ is the time it takes to perform a weak insertion, $D_S(n)$ is the time it takes to perform a weak deletion and $P_S(n)$ = the time it takes to build a tree.

Step 1. For some time we will do the following:

- Continue to update OLD-MAIN, so we have an up to data structure.
- Spend $I_S(n_0 + \lambda n_0) + P_S(n_0) / \lambda n_0$ time building the new tree with every insertion and $D_S(n_0 + \lambda n_0) + P_S(n_0) / \lambda n_0$ time building the new tree with every deletion.
- Store each update in a buffer BUF. The updates are inserted into the new tree when it is built.

Step 2. For some more time, until BUF is empty we shall do the following:

- Continue to update OLD-MAIN, so we have an up to data structure.
- Spend $I_S(n_0 + \lambda n_0) + P_S(n_0) / \lambda n_0$ time processing updates in BUF with every insertion and $D_S(n_0 + \lambda n_0) + P_S(n_0) / \lambda n_0$ time processing updates in BUF with every deletion.
- Store each update in BUF if it is not empty, otherwise the update can be inserted in the new MAIN.

Once step 2 is complete the new main can take over from the old main, which is removed from memory. This process shows us how much time we need to spend building the data structure when an update occurs. However additional time can be spent building the tree when no updates are occurring. For example when there are no observable entities in range.

Basic Procedure for Rebuilding the k-d Tree

The tree rebuilding process is similar to the incremental building scheme outlined above. However rather than using a hybrid split policy the median split position of cases is found. The tree is rebuilt to a specified maximum bucket utilisation. For example an 80% bucket utilisation, where each bucket can store at most 10 cases would result in a data structure where each bucket had a maximum of 8 cases. Specifying a less than 100% bucket utilisations would reduce the impact of buffered and new cases on the tree, however it would increase the trees size and memory usage.

The basic recursive procedure for rebuilding the k-d tree is described below. Initially the method, BUILDKDTREE, is called to begin the process. This function creates a root node containing all the cases in the case base and calls a recursive algorithm, OPTIMISEKDTREE, which performs the rebuild. Every node within the tree represents a subset of the case base and the root node represents the whole case base.

Algorithm BUILDKDTREE (*C*)
Input: *C*, a set of cases.
Output: The root node of a k-d tree storing *C*.
1. Create new root node *V*.
2. *V*.ADD(*C*)
3. *V*.OPTIMISEKDTREE
4. **return** *V*

Algorithm OPTIMISEKDTREE()
Input: None.
Output: None.
Preconditions: *C*, a set of cases.
splitDimension, the current split dimension.
1. **if** *C* contains less than the maximum threshold
2. **then return**
3. /* Determine the new split dimension from the current split dimension. */
newDimension ← NEXTDIMENSION(*splitDimension*)
4. Split *C* into two subsets with a line *l* through the median coordinate on the dimension *splitDimension*, of the points in *C*. Let C_1 be the set of points to the left of *l* or on *l*, and let C_2 be the set of points to the right of *l*.
5. Compute the size *S* of the two subset regions created in step 4. Let S_1 be the region equal to and to the left of *l* and let S_2 be the region to the right of *l*.
6. left ← CREATENEWCHILD(R_1, C_1, *newDimension*, *l*)
7. right ← CREATENEWCHILD (R_2, C_2, *newDimension*, *l*)
8. left.OPTIMISEKDTREE()
9. right.OPTIMISEKDTREE()
10. **return**

The algorithm described above uses two sub-procedures NEXTDIMENSION and CREATENEWCHILD. The NEXTDIMENSION procedure determines which dimension should be used for the next split. In our system the dimensions are rotated however, other approaches (such as, the dimension with the widest distribution of points (Henrich *et al.* 1989)) can be used. The CREATENEWCHILD procedure generates a new instance of a node according to the passed values.

The most time consuming part of the algorithm is finding the median split position as it requires linear time. The algorithm built time T(*n*) for a set of *n* points is therefore:

$$T(n) = \begin{cases} O(1), \text{if } n = 1 \\ O(n) + 2T([n/2]), \text{if } n > 1 \end{cases}$$

Hence a tree for a set of *n* points can be constructed in O(*n* log *n*) time.

IMPLEMENTATION AND RESULTS

Our system is implemented in Java and follows closely the design outline here and in Hartley *et al.* (2005). Games agents are visualised in Quake 2 through the QASE API (Gorman *et al.* 2005), as illustrated in Figure 5. A detailed discussion of the full system implementation is reserved for future work. Here our experimentation focuses on evaluating the data structure and its performance to determine its capability for use in an online case based system such as ours.

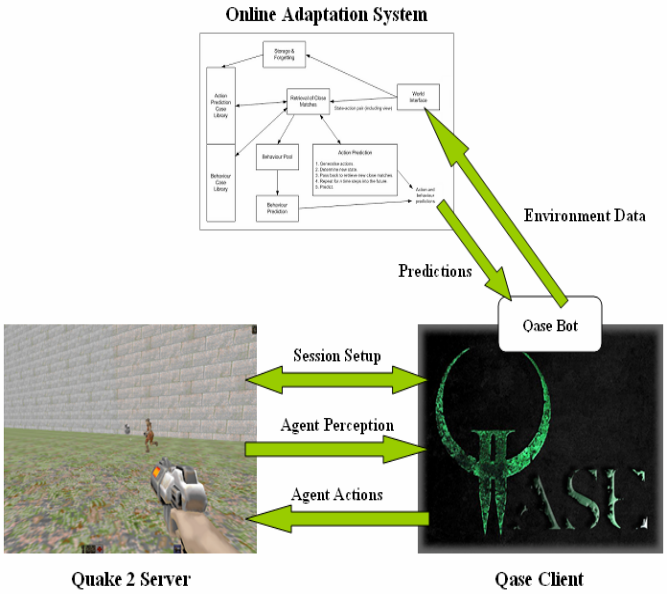


Figure 5: The proposed online adaptation system connected to Quake 2 through the QASE API (Gorman *et al.* 2005)

Since the development of the data structure was motivated by its ability to perform well online and adapt to cases, it is essential to analyze its performance in this area. Initial experiments focused on how our algorithms scale and optimisation times.

Table 1 outlines the relationship between the number of cases in a case base and the build time that is required to create a balance tree with the optimisation routine. A visual representation of our tree structure is pictured in Figure 6. Uniformly distributed 4-dimensional points were for our experiments. The points were presorted with respect to their distance to the point 0, 0. For all experiments the average time of 3 builds was taken, the maximum bucket capacity was set to 7 and the rebuild bucket capacity was set to 100% (i.e. 7 cases). A 3.0 GHz Pentium 4 processor was used, with 512MB of RAM.

Number of Cases	Average Build Time (ms)
500	41
1000	88
1500	156
2000	260
3000	542
5000	1526

Table 1: Build time to create a balanced tree. Where ms = milliseconds.

The results in table 1 indicate that the build time for 5000 cases is relatively large, about 1.5 seconds. By using the dynamization approach outlined in the previous section this time can be spread over multiple updates to the structure, thereby reducing its impact on the system. However dynamization would not be worthwhile when only a small number of cases exist (i.e. less than 500) as the build time is very fast. In addition our O(n log n) time construction analysis in the pervious section matches the results in table 1 quite closely.

The QASE API requires an agent to make a decision every 100 milliseconds (i.e. 1 frame), however a case would not need to be stored this often. Storing a case every 500 milliseconds seems an appropriate approach. If dynamization is used our initial results would appear to fit well within this time scale, however further implementation and analysis is required. For example prediction time and its CPU usage also needs to be taken into account.

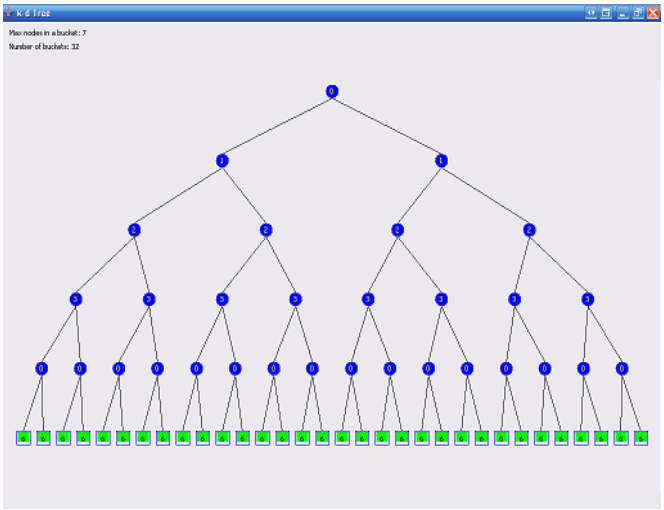


Figure 6: A visual representation of our k-d tree.

CONCLUSIONS

In this paper we have proposed a case storage and retrieval technique for our online prediction system. An adaptive k-d tree based approach was developed, which employs a hybrid split strategy to build trees online as cases are observed. An optimisation technique was also developed to rebuilds trees as they grow out of balance. The approach offers better adaptation to the distribution of cases than fixed partitioning, such as the technique used in Dinerstein and Egbert (2005), as the data structure built based on cases in the case base.

The initial experimental results show that the build time of trees is relatively slow and cannot be done every frame of a game loop. However when incorporated with the dynamization technique outline above this limitation is mitigated. Future work includes fully implementing the online prediction system and dynamization approach, including a more in-depth evaluation.

REFERENCES

Bentley, J. (1975) Multidimensional Binary Search Trees Used for Associative Searching, Communications of the ACM, 18 (9), 509-517.

Bentley, J. and Friedman, J. (1979) Data Structures for Range Searching, Computing Surveys, 11 (4), 397-409.

Dinerstein, J. and Egbert, P. (2005) Fast multi-level adaptation for interactive autonomous characters. ACM Trans. Graph. 24, 2 (Apr. 2005), 262-288.

Fagan, M. and Cunningham, P. (2003) Case-based recognition in computer games. Technical Report TCD-CS-2003-01 Trinity College Dublin Computer Science Department.

Fairclough C., Fagan, M., MacNamee, B. and Cunningham, P. (2001) Research Directions for AI in Computer Games. Technical report, Trinity College Dublin, 2001

Gaede, V. and G¨unther, O. (1998) Multidimensional access methods. ACM Computing Surveys, 30(2):170–231, 1998.

Gorman, B., Fredriksson, M., and Humphrys, M. (2005) QASE - An integrated API for imitation and general AI research in commercial computer games. In Proceedings of 7th *International conference on Computer games: Artificial intelligence, animation, mobile, educational, and serious games* (CGAMES), Angoulême, France.

Hartley, T., Mehdi, Q. and Gough N. (2005) *"Online Learning from Observation for Interactive Computer Games"*, 6th International Conference on Computer Games: AI and Mobile Systems CGAIMS 2005, July, Louisville, Kentucky, USA.

Henrich (1996) "A Hybrid Split Strategy For k-d-Tree Based Access Structures." 4th ACM Workshop on Advances in GIS. New York: ACM Press, 1996. 1-8.

Henrich, A., Six, H. and Widmayer, P. (1989) The LSD tree: Spatial access to multidimensional point and non-point data. In P. M. G. Apers and G. Wiederhold, editors, *Proceedings of the 15th International Conference on Very Large Data Bases*, pages 45–53, Amsterdam, The Netherlands, August 1989.

Kerkez, B. and Cox, M. (2001) Incremental Case-Based Plan Recognition Using State Indices, Cased-based Reasoning Research and Development: Proceedings of 4th International Conference on Case-Based Reasoning (ICCBR 2001), Aha, D.W., Watson, I., Yang, Q. eds., pp. 291-305, Springer-Verlag, 2001.

Lent, M. and Laird, J (2001) Learning procedural knowledge through observation. In: International conference on Knowledge capture, Victoria, British Columbia, Canada, ACM Press (2001) 179–186.

Mao, W. and Gratch, J. (2004) Decision-Theoretic Approach to Plan Recognition, ICT Technical Report ICT-TR-01-2004.

Mitchell, T. (1997) Machine Learning. McGraw-Hill.

Overmars, M. (1981) Transforming semi-dynamic data structures into dynamic structures. Technical Report RUU-CS-81-10 Institute of Information and Computing Sciences, Utrecht University.

Overmars M. and Leeuwen J. (1981) Worst-case optimal insertion and deletion methods for decomposable searching problems. Inform. Process. Lett., 12(4):168-173, 1981.

Spronck P., Sprinkhuizen-Kuyper I. and Postma E. (2003). Online Adaptation of Game Opponent AI in Simulation and in Practice. *GAME-ON 2003 4th International Conference on Intelligent Games and Simulation* (eds. Quasim Medhi, Norman Gough and Stephane Natkin), pp. 93-100.

AI-TEM: TESTING AI IN COMMERCIAL GAME WITH EMULATOR

Worapoj Thunputtarakul and Vishnu Kotrajaras
Department of Computer Engineering
Chulalongkorn University, Bangkok, Thailand
E-mail: worapoj.t@student.chula.ac.th, vishnu@cp.eng.chula.ac.th

KEYWORDS

Testbed , Artificial Intelligence, Commercial Game.

ABSTRACT

Many artificial intelligence (AI) game researchers find that it is difficult to find a game environment that they can appropriately test their AI on. They usually have to develop parts of an existing game, using tools that come with the game. Some even have to re-write their testing game from scratch. Finding a perfect game environment that one can use to test his AI is not easy, especially if a commercial-quality game is required. Huge amount of time and effort are lost in finding such ideal testbed. This paper presents AI-TEM environment, a testing environment for testing AI by using console game emulator and its ROM data to simulate and run a commercial game. AI-TEM can be used to plug many AI onto many commercial games. Researchers interested in higher-level abstractions of game AI can test their already developed AI algorithm on a commercial game. We believe that AI-TEM adds a wider range of possibilities to AI testing.

1 INTRODUCTION

Research and developments related to computer games have always focused on graphical technology. However, players have begun to demand for more playability. Recent games have incorporated smart AI into their gameplay and became very successful because of that. Therefore, research in AI is important for the game industry. On the other hand, games provide interesting testing environments for AI researchers.

One problem faced by many researchers is to find or develop a proper game environment to use in their AI testing (Graepel et al 2004, Kendall and Spoerer 2004, Ponsen et al 2005). A game that should be used to test AI should have the following qualities: It should be a game that has many ways to play, many ways to win, and the game should be complex enough to separate expert players from novice players. It should be a commercial quality game. Because if researcher's AI can win against its initial game AI, then researchers can claim that the newly developed AI truly has enough quality and efficiency to use in commercial game (Spronck et al 2004).

There are testbeds developed for testing AI (Aha and Molineaux 2004, Bailey and Katchabaw 2005), but many of them do not come with a complete game ready to be used. Researchers will have to find a game or game engine to integrate with it. Large amount of time may be used.

This paper proposes another way to test game AI in an environment that has been well made and well designed, called AI-TEM (AI-Testbed in EMulator). AI-TEM uses an emulator of Game Boy Advance (GBA), developed to test many AI methods.

In this paper, an emulator means a game console/handheld emulator that simulates the working of game console/handheld hardware such as GBA, PlayStation, and arcade machine on any personal computer. There are many emulators of console/handheld game hardware. VisualboyAdvance (VBA) (VisualboyAdvance 2005) and VisualboyAdvance Link (VBA Link) (VisualboyAdvance Link 2005) are GBA emulators. ePSXe is a PlayStation emulator. Even arcade machines have MAME as their emulators.

ROM (Read Only Memory) is the game data dumped from the original game cartridge or disc. Using a game ROM with its emulator, a game can be simulated and played on PC.

We used GBA emulator, VisualboyAdvance, for developing a prototype of AI-TEM. And we used Street Fighter Zero 3 (STZ3) game ROM as our test ROM, so that we could experiment and write additional tools for a real example. VBA is open-source, therefore we can modify its functions. There are many interesting games on GBA that can be used to test AI. VBA also has plenty of resources, technical documents and support tools provided for us. The VBA Link is an extended version of VBA. The VBA Link team modifies original source code to make linkage possible. In this paper we will call both VBA and VBA Link as VBA.

STZ3 is a fighting game. In this type of game, a player must select one character from many characters, and fight one by one with an opponent character. A player must decide what action he will perform in many different situations based on his character and opponent character's status. Therefore, we believe this is a good game for tuning our testbed and for AI research.

2 AI-TEM FRAMEWORK

The concept of AI-TEM is generally simple but there is some low-level work involved. Researcher's AI may need to know game state data, such as object position or character animation, but it cannot access the source code of the game. The AI can only access the source code of the emulator. So we must get the game state data from memory data the emulator is emulating. We can see only a binary (hexadecimal) value of game data that changes in every cycle of a game execution. We must therefore find out which address stores the value that we are interested in such as position, animation, etc. We will use values in those addresses as game state data for AI testing. When we know the game state, AI can be written to react in each situation, by sending a controller input, or forcing memory address value. Using this concept, we can use AI-TEM as an AI testing tool.

Finding each address that stores those game state data is difficult if done manually. Some values can be found easily, while some are rather hard to find. User should have some knowledge about programming in order to be able to identify address more successfully. Some examples of how to find the address of game data are demonstrated below.

Example 1: Finding address of character's health. Starting by identifying all the values used in the game. Then the game is played and the character's health is forced to decrease. The value that represents the character's health should in fact decrease too. All game values are then searched and compared with values before the health decreases. It is common to find many values decreasing. The process should be repeated, with different values of health, until one address is identified.

Example 2: Character's bullet position. The concept is the same as the character's health example. When a bullet moves forward, its position value should increase continuously. But the time period that the bullet is alive is very short. Therefore, repeating the experiment as many times as one wants becomes difficult. If users must press a command every time that they want to find a value, it will not be convenient. A tool that can arrange this situation is needed, such as movie recorder (section 3.0).

We had developed some tools to help finding the address more easily and will describe it in section 3.0 below. There are other techniques to find values that we will not discuss in this paper.

Therefore, in the beginning phase of using AI-TEM, the user must find the address of game state data that their AI module needs to know. AI-TEM is depicted in figure 1 and 2, and discussed in more detail in the sections below.

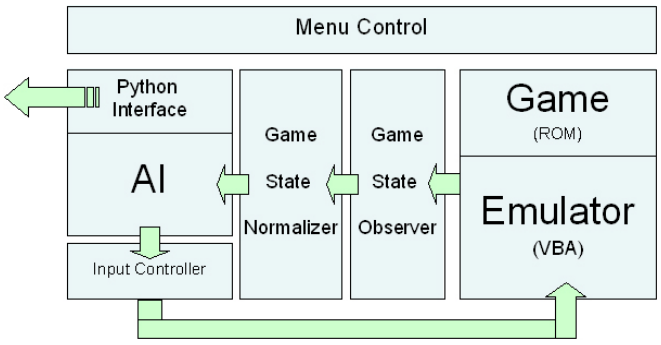


Figure 1: AI-TEM system overview.

2.1 Emulator Core (VBA)

The core of this testbed is VBA (Link) emulator. It is used to run game ROM and simulate the game. It also has many tools useful for getting game state and testing AI. These tools will be described in section 3.1.

2.2 Menu Control

We add a menu into the emulator to control the working of AI-TEM, to turn on/off AI module or switch between different AI modules. We can also activate other utilities that the system may want to use.

2.3 Game State Observer

A game state can be known by observing data on the memory address of the emulator and locating which address stores the data that we want to know. (In STZ3, game states that we are interested in consist of position of a character, position of the character's bullet, the character's health, and current animation of the character.) We implemented this module by modifying the memory viewer tools of VBA. Users can identify addresses and size of data (8, 16, 32 bits) that they are interested in. When AI-TEM is running, in every frame, Game State Observer will copy the values from those addresses to the data structure that an AI module can use.

2.4 Game State Normalizer

Before Game State Observer sends game state data to AI module's data structure, the game state data must be normalized or interpreted, depending on the game and format of data that we obtain from the memory. Example: For STZ3, we use address 0x20007C2 (16bits) as the address that stores the character 1P position in the X axis. The range of value that we got from that address is 44 (002C) to 620 (026C), 576 units. But when using it, we should normalize x position value to 0 - 576 for user friendliness. Therefore, we must subtract 44 from the value copied from the memory of the emulator. This normalization process is not necessary if researchers do not care about the format of raw data from the memory. We currently provide this module as a code template for users to modify.

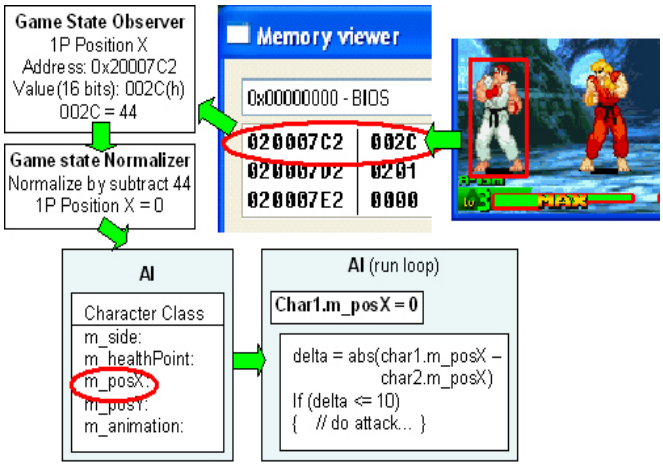


Figure 2: Work flow diagram of AI-TEM system.

2.5 AI

This module is where a user of AI-TEM will put his AI module in. In every cycle of emulation, the emulator will execute this module. This module evaluates the game data and decides what controller input it will send to the controller module. There is enough of VBA CPU power for calculating non-intensive work, such as script (Dynamic script (Spronck et al 2004), Genetic Programming result script, etc.). With extension, other AI methodologies can also be added. In our experiment with the system, we write two static scripts for testing the use of AI-TEM. The detail and result of this script will be shown below in section 4.1.

2.6 Python Script Interface

Python (Python 2006) is an interpreted, interactive, object-oriented programming language. It is also usable as an extension language for applications that need a programmable interface. Python is portable: it runs on many

OS such as Windows and Linux. It is one of the most famous script languages used in many applications.

We modified the emulator to have an ability to use python script language, providing interface functions for a script writer to obtain game state data and to control the game via any AI module. A script writer can write their python script separately without running the emulator, and can change script without recompiling AI-TEM. This will benefit users who want to test their AI with static script. If researchers can generate AI output in python script format, it can be tested conveniently without the need of rerunning the game. Example of an interface function used in the testing of STZ3 is shown below.

```
int GetCharacterPositionX (int C)
int PressButton(int button)
```

The first function will return a position in the x axis of character C. The second function will send a parameter 'button' to the Input Controller module. It allows a python script to command character. Below is the example of python script that uses those interface functions.

```
import myLib
def Main_AI_Run_Loop():
if (myLib.GetCharacterPositionX(P2) <10)
{
    return myLib.PressButton(PRESS_B)
}
return 0
```

`myLib` is a library of interface functions that we provide from AI-TEM, it allows user of python script to use function `GetCharacterPositionX` and `PressButton`. This script results in character kicking (press B) when its opponent comes closer than a specified threshold.

2.7 Input Controller

The original VBA captures signals from joystick or keyboard and send them as input to a game. We modified the system so that our AI module can replace input signals from normal controller with its own signals.

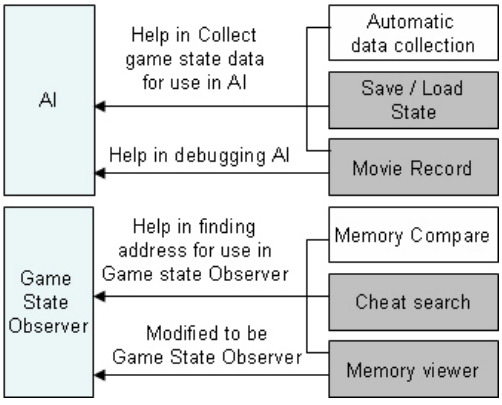


Figure 3: Usage of tools in AI-TEM system.

3 AI-TEM UTILITY TOOLS

Other than modules described in section 2.0, AI-TEM has other utility tools that can help in many tasks. Some of them are original VBA tools that we use in our testbed. Some are modified tools we made for our own use. The working of these tools is shown in figure 3 and described below.

3.1 VBA Tools

These are original VBA tools that we had used in AI-TEM.

Memory Viewer: used for displaying content of every memory data address in a variety of formats, 8, 16, 32 bits, sign, unsigned and hexadecimal. Our Game State Observer is modified from this tool.

Cheat Search: this tool is originally used for finding an address of data that we want to find, by searching all of memory and finding a value that matches a condition given by user. For example, users can use it to find a value in the address that is equal, greater or less than some specific value. In AI-TEM, this tool is used to help in finding address that Game State Observer will observe.

Movie Recorder: this tool can record game movie in two formats. VMV format will record only initial game state and inputs given by controller. It has a very small file but can playback only in emulator. AVI can playback in many movie player programs but its file is larger and uses a lot of CPU power. Movie recorder can help in a data collection process (section 3.2) and can help recording the testing output or debugging.

Save/Load State: When running a game in the emulator, the game state can be saved and reloaded to continue to play at the same point where it was saved. This ability is useful when researchers want to test decision conditions of their AI. They can save game state before their AI makes decision and can reload it to try another decision in perfectly the same situation.

3.2 Modified Tools

Memory compare tool: As said in section 2.0, finding address of value that AI module needs to know is difficult. Therefore, we modified the original VBA tools to be Memory compare tool. This tool will help in finding an address of data, by comparing many game states data, given a condition of data that users are seeking. Example: A user wants to find the address of character's health in STZ3. He will dump game states of various situations from the emulator. We define the game state of situation N as GSn.

```
GS1: character's health is 100%.
GS2: character's health is 50%.
GS3: character's health is 75%.
```

The user will set conditions of the value he wants to find. In this case, a health value address will have conditions as follows: The value in our required address from state GS1 must be greater than the value from state GS2. The value in our required address from state GS2 must be less than the value from state GS3. And the value in our required address from state GS3 must be less than the value from state GS1. This tool will compare every data in those game states and find the address that matches all user-given conditions automatically, without any need to run the original cheat search (The cheat search tool requires users to repeatedly experiment and find any address manually.). If users provide enough game states and proper conditions, finding a required address should be straightforward. We believe this tool can save a great deal of time finding those values.

Automatic Data Collector: There are some situations that we want to collect a lot of data from game state. It is difficult to collect them manually. Example: In order to imitate human reaction and decide its next response, an AI module must know the animation of both characters. For example, if an opponent character is going to punch, our character must detect the opponent's movement and perform a guard. After knowing the address that stores values of character animation, we need to find out which value in that address corresponds to which animation. (for example, 0 means stand, 72 means crouch, 124 means jumping) Therefore, we need some tools to help collecting game data (character animation data).

In STZ3, we modified the original VBA function that was used for forcing values of addresses (Cheat function) to be a tool for helping us to collect animation data. By writing a value from the start animation value to the end of animation value, we forced each character to do all of its actions. We captured the character's image of each action and saved it with its animation value as its file name. We then knew the animation value for each of a character's action. Human intervention was needed to identify the meaning of each action. An animation database was then produced.

4 EXAMPLE EXPERIMENT IN STZ3

This section will discuss the result of using AI-TEM with STZ3 ROM to create a simple, static script AI. Table 1 contains the addresses of STZ3 game state data that we know by the method discussed in section 2.0. Table 2 contains some character animations from a character named RYU, which we collected after normalization, by using stand animation as a basis. (Each animation is composed of many frames, so there are many values in each animation. Figure 4 shows some pictures of animations from table 2.

Table 1**:** Example address of STZ3 game state data.

Game State Data	Address	Data Size
character 1 position x axis	0x20007C2	16 bits
character 1 position y axis	0x20007C4	16 bits
character 2 position x axis	0x20043D2	16 bits
character 2 position y axis	0x20043D4	16 bits
character 1 Animation	0x20007D0	32 bits
character 2 Animation	0x20043E0	32 bits

Table 2**:** Example of character (RYU) animation.

Animation	Value
Stand	0, 12, 24, 36, 48, 60
Crouch	276, 288, 300, 312, 72, 228, 240, 252, 264
Jump	420, 432, 468, 480, 492, 504, 516, 528, 444, 456
Punch (Figure 4)	8484, 8496, 8508, 8520, 8532, 8544, 8556
Kick (Figure 4)	9096, 9108, 9120, 9132, 9144, 9156, 9168

4.1 AI Script Experiment

In order to test our AI module and python interface, we had implemented a static script to control a character in STZ3. Our test condition for our static script is the character RYU VS RYU in versus mode. This static script also allows us to test our AI in a controlled situation. We implemented a static script for character RYU. Our 1P's RYU can detect states of original game AI 2P's RYU.

Our first version of the script just randomly performs action. The result was not as bad as we originally believed. Even though it had no intelligence, it performed action continuously and was able to beat the original game AI at the easiest level. We improved our script in many aspects, using animation data that we collected. Our static AI can now sense distance between characters. We also script it not to use special moves often, since special moves leave characters defenseless. More combination attacks were also added. We obtain a better result, as expected. Our static AI can now beat the original game AI in middle level.

This experiment convinced us that AI can make use of the game state and animation of the opponent by using data in section 4. We also have a fully working python interface ready for creating future controlled situation.

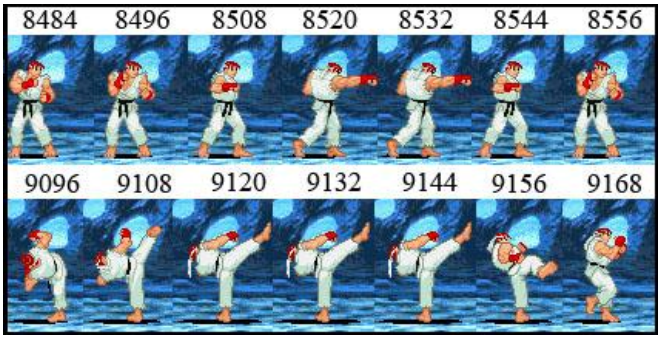


Figure 4: Example pictures of animation values.

5 DISCUSSION

5.1 Outline steps of Using AI-TEM

Researchers must first find a game that is suitable for testing their AI method or matches their experimental plan.

Researchers then identify the game states data that their AI module needs to know. In normal AI method, such as scripting, an AI module needs to know only current situations of the game. But in some AI method such as some type of Reinforcement Learning, it needs to know a complete set of actions that the agent can perform in every situation.

After that, they must find the address of game state data that their AI needs to know.

After the addresses are found, data must be collected from those addresses and translated into a form that the AI can understand.

Finally, AI can be implemented. This topic will be discussed in more detail in section 5.2.

5.2 Which AI method can AI-TEM be used with?

AI-TEM does not limit AI methods that it can be used with, because its concept is only using an emulator as game engine. Some AI methodology, however, requires extra functions. For example, using Genetic Algorithm requires running tests large amount of times, may be hundreds or thousands generation. Therefore, automatic result recorder is needed. High speed running mode will also be an additional welcome, for it can save time to train AI.

High speed mode is already available in VBA and many other emulators. Not all games may allow us to provide automatic running mode. This is because, if we cannot find memory data address that tells us about the beginning and

the end of the game, we cannot force the situation. But in general, automatic result collection can be done. Therefore, various AI techniques can be used.

5.3 What kind of Game/AI-Subject should use AI-TEM?

If researchers are interested in first person shooter, real-time strategies or D&D-style RPG game, there are games that come with tools. Good testing environment for such games can be built with such tools. Also, there are very few of these games on consoles. AI-TEM may not be the first choice for testing such games. If researchers are interested in simple platform action game, writing a game from scratch or finding some open source clone game is not a bad choice because all environments of the game can be fully controlled. However, developing games, from tools provided by a game, or from an open source clone cannot easily get us commercial-quality game. This is where AI-TEM can come in. AI-TEM can be used to test an AI developed on a simple, but fully controllable environment, against real commercial game. In the case of racing game, it is difficult to know game state data such as opponent car position and the track situation. As a result, AI-TEM will not be appropriate. For fighting game, we think that using AI-TEM is suitable, because this type of game is rather difficult to make and even more difficult to make it as good as commercial game. Therefore, we think the tradeoff in the case of fighting game is worthwhile. For sport game, we think that it is still suitable to use AI-TEM, because of the same reason as fighting game. Even though there may be many game states that an AI module needs to know, finding them may be easier than creating a high quality sport game from scratch. Some AI researchers use Robocup simulation league to be a testbed for their football AI research (Sean Luke 1997). However, Robocup simulation league rules are still not the same as real football rules.

For other types of games/AI-subjects, researchers have to consider the same factors as in this section.

5.4 The Limitation of using VBA in AI-TEM

To play a multiplayer mode in VBA (Link), two or more instances of emulators have to be used. Controlling many emulators at the same time while testing is not very convenient. It will be better if the second instance can run in the screen-off mode, in order to save CPU power.

Sometimes two connected emulators do not synchronize. This may damage the automatic module in long run. Detecting game state of both VBAs becomes necessary. We can then reload the game again if they do not synchronize.

Although there are some inconveniences, AI-TEM generally works well in our experiment. The emulator can be fixed to tackle the problem.

6 CONCLUSION AND FUTURE WORK

Our work provides an environment for testing AI on a wider range of commercial-quality games. Our experiment shows that, with appropriate game ROM, AI-TEM meets the three requirements in section 1 (test with a commercial-quality game, the game should not be too simple and there are many ways to play the game). Researchers can use AI-TEM to test their AI against the game's original AI or against a human opponent. Emulator players form huge communities, therefore many players can help with AI testing. AI developed by researchers can also be tested against AI running on script. A well designed script can help an evolutionary or learning AI improve in an appropriate direction. Although the GBA is not as powerful as next generation hardware, many games on GBA are regarded as classics and have been re-released on several new platforms. Therefore, AI-TEM is very much viable as testing environment for commercial-quality games. And the framework of AI-TEM should be adapted to more emulators in the future.

We plan to improve the implementation of the system and its associated tools. To provide a package for AI research in the future, we plan to collect the animation data of all characters in STZ3. We also want to build a cooperative AI for sport games using our testbed. The game WORLD SOCCER Winning Eleven is a perfect candidate ROM for the task. We also have plans to use another emulator with AI-TEM, such as MAME or ePSXe, in order to access more types of games. Multiplayer games can be run on MAME and ePSXe without synchronization or performance problems because only a single instance of emulator is needed.

REFERENCES

Aha, D.W., & Molineaux, M. 2004. Integrating learning in interactive gaming simulators. Challenges of Game AI: Proceedings of the AAAI'04 Workshop (Technical Report WS-04-04). San Jose, CA: AAAI Press

Bailey, C. and M. J. Katchabaw. 2005. An Experimental Testbed to Enable Auto-Dynamic Difficulty in Modern Video Games. Proceedings of the 2005 GameOn North America Conference. Montreal, Canada.

Graepel Thore, Ralf Herbrich, Julian Gold. 2004. Learning to fight. International Conference on Computer Games: Artificial Intelligence, Design and Education.

Kendall Graham, Kristian Spoerer. 2004. Scripting the Game of Lemmings with a Genetic Algorithm. Proceedings of the 2004 Congress on Evolutionary Computation, IEEE Press, Piscataway, NJ, pp.117-124

Ponsen Marc J.V., Hector Munoz-Avila, Pieter Spronck, and David W. Aha. 2005. Automatically Acquiring Domain Knowledge For Adaptive Game AI Using Evolutionary Learning. Proceedings The Twentieth National Conference on Artificial Intelligence.

Sean Luke, Charles Hohn, Jonathan Farris, Gary Jackson, James Hendler. 1997. Co-Evolving Soccer Softbot Team Coordination with Genetic Programming. First International Workshop on RoboCup, at the International Joint Conference on Artificial Intelligence.

Spronck Pieter, Ida Sprinkhuizen-Juyper, Eric Postma. 2004. Online Adaptation Of Game Opponent AI With Dynamic Scripting. International Journal of Intelligent Games and Simulation, Vol. 3, No. 1, University of Wolverhampton and EUROSIS, pp.45-53.

Python (2006). Python Language
http://www.python.org

VisualboyAdvance. (2005). GBA Emulator
http://vba.ngemu.com

VisualboyAdvance Link. (2005). GBA Emulator
http://vbalink.wz.cz/index.htm

ARTIFIST: ARTIFICIAL INTELLIGENCE FRAMEWORK FOR INTERACTIVE STORYTELLING

Minna Ruuska and Antti Virtanen
Department of Computer Science
Tampere University of Technology
Korkeakoulunkatu 10, 33720 Tampere,
Finland
E-mail: minna.ruuska@tut.fi, antti.virtanen@tut.fi

KEYWORDS
Interactive Storytelling, Artificial Intelligence, Computer Games, Multiagent Systems

ABSTRACT

In this paper we present an interactive storytelling framework suitable for games, where the story line is composed of actions made by multiple agents. The agents are given rules of conduct by constructing probabilistic decision trees and graphs. The agents choose their actions simultaneously using the trees, the graphs, and their individual knowledge bases. The probabilistic nature of the reasoning algorithm enables creation of life-like, believable agents that act logically most of the time, but may occasionally do something surprising. This should help maintaining the interest of players even after multiple sessions, since the story line changes over sessions.

INTRODUCTION

Recently, AI has become a major selling point of computer games. New games, such as Oblivion (Oblivion) and Mass Effect (Mass Effect), have been advertised for having non-player characters (NPC) that can act independently and do all the same things the player can. This approach creates games, where the story line is composed of the choices made by the characters rather than being fixed. Creating such AI is far from simple, especially, if one wants AI characters to be capable of more than plain shooting and fighting. Moreover, many issues in such AI generalize over a number of games creating a need for an AI middleware.

Several different approaches have been used for creating multiagent interactive storytelling environments. For instance, (Young and Riedl) have used hierarchical partial order causal link planning algorithm, (Cavazza et al) have used hierarchical task network planning, and (Mateas and Stern) have used the reactive-planning language ABL added with dramatic beats. However, the idea of planning as a primary reasoning system has a few problems. First of all, most of the time, we humans do not make decisions by carefully pondering the consequences of each action. In most cases we just do what we feel like doing or what we are used to doing. Another problem is efficiency. As the number of goals increases, finding the optimal action takes a lot of computing time. The human players also constantly change the game world forcing recalculation of plans. In most games the goal is to make the characters seem human rather than rational, and efficiency is a paramount concern. Therefore, planning is not necessarily the best computing model for game AI.

Our vision of game AI relates strongly to improvisation theatre. Imagine a group of actors who are to present an improvised play at a renaissance fair. They are given a topic for the play and a set of phrases and actions typical to the time period. The characters and relationships between them are settled, and the play may begin. From these premises even the same group of actors may create different plots and develop the play differently. The major goal is to create an entertaining and believable story. In our system, ARTIFIST (Artificial Intelligence Framework for Interactive Storytelling), players and NPCs are like actors, improvising based on the initial situation and rules of conduct defined by the game designer. Each choice of action for NPCs is based on the

agent's personal characteristics and its view of the world. Since the choices are not deterministic and the player can affect the story by choosing his actions, each playing session is unique. The vision behind Chris Crawford's Storytron (Storytron) engine is somewhat similar and some parts of it bear close resemblance to ARTIFIST. However, the ultimate goal of Storytron is to generate rich interactive stories rather than actual playable games.

SYSTEM DESCRIPTION

ARTIFIST consists of three main components: agents and background processes that are run in the scheduler of ARTIFIST, probabilistic decision trees, and an action graph (Figure 1). The game can have several types of agents (e.g. humans, dogs, cats, policemen, thieves) that can be inherited from each other. Agent types differ from each other in the structure of their knowledge base and decision trees they use. The agents of the same type differ in the contents of the knowledge base, possessions (money, armour etc.), and personal characteristics (i.e. honesty, aggressiveness). Each agent is handled by the algorithm which runs as a thread in the scheduler of the AI engine (Figure 2). The algorithm itself is the same for all agents, but the knowledge base and the decision trees are different.

Initiative Actions

Initiative actions are chosen using probabilistic decision trees. They consist of condition nodes and leaf nodes. Leaf nodes are either initiative actions or subtrees. Figure 3 presents a small example of a subtree. The same branch can appear in several different trees as well as several times in the same tree. Thus, a decision tree structure is not really a tree in mathematical sense but rather an acyclic directed graph.

Actions are obtained by calculating a path from the root node to a leaf node which is an action node. On each condition node, the branch is selected by evaluating child nodes and making a probabilistic choice amongst them. This probabilistic approach neatly creates agents that usually take the obvious action for a given situation, but may occasionally do something less apparent. Search for new initiative action is started

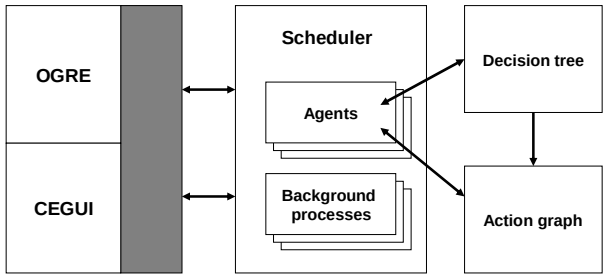


Figure 1 System overview

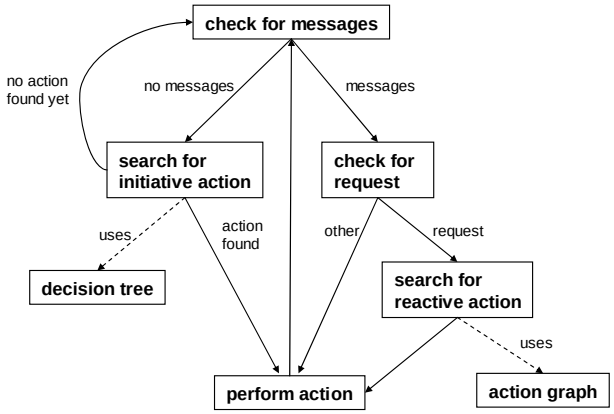


Figure 2 The reasoning algorithm of the agents

whenever agent is not engaged in an interaction or the conversation has paused. Not all agents, however, have to have decision trees: it is sometimes necessary to create an agent that is merely reactive and cannot initiate actions.

Reactive Actions

Reactive actions are managed by using a messaging system. Messages received by agents can be divided roughly into two categories: requests for communicational act during an interaction and perceptions from the environment. A perception could be, for instance, a bomb exploding or Jill and Tom kissing. The former would probably cause a visible reaction of the agent running to the opposite direction, while the latter would simply result in the agent deducing that Jill and Tom like each other, which might be of interest to the agent or not. These messages are arranged according to their priorities – emergency messages with high priorities are handled first and more subtle information later. If the message queue becomes too long, the least important

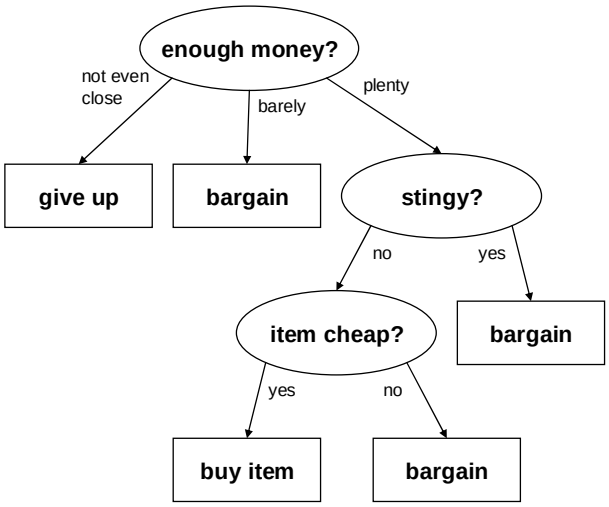


Figure 3 An example of a decision tree branch

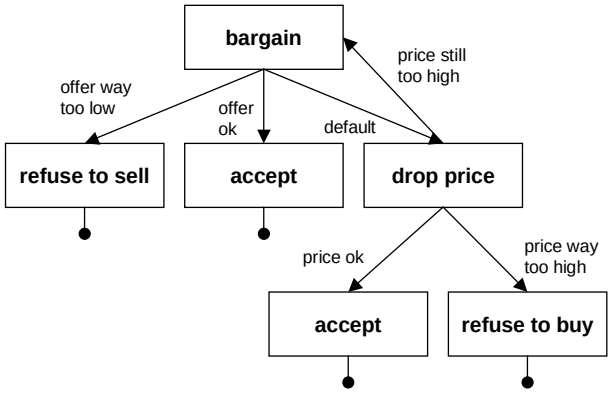


Figure 4 A fraction of an action graph

messages are discarded. The priority system prevents agents from being occupied with trivial issues, if surprising important things happen.

Discussions

Verbal interaction between agents is managed by special *interaction objects*. They contain information about the discussion at hand, and provide means for communication via a message transmission system. When an agent initiates an interaction with another agent, an interaction object is created and a request for answer is sent. The receiving agent will then choose the most appropriate follower action by using the action graph, which consists of action nodes as depicted in Figure 4. The neighbours of the previous action node are suitable answers to the last remark. The action graph contains the actions for all participants. Each node has associated code that executes the action (visible and audible reactions, effects on agent's knowledge base and on global objects) and a utility function that evaluates the utility of the action in current situation. The action is selected amongst the neighbouring nodes of the previous action probabilistically, using utilities as weights. For impossible actions the utility function returns zero. After performing an action, the agent sends a request to the other participants, if it expects to have some an answer. Thus, an initiative action starts an unpredictable sequence of actions which ultimately ends to an *end-action* or an *end-interaction* node that does not require an answer. Though the sequence is unpredictable, it is limited by the graph and utility functions, and therefore the interaction always makes sense.

Player Interaction

Any form of interaction requires the players to be able to make their own choices. Currently, a set of possible actions is gathered from a decision tree and offered to the player. In situations with a large number of possible choices, typically only the most promising ones are offered for selection. Thus the player may select only one of the actions a NPC in the same situation could select. This guides the player-NPC communication towards realistic and meaningful results still offering a lot of freedom for the player to test different approaches. Furthermore, we feel that parsing algorithms for natural language are not sophisticated enough for free text input to be used in most games. Nevertheless, if a sufficiently good natural language processing system was available, it could be integrated to our system in a manner fairly similar to that described in (Mateas and Stern 2004). Choices of reactive actions are gathered the same way for player agents and NPCs. If only one possible choice exists, it is executed automatically; otherwise all choices are offered to the player.

Background Processes

Sometimes it is necessary to model phenomena that change over time, for instance, manage the weather, increase capabilities of the player or his enemies towards the end of the game, or escalate dramatic elements to achieve better story arch. We

might even want to add in extra characters during the game. Needs like these are handled by *background processes* which run as threads in the scheduler of ARTIFIST. The background processes may affect the game world in any way the designer sees fit.

IMPLEMENTATION

We have designed ARTIFIST to be suitable for real computer games rather than being just a research project. To pursue this goal we have tried to make the system fast and as easy to use as possible. ARTIFIST is written in C++ and the whole design is object oriented and designed to be portable and extensible. At the moment, the AI scheduler does not create real threads, but instead runs them in a single thread solely responsible for all agents and background processes. The rendering is done by another system thread. Instead of using a full-scale game engine we have used free Ogre3D (Ogre3D) and CEGUI (CEGUI) libraries for graphics and user interface.

EXAMPLE APPLICATIONS

To demonstrate the capabilities of ARTIFIST we have created a few example applications. We have created simulations where the agents walk around and have short conversations without user interaction. Our current simulations are basically a proof-of-concept, but we have been planning various more challenging simulations with much more complicated agents. In addition to simulations we have created one playable game, The Murder Game.

The Murder Game

In the Murder Game the player has the role of a policeman trying to figure out which one of the suspects is guilty of a murder. The whole game takes place in an interrogation room, where the player can call in suspects for questioning. The suspects are shown as photographs with a few alternative facial expressions which vary according to the agent's state of mind. During the interrogation, the player can ask questions. The suspect then answers them either honestly or dishonestly depending on which answer he finds advantageous to himself, and how honest he is in general. An example of a short interview is listed in the Figure 5. In addition to ordinary questions, the player can ask the suspect to authorize a search for murder weapon in his accommodation. If the suspect agrees, the search is carried out, and the player is given information about the result. Finally, the player can arrest one of the suspects and thus end the game. The player is declared winner, if he was right about the murderer. Otherwise the real murderer is revealed, and the player has lost the game. For simplicity we decided to limit the NPCs to being purely reactive; they do not make initiative actions or communicate with each other autonomously.

PLAYER: Ask Al in
PLAYER: Do you think Tom could have done it ?
Al: That's not possible.
PLAYER: Why not?
Al: Well Tom liked Tim, so I think he is not the killer.
PLAYER: Do you think Ken could have done it?
Al: That's not possible.
PLAYER: Why not?
Al: Just a hunch.
PLAYER: Do you like Ken?
Al: Yes, I like Ken.
PLAYER: Does Ken have access to cottage 1?
Al: No.
PLAYER: That's all for now.

Figure 5 An interview from the Murder Game

EVALUATION

Although the examples presented here are relatively simple, it is easy to imagine that richer and more complex games could be created by combining their features. Changing the Murder Game agents to act autonomously and adding vivid 3D animation would create quite an interesting game. Suspects would exchange information, have conversations and arguments that could be overheard. Possibly even additional murders would be committed to hide or revenge the first one. On the other hand, an interrogation scene, such as implemented in the Murder Game, could spice up many more conventional games.

While implementing example applications, we have found ARTIFIST to be relatively easy to use, though its usability would benefit tremendously from a graphical tool for creating decision trees and action graphs. So far, graphs and trees have been drawn on a paper in the design phase and constructed by hand in the implementation phase using the pictures as a guideline. The pictures have

made communication about the applications easy, and we believe that even a person with no programming experience could contribute to AI design by drawing and discussing them. Thus, ARTIFIST enables better co-operation between game designers and programmers and respectively should result in better games. The transparency of decision trees and action graphs also makes locating and correcting errors easier. However, the fact that the story line is formed by actions of several independent agents is bound to cause unpredictable quirks. This was apparent even in our small example simulations. Similar surprising problems have been reported also by the Oblivion development team (Oblivion).

The probabilistic nature of the AI system is the core idea behind ARTIFIST and enables creation of life-like believable agents. The ARTIFIST NPCs take the logical course of action most of the time in a given situation but may occasionally do something else. This enriches the playing experience, since there is always something new to find in the game, even after many playing sessions. It also prevents creation of dominant strategies as the NPS's do not act deterministically.

FUTURE WORK

The next step in our work will be implementing a graphical user interface for constructing decision trees and action graphs. This should facilitate making more extensive example applications and, thus, help us explore the true capabilities of ARTIFIST. Another area we are going to look into is making ARTIFIST fit for massive multiplayer games as well as single player games. This requires modifications to the player interaction system and the internal technical solutions, as the computing has to be divided among multiple processors and computers. We have also designs for integrating alternate reasoning algorithms to cover special situations, where the decision trees and action graphs do not work well.

CONCLUSIONS

Creating complex interaction between characters without the help of an AI system is very difficult and most games offer quite restricted communication with NPCs. Creating interaction scenes with ARTIFIST is significantly easier than doing the same from scratch. The difference could be even greater, but unfortunately ARTIFIST currently lacks sophisticated developer tools. ARTIFIST is well suited for many types of constantly changing, dynamic game worlds, because the reasoning algorithm of the agents is relatively light and does not require continuous replanning. The probabilistic nature of our algorithm gives NPCs a life-like, human feel.

REFERENCES

Cavazza, M., Charles, F. and Mead, S. J. 2002. "Interacting with virtual characters in interactive storytelling", In *International Conference on Autonomous Agents*, Bologna, Italy, 318 - 325

Mateas, M. and Stern, A. 2004. "Natural Language Understanding in *Façade*: Surface Text Processing", In *Proceedings of Technologies for Interactive Digital Storytelling and Entertainment*, Darmstadt, Germany, 3 - 13.

Mateas, M. and Stern, A. 2005. "Structuring Content in the Façade Interactive Drama Architecture", In *Artificial Intelligence and Interactive Digital Entertainment*, Los Angeles, USA,

Young, R. M. and Riedl, M. 2003. "Towards an architecture for intelligent control of narrative in interactive virtual worlds", In *International Conference on Intelligent User Interfaces*, Miami, Florida, USA, 310 - 312.

CEGUI. http://www.cegui.org.uk/. 20.5.2006

Mass Effect. CGOnline magazine. http://www.cgonline.com/index.php?option=com_content&task=view&id=530&Itemid=1. 20.5.2006.

OGRE. http://www.ogre3d.org/. 20.5.2006

Oblivion. http://en.wikipedia.org/wiki/The_Elder_Scrolls_IV:_Oblivion. 30.5.2006.

Storytron. http://www.storytron.com. 7.7.2006

BIOGRAPHY

The authors are post-graduate students in Tampere University of Technology, Department of Computer Science, Institute of Software Systems.

Session 4

A Model of Facial Parameter Extraction and Animation

Mariofanna Milanova, Sireesha Sakamuri

Computer Science Department, University of Arkansas at Little Rock, Arkansas, USA
E-mail:mgmilanova@ualr.edu

Abstract

Recent advances in multimedia-related technologies and new applications such as virtual agents, video conferencing, visual effects in movies, and virtual players in computer games are motivating much research in digital character and face animation In this paper we present a system for the implementation of photo realistic avatar using video captured from the user. This is achieved by constructing a dynamic video map of facial expressions and mapping them to a 2D model. The dynamic video map reflects user's facial expressions with constant updates directly from the input video. The goal of this project is to provide a vivid representation of participants with the use of dynamic video map in perceptually important facial regions, notably eyes and mouth as compared to all of the facial parameters defined by MPEG4. A generic model is used to initialize the system and geometry updates can be done much lower than video frame rate.

1. Introduction

Although research has been done on facial modeling and animation for two decades it still remains a challenging task to create and animate a realistic faces. The complexity of the face makes facial modeling and animation a challenging task. The goal of facial animation is to create life-like synthetic agents that would closely mimic the facial movements of human beings while using low bandwidth.

A novel approach taken by our system is to consider few important facial regions, which would reasonably animate the model and there by reduce a lot of bandwidth. It is a hybrid model with video elements and model-based approach. We implemented an automated system that performs face detection, face tracking and facial feature extraction.

This paper is organized as follows. The second section summarizes the concept of MPEG4 facial animation. A description of various components of our proposed system is given in the third section followed by relevant implementation details and the result of several tests. The last section provides the future possible research and the conclusion.

2. MPEG4 facial animation specification

The main idea behind MPEG4 face animation is to transmit the commands causing changes in facial expressions, and to have the client side synthesize the facial image with the corresponding effect, thus eliminating the need to transfer images at all [2]. MPEG-4 specifies a face model in its neutral state, a number of feature points on this neutral face as reference points, and a set of Face Animation Points (FAP), each corresponding to a particular facial action deforming a face model in its neutral state. The FAP value for a particular FAP indicates the magnitude of the corresponding action, e.g., a big versus a small smile [4], [5], [6]. Deforming the face model in its neutral state according to the specified FAP values for the corresponding time instant generates a particular facial action sequence. Then the model is rendered onto the screen [7].

3. Our proposed system

Our System takes an input video sequence, some of the facial feature points defined in MPEG4 standard, a dynamic video map and a facial model as input and generate a realistic video avatar as output (Figure 1).

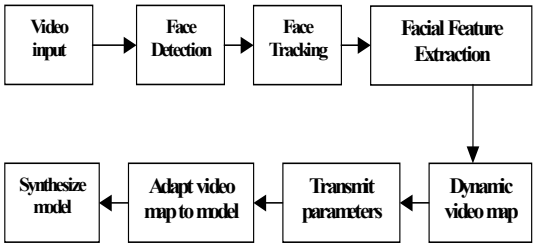


Figure1. Outline of our proposed system

3.1. Face detection

Our project uses a cascade of boosted classifiers and an extended set of Haar-like features to quickly detect profile views of faces in images. Haar-like features are intended to be a computationally efficient way to simulate early features of the human visual system. Individually each feature is a weak classifier, and they are combined to form a series of strong classifiers. Because later classifiers in the cascade rarely need to be evaluated, the cascade takes few computations on average.

This algorithm performs rapid object detection with a cascade of boosted classifiers on Haar-like features. It requires a large database of training images and a significant number of processor hours to construct a classifier so we used a pre-built classifier library.

3.2. Face tracking

Automatic tracking of human faces from image sequences is an important and challenging task in computer vision. We have implemented a real time face tracking system with Conditional density propagation algorithm, which is a robust technique for tracking objects through video sequences in the presence of clutter. The Condensation algorithm overcomes the pitfall of the Kalman filtering by allowing the probability density representation to be multi-modal, and therefore capable of simultaneously maintaining multiple hypotheses about the true state of the target. This allows recovery from brief moments in which the background features appear to be more target-like (and therefore a more probable hypothesis) than the features of the true object being tracked. The recovery takes place as subsequent time-steps in the image sequence provide reinforcement for the hypothesis of the true target state, while the hypothesis for the false target is not reinforced and therefore gradually diminishes [3].

3.3. Face feature extraction

This part deals with the facial feature extraction of the person in front of the camera. For this purpose we implemented a version of Motion Estimation Algorithm.

In order to find a new location for the control point

- Construct a footprint of the control point on frame i: Fi ;
- For each pixel j in search area on frame i+1, construct footprint (Fi+1, j);
- Find a pixel P, such that (Fi - Fi+1, P) is minimal
- Set the location of P to be the new location of the control point on frame i+1.

In this way we extract the facial features of the person. The set of new control points are the facial animation parameters. We considered important facial regions like eyes, nose and mouth, which produced reasonable results.

3.4. Constructing a dynamic video map

Once we calculate a set of new control points (the FAPs) we create a dynamic video map of these facial animation parameters and transmit them to the receiver. Video map is just a visual representation of the extracted facial animation parameters. As an example, a video map of the below face will comprise of only the control points.

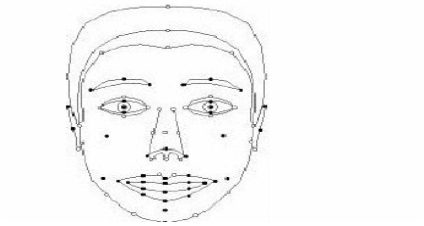


Figure2. A video map of the face

3.5. Adapt the parameters to the model

A single feature point must be mapped to many vertices along the real face model. Influenced points are a list of vertices in a vicinity of a feature point that must move along with this feature point. Individual weights may be assigned to each influenced point, usually proportional to the 1/ (distance from the reference point). This technique of mapping one feature point to multiple vertices reduces a lot of complex computations while producing reasonable results [1].

3.6. Experimental results and performance

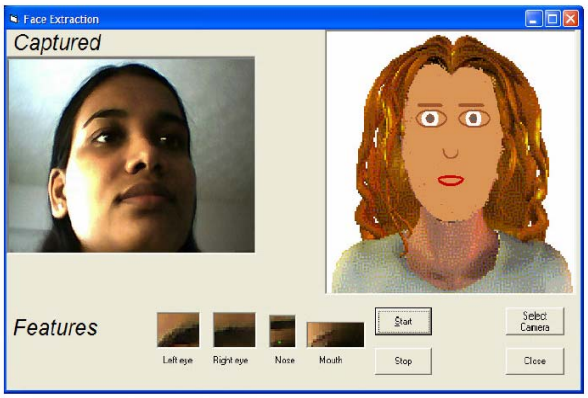
The proposed approach is implemented as a windows application with the capability of capturing the real time video sequence from Microsoft DirectX SDK. It then applies different techniques like face detection, face tracking and facial feature extraction to extract the facial features and then adapt it to the model. One novel approach is to consider few important facial regions, which can produce reasonable amount of animation with less computation cost. Creating a wide range of actions is one of the important criteria to be evaluated. Our proposed system is evaluated in this criteria and our technique produced reasonable amount of expressions..

Most of the new applications for face animation (such as video conferencing and interactive agents) need real-time operation. This means that the system has to receive continuous input (in some form) and produce output in real-time. A streaming real-time animation not only needs to be reasonably fast but also has to guaranty synchronization and work with input and output in standard formats. Our proposed model satisfied these criteria.

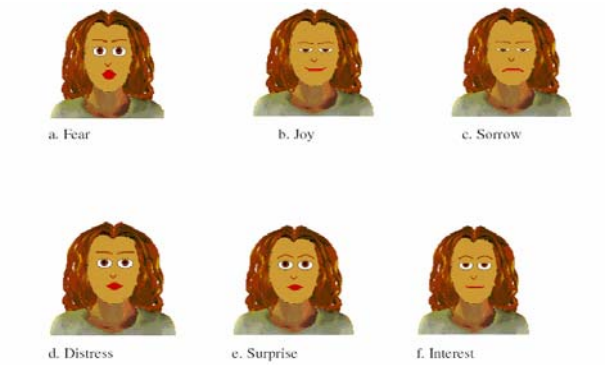
Existing multimedia technologies and standards can help improve the performance of face animation. In some cases compliance with these standards can be required, due to their popularity. Our system uses MPEG-4 standard, which is the first international standard.

Our system is computationally efficient and do not require any expensive equipment. Our tracking module can track the images even in the presence of background clutter. Another great advantage is mapping one feature point to a list of influenced points, which greatly reduced the computations.

Below is the screen shot of our video conferencing system.



Following are the different facial expressions obtained by our system



4. Future research and conclusion

Following is the future possible research in this project.

- Extend the model to 3D model;
- Increase the 8*8 Pixel area for better appearance;
- Add artificial intelligence and learning for the model;
- Consider the facial textures.

We have presented our approach in facial animation using MPEG4 parameters. With the exponential diffusion of information, education, entertainment and e-commerce application developers are seeking new types of interactive content with new goals to attract wider customer audience through appealing avatars. Our approach can be used to construct real time avatars and pedagogical agents.

5. References

[1]. Model-based Coding, Extraction, coding and evaluation of Face Model Parameters by Jorgen Ahlberg http://www.icg.isy.liu.se/~ahlberg/papers/

[2]. MPEG4 Facial Animation, Igor S. Pandzic and Robert Forchheimer, John Wiley & Sons Publishers.

[3] Condensation Algorithm - Home page. http://www.dai.ed.ac.uk/CVonline/LOCAL_COPIES/ISARD1/condensation.html

[4] F.I. Parke, K.Waters. Computer Facial animation, ISBN 1-56881-014-8, A.K. Peters, 1996.

[5] F.I Parke, A parametric models for facial animation", IEEE Computer Graphics and Applications, 2(9), 61-68 (1982).

[6] Real-time 3D Character Animation, Nik Lever, Focal press, 2003.

[7] Pedagogical Agents on the Web. Lewis Johnson, Erin Shaw, and Rajaram Ganeshan, Center for Advanced Research Technology for Education, USC / Information Sciences Institute, 2002.

Semantics Mining for Opponent Strategy Estimation

DAVID AL-DABASS, RICHARD CANT, CAROLINE LANGENSIEPEN
School of Computing and Informatics,
Nottingham Trent University
Nottingham NG11 8NS, UK.
E-mail: david.al-dabass@ntu.ac.uk.

Abstract: Two approaches to winning a game are considered. In the first, opponent's strategic intensions are reduced to a parameter vector which drives a multi agent evolutionary semantic net whose output is the tactics trajectory of game steps. Differential analytical models are used in a multi stage semantics mining process to estimate the opponent's strategic intensions from his game steps trajectory. A compound 6th order recurrent semantic architecture is used to translate the strategy parameters into tactics in a Kalman like process which emulates the action of 'mirror' neurons in biological intelligence. In the second approach, the strategic intensions are implicit in the solution of the game and Genetic Algorithms are considered for the Sudoku game. GAs are widely regarded as being relatively immune to the problems of local minima that affect many optimization schemes. However there are situations in which the population becomes too uniform in composition and the algorithm gets stuck. We discuss the use of a simulated disease mechanism to overcome this problem. The mechanism acts by reducing the fitness of individuals that are similar to the rest of the population, thereby giving a competitive advantage to those individuals that display unusual traits. This disease concept is tested using a simple genetic algorithm example.

KEYWORDS: Strategy estimation, Genetic algorithms, Evolutionary Computation, Simulated Disease, Competitive Learning, Neural Networks.

I. INTRODUCTION

Opponent's behaviour in games often exhibit random characteristics and cyclic behaviour attributed to a set of 'causal' parameters that represent strategy. Given the tactical behaviour time trajectories, recurrent hybrid nets are proposed to form a semantic architecture to determine the time derivatives of the trajectory and, using these derivates, to acquire the values of the strategy parameters in real time. The recurrent hybrid nets used posses de-noising properties which can be easily set. The short-term behaviour may be approximated by the dynamics of compounded second order systems [4]. Such systems are sufficiently detailed to represent the significant features of complicated game tactics but can be analyzed without excessive computation [Press et. al, 1992]. Three 'strategy' parameters govern the behaviour of a differential semantic net: natural frequency (ω), the damping ratio (ζ) and the external input (u), and two state variables. In a previous paper [6], 3 parameter estimation algorithms were tested for an equivalent 2nd order system ($\omega^{-2}.x''+2.\zeta.\omega^{-1}.x'+x=u$, $x(0)=x_0, x'(0)=x'_0$). The algorithms combine successive 1st order filters of specified cut-off frequencies to provide smoothing and higher order derivative estimation, with non-linear static parameter estimators. Three hybrid methods for parameter estimation were proposed and tested using the following 'architectures': 1) By using three sets of estimated 1st and 2nd 'tactics' time derivatives; 2) By using two sets of estimated 1st, 2nd and 3rd tactics time derivatives; and 3) By using a single set of estimated 1st, 2nd, 3rd and 4th tactics time derivatives of the measured opponent game moves. The sensitivity of these architectures to noise is investigated for a range of noise levels,- deliberate random moves to mislead the opponent.

A. Differential semantic models: an agent (game player) may show a temporal behaviour even when the input parameters to the strategy semantic model are constant, figure 1. The causal parameters themselves may be the output of other nodes, which may either be recurrent nodes or static nodes,- the latter may be logical or arithmetic. To model this oscillatory behaviour a second order integral hybrid model is proposed, figure 2. This model is based on the well known second order dynamical system which has the following form:

$$\omega^{-2} x'' + 2.\zeta.\omega^{-1}.x' + x = u \quad (1)$$

Where x is the trajectory of game steps (tactics) and ω, ζ and u are the natural frequency, damping ratio and input respectively of the strategy semantic model and represent the 3 causal parameters that form the input. To configure this differential model as a recurrent network, a twin integral elements are used to form a hybrid integral-recurrent net as shown in Figure 2-a.

B. Hybrid Recurrent Nets: The net shown in Figure 2 is a direct representation of equation-1 and is determined as follows:

i) The output (tactics) x of the strategy net is fed back to the first subtraction node on the left; as the input from the left of this node is u the output is (u – x).

ii) The middle input (to the whole net) from the left is ω; it is fed as 2 separate inputs to the multiplication node x to form ω^2 at its output, shown with an up arrow feeding as the lower input of the multiplier node above it, which is the second node from the left in the top chain of nodes.

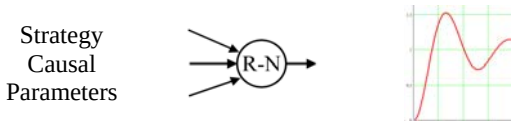


Figure 1. A Recurrent strategy semantic node (SSN) exhibits a temporal behaviour at the output despite having constant causal parameters.

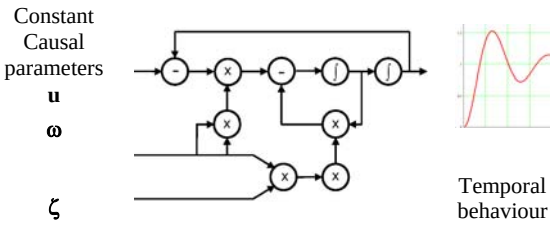


Figure 2-a. Hybrid integral recurrent net to model the temporal behaviour of strategy semantic node in Fig. 1

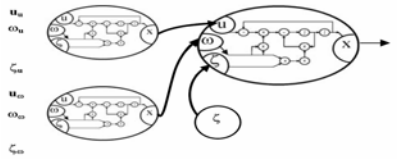


Figure 2-b. Two of the strategy parameters of semantic layer i have 2nd order dynamics.

iii) The output of this multiplier node is therefore $\omega^2.(u - x)$, i.e. the RHS of equ. 1-A.
iv) The bottom input from the left (to the whole net) is ζ which is fed as the lower input to the first of the two multipliers in the bottom chain of 2 nodes,- as the top input to this node is ω the output is ζ.ω. which is multiplied by 2 in the 2nd node in the chain to produce 2. ζ.ω.
v) The last node on the right in the top long node chain is an integrator node that generates x as stated in ii) above. As it is an integrator node, the input to it must therefore be the derivative of x, i.e. x'. This is multiplied by the output of the right node in the bottom 2-node chain (which is 2. ζ.ω) to produce 2. ζ.ω. x' , which is the 2nd term in the LHS of equ. 1-A or the 2nd term on the RHS of equ. 2-1-B.
vi) By subtracting this output from the output of the middle node in the top row, we get the full RHS of equ. 1-B, i.e. $\omega^2.(u - x)$ - 2. ζ.ω.x'.
vii) As the output of the second integrator from the right (in the top chain) is the first derivative of x, x', the input to this integrator node must be x'', i.e. the LHS of equ. 1-B.
viii) Connect the output of the middle node of the top chain (which is $\omega^2.(u - x)$ - 2. ζ.ω.x') into the input of the 2nd integrator from the right (x'') to complete the equation.

II. SIMULATION STRUCTURE FOR STRATEGY ACQUISITION

A. Inference Networks: The strategy semantics embedded within the game player is continually changing and need dynamic models to represent and acquire their parameters from observed data. In a normal inference network the cause and effect relationship is static and the effect can be relatively easily determined through a deduction process by considering all the causes through a step-by-step procedure which works through all the levels of the network to arrive at the final effect. A reverse process is needed here where the observed moves are attributed to a given strategy semantic model with unknown parameters and the task of the semantic acquisition process is to determine the values of these parameters.

B. Strategy Acquisition Dynamics: Work in this paper extends these ideas to recurrent models where some or all the strategy parameters are time varying. The effect is now a time dependent game-move pattern, which forms the input to a differential estimation process to determine the strategy semantics in terms of time varying causal parameters. These causal parameters will themselves embody knowledge (meta semantics) which may be obtained through a second level estimation process to yield 2nd level causal parameters. These processes consist of a differential part to estimate the higher time derivative of the game moves, followed by a non-linear algebraic part to compute the causal parameters.

$$\begin{bmatrix} N_{i+1} \\ x1_{i+1} \\ x2_{i+1} \\ Ex_{i+1} \\ Exd_{i+1} \\ Exdd_{i+1} \\ Extd_{i+1} \\ Exqd_{i+1} \end{bmatrix} := \begin{bmatrix} N_i + 1000\left[\left(rnd(n) - \frac{n}{2}\right) - N_i\right]\cdot d \\ x1_i + x2_i\cdot d \\ x2_i + \left(u - 2\cdot\zeta\cdot\omega^{-1}\cdot x2_i - x1_i\right)\cdot\omega^2\cdot d \\ Ex_i + G\left(x1_i + N_i - Ex_i\right)\cdot d \\ Exd_i + G1\left[G\left(x1_i + N_i - Ex_i\right) - Exd_i\right]\cdot d \\ Exdd_i + G2\left[G1\left[G\left(x1_i + N_i - Ex_i\right) - Exd_i\right] - Exdd_i\right]\cdot d \\ Extd_i + G3\left[G2\left[G1\left[G\left(x1_i + N_i - Ex_i\right) - Exd_i\right] - Exdd_i\right] - Extd_i\right]\cdot d \\ Exqd_i + G4\left[G3\left[G2\left[G1\left[G\left(x1_i + N_i - Ex_i\right) - Exd_i\right] - Exdd_i\right] - Extd_i\right] - Exq \end{bmatrix}$$

Figure.3: Iterative Euler integrator for the noise source, behaviour generator and higher time derivative estimator.

To prepare for behaviour simulation with noise, the Runge-Kutta integration routine had to be abandoned as it uses 4 evaluations of the derivative vector which results in different noise values being used in each evaluation. A simple iterative Euler integration with a single evaluation was used, see Figure3.

The top term represents the random number generator feeding an integrator whose output N forms the noise source. The 2nd and 3rd terms represent the state space terms of the 2nd order system generating the observed trajectory x1 and its derivative x2. The 4th term shows a 1st order filter whose input is the noise corrupted trajectory x1+N with a gain G and integration step d. The 5th to 8th terms show the successive first order filters that generate the time derivatives of x, i.e. x', x'', x''' and x''''.

III. STRATEGY PARAMETER ACQUISITION EXAMPLES

A. Derivative Estimation: Deliberate misleading random moves added to the trajectory undergoes

successive filtering as it passes through the stages of the higher time derivatives. Low values of G correspond to low cut-off frequencies in the filter, which result in smoother derivative estimates. Figures 4 below show 2 such cases for G=20 and 30 respectively, the heavier filtering effect of G=20 on the derivative trajectories is quite noticeable. The noise level is 10% of the nominal value of x at 1.0.

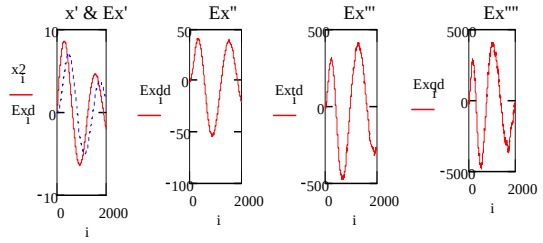


Figure 4-a: Higher derivative estimation: n=0.1, G=20.

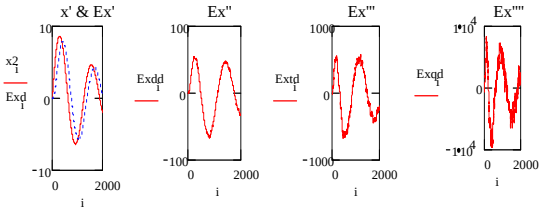


Figure 4-b: Higher derivative estimation: n=0.1, G=30

B. Strategy Parameter Acquisition: The highest time derivative used in this architecture is x" and thus unaffected by errors in estimating x''' and x''''. The gain was set to 20, and a low noise level of 0.1% was injected into the measured trajectory. The 3 estimated parameter trajectories are shown in Figure 5-a. After transient period, all 3 parameters converged to give good estimation accuracy with ω=8.9, ζ= 0.097 and u=1.004. The lag index shift, L, clearly had a direct effect on the accuracy of estimation where it was found that best results were obtained when L=95.

Although these 2 architectures use higher derivatives x''' and x'''', they performed equally well using the heavy smoothing of G=20 and suitable lag compensation as shown Figure 5-b and 5-c.

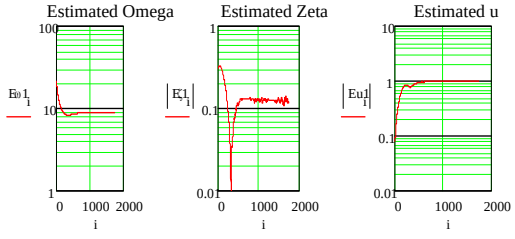


Figure 5-a: Parameter estimation of ω, ζ and u using Architecture 1, n=0.001, G=20, L=95.

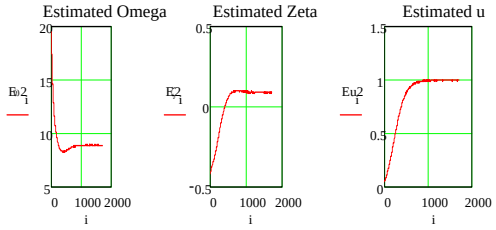


Figure 5-b: Estimated ω, ζ, u using Architecture-2, n=0.001, G=20, L=95.

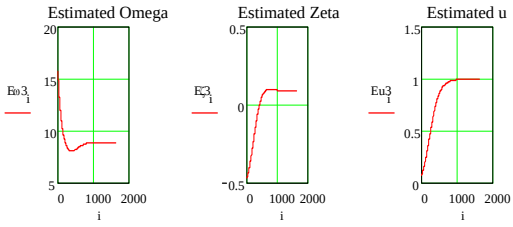


Figure 5-c: Estimated ω, ζ, u using Architecture-3, n=0.001, G=20, L=95.

C. Sensitivity to Misleading Moves: The noise level was increased to 1% and 10% and the performance of all 3 architectures was demonstrated as shown in Figures 6-a, 6-b, 6-c, 10, 11 and Figures 7-a, 7-b and 7-c respectively.

1% Noise level:

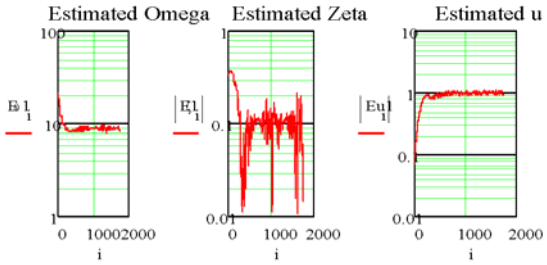


Figure 6-a: Architecture-1: Estimated ω, ζ, u using, n=0.01, G=20, L=95

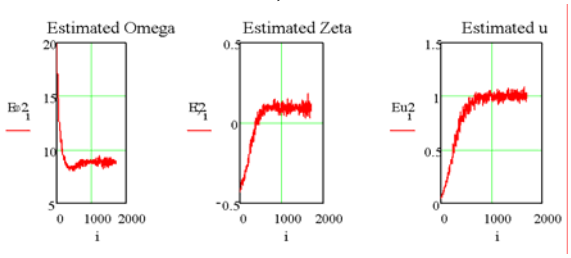


Figure 6-b: Architecture-2: Estimated ω, ζ, u using n=0.01, G=20, L=95.

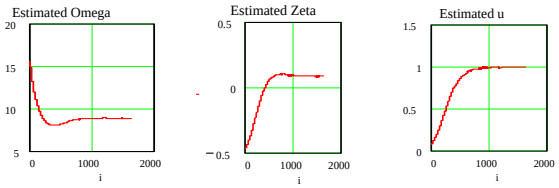


Figure 6-c: Architecture-3: Parameter estimation of ω, ζ and u using n=0.01, G=20 and L=95

It is clear that the erratic noise in the parameter trajectories reduces from Architecture-1 to 2 and from 2 to 3.

10% Noise level:

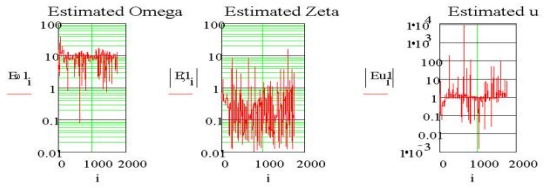


Figure 7-a: Architecture-1: Estimated ω, ζ, u using Algorithm-1, n=0.1, G=20, L=95.

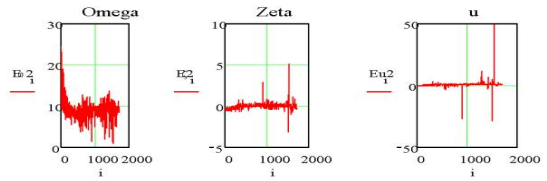


Figure 7-b: Architecture-2: Estimated ω, ζ, u using Algorithm-2, n=0.1, G=20, L=95.

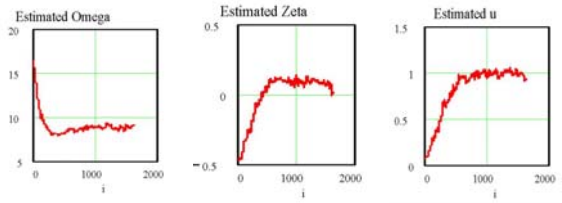


Figure 7-c: Architecture-3: Parameter estimation of ω, ζ and u using Architecture 3, n=0.1, G=20 and L=95

IV. DISCUSSION

A semantic architecture was put forward and its effectiveness tested to derive strategy parameters embedded within game step trajectories. Variations of the architecture were tried to test the robustness of estimating strategy parameters for increasing noise levels to reflect deliberate random moves to mislead opponents. Heavy smoothing was provided by low cut-off frequencies, which caused the derivative estimators to have increasing lags in the successive stages. A numerical lag compensation technique was introduced which selected progressively distant values of the higher derivatives. This, together with the increased smoothing applied to the higher derivatives, gave architecture variant-3 the leading edge in strategy parameters acquisition accuracy. Further work will investigate deeper levels of semantic agents to provide more accurate tracking of strategy parameters.

V. SIMULATED DISEASE TO IMPROVE GENETIC ALGORITHMS

Genetic algorithms are a popular research topic and have been applied in many different fields and a variety of different forms [1,2]. One of the attractive features of this approach is that it seems to be relatively immune from the problems with local minima that afflict many optimisation methods. However this does not necessarily imply that the technique is immune from such difficulties as failure is rarely reported in the literature and hence there may have been many unsuccessful attempts to use the method that are known only to their authors!

The mechanism that might lead to such a failure is if the entire population becomes "stuck" in the region of the local minimum. If the search space is large and the fitness function does not offer any incentive to move away from the minimum then this situation may persist indefinitely. A biological analogy for this would be the evolution of an ability such as flight where there is little or no competitive advantage to be had until the ability reaches a fairly advanced stage. Indeed wings could be a serious impediment to the survival of a creature if they aren't actually good enough to work!
In such a situation the continued diversity of the population is the key to progress and the fitness function does not provide any mechanism to provide this.

In this paper we will investigate the idea of using a simulated disease mechanism as a means of increasing the diversity of the population. The hope is that the disease will eliminate genetic patterns that have persisted for a long time – allowing new patterns to emerge that may provide a better solution to the problem.

VI. TRIAL PROBLEM - SUDOKU

The origins of the idea lie in a toy genetic algorithm model that was being developed initially as a teaching example. In recent years the Japanese Sudoku puzzle has become popular worldwide. These puzzles are constructed on a 9x9 grid and the idea is to fill in a numbers in each cell such that every row and column and 3x3 sub-grid contain all the digits from 1 to 9. Figure 8 shows an example starting position whilst Figure 9 shows the solution. Extensions to larger sized grids and more dimensions exist but the 9x9 size is most popular.

				7		2		
		5	8					
2				4	1	5		
3					4	1	8	9
	1						6	
5	8	7	1					3
		6	5	9				7
					7	9		
		1		3				

Figure 8: Sudoku puzzle example

Now it must be said at the outset that genetic algorithms are not a particularly good way of solving these puzzles. It is straightforward to write a program that follows the same kind of strategies as human solvers and such programs are totally effective and very fast. However it was felt that this was a good domain in which to demonstrate the brute force effectiveness of genetic algorithms, which can solve the problem without resort to any form of logical deduction provided a suitable fitness function can be defined.

For the initial development the fitness function was defined to be the total number of violations of the puzzle rules. A good fitness value is thus a low number, zero representing a correct solution.

6	4	8	9	7	5	2	3	1
1	7	5	8	2	3	6	9	4
2	3	7	5	4	1	5	7	8
3	6	2	7	5	4	1	8	9
9	1	4	3	8	2	7	6	5
5	8	7	1	6	9	4	2	3
4	2	6	5	9	8	3	1	7
8	5	3	2	1	7	9	4	6
7	9	1	4	3	6	8	5	2

Figure 9: Sudoku Example - Solution

The initial population was defined by randomly filling in the empty squares. At each generation the population was sorted by fitness and duplicate patterns were removed and replaced by new random combinations. The bottom half of the population was then removed and replaced by new individuals that were generated by a procedure of parenting followed by possible mutation. To generate a new individual two different existing individuals were selected at random from the top half of the population. The new individual was then generated by randomly selecting each place on the grid from the two parents. A further random selection allowed occasional mutation of the result.

When this algorithm was tested it converged and solved the first problem that was tried moderately rapidly. However further tests showed that, although initial convergence was quite consistent, the algorithm often became stuck with around about four errors remaining in the best solution found. Four errors may seem a small number but the sudoku search space is very large and so a solution with four errors could easily have thirty (out of eighty-one) locations that are different from the correct solution.

At this stage a modification to the way that the problem is mapped onto the GA is usually effective and so it proved in this case. When the system was modified to force compliance on one constraint (the rows constraint) for all solutions from the point of initial pseudo-random generation onwards then consistent and comparatively rapid convergence was obtained. However it is interesting to see whether the original method could be made to work since there may be situations in which the size of the search space cannot be reduced in this way.

VII. THE DISEASE MECHANISM

The motivation behind the disease idea is to keep the evolutionary process going by eliminating genetic sequences that have existed in the population for a long time – even if they satisfy the fitness criterion. The mechanism that we use for identifying such sequences is a competitive learning neural network of the kind described by Kohonen, [3] This neural network runs in parallel with the genetic algorithm. For each generation of the evolutionary process one training cycle of the network is run.

Because the Sudoku system is discrete, the standard competitive learning techniques have to be modified slightly. The procedure works as follows. Initially a number of network nodes are set up at random. The format of these nodes is an array of 81 integers in the range 1-9 just like the solutions themselves. During a training cycle each member of the population is presented to the network in turn and the winning node (that is the node that most closely resembles the input) is identified. Next this node is modified to make it more like the input. Usually, in a competitive learning network, this is done by adjusting all the weights (which are floating point numbers) by a small amount. Here however the weights are integers and intermediate values have no meaning because the order of the numbers forms no part of the puzzle definition. Consequently we have adopted a process in which a small number of weights (typically 2) are selected at random and then made to match the input exactly. Optionally we include a neighbourhood mechanism whereby a set of nodes that are near to the winning node are also adjusted to resemble the input. In this case however fewer cells are changed (typically 1).

When the GA is initialised, the network cannot converge to anything in particular and so remains random. As the GA starts to converge so particular patterns start to persist in the population, allowing the network to converge on them in turn. At this stage the process of removing over-persistent patterns can begin. When the winning node for each member of the population is computed, the degree of resemblance is known. This number is used in turn to weight a random process of “killing”. Thus the longer an individual exists unchanged within the population the more the network will contain nodes that are trained to resemble it and the higher its probability of being killed by the disease. This process will be all the more rapid if there

are many other members of the population that are similar.

VIII. INITIAL RESULTS

Initial results showed the expected behaviour. The "death rate" started very small and then increased as the GA started to converge. This can be observed in Figure 10. Early tests also showed that the algorithm converged on a number of combinations of problem and random number seed where previously if had not. However one must be very cautious in assessing the results of random processes until a statistically significant amount of data has been gathered. It soon became apparent that this was going to be a difficult task.

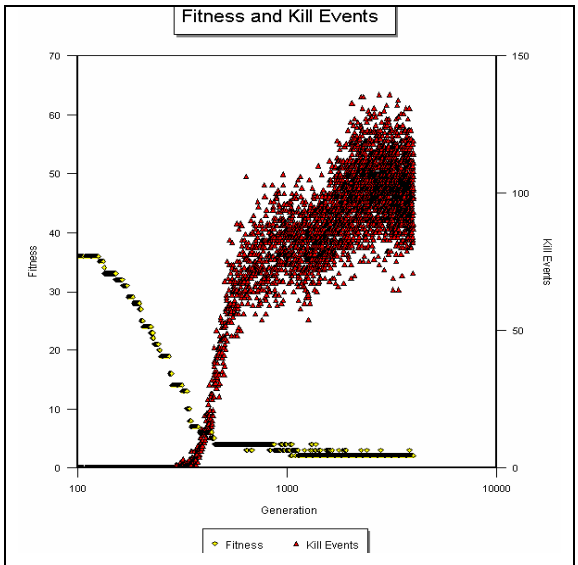


Figure 10: Fitness and Kill Events.

The problem is that the algorithm takes anything from a few minutes to many hours to converge, assuming that it is going to converge at all. The original intention was to gather data for different parameter values and so to optimise things like the kill probability, neighbourhood size and so-on. However to do this many runs would have been required for each parameter value and this was clearly never going to be practical. The alternative approach was to look at secondary factors, such as the rate at which new "good" solutions (with low numbers of rule violations) were being generated. This seemed promising at first, but, as the number of results accumulated, what had looked like a developing trend dissolved into randomness.

Another way of looking at the population is to measure the closeness of its members to the actual solution rather than just the number of rule violations. We call this number the secret fitness since in a real application one would have no mechanism for knowing it. The secret fitness gives a useful early warning of whether the solution is actually going to converge. It was observed that, on the occasions when convergence never happened, this number started high and remained so. When the algorithm did converge the secret fitness had a much lower value from the start.

IX. VARIATIONS ON THE ALGORITHM

In an attempt to obtain better results a few variations on the algorithm were tried. The most promising of these seems to be the introduction of a "seasonal" factor.

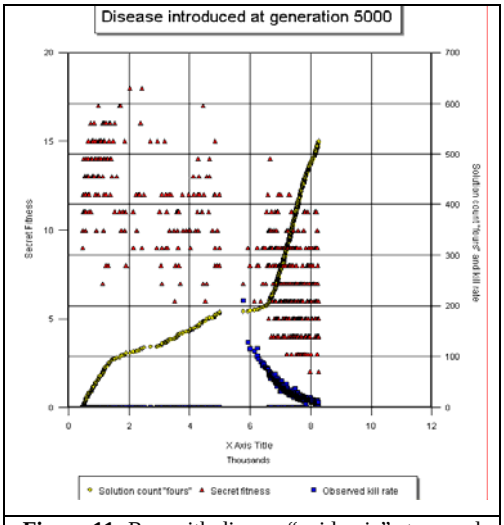


Figure 11: Run with disease "epidemic" at around generation 5000

When the kill rate is very high the normal progress of the GA is interrupted but a low kill rate does not really have much effect on the content of the population. It was conjectured therefore that it might be effective to have short periods of high kill rate, "epidemics" to re-randomise the population, interspersed with longer periods of lower kill rate to allow the GA to converge.

X. FURTHER RESULTS

This "seasonal" mechanism did provide the best evidence so far of the algorithm working. In Figure 11 we see the plot of new population members of fitness 4 (the best value that is common enough to provide a reasonable statistical measure) against generation number. The other curves are the secret fitness of the new solutions and the rate at which solutions were observed to be killed. Up to generation 5000 the kill probability was set to zero and no kills were observed. After generation 5000 a high kill probability was set with a progressive decline in subsequent generations. This pattern is directly reflected by the observed kill rate. We can observe that the rate of generation of fitness 4 solutions (as indicated by the slope of the curve) remains moderate up to generation 5000, pauses whilst the kill rate is high and then shoots up more rapidly thereafter. The change is also shown by the secret fitness. After generation 5000 there seems to be a change in population, leading to the GA converging to a solution at about generation 9000.

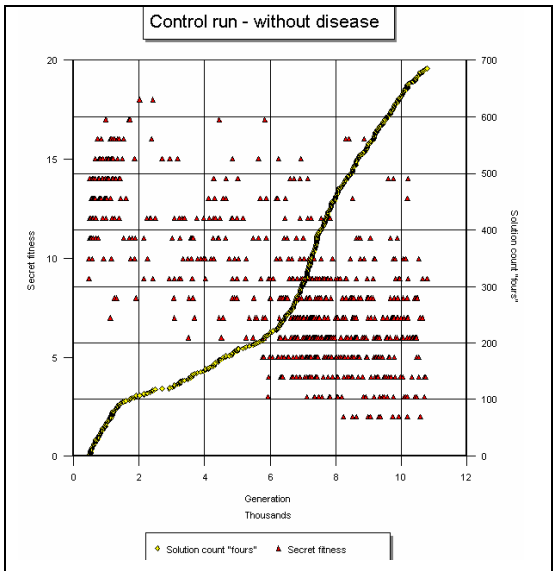


Figure 12: control run as figure 4 but without disease

However caution is required here since a "control" run with the same random number seed produced the result shown in Figure 12. Figure 12 is qualitatively similar to Figure 4 but we note that the changes are not so sharp and the process took 12000 generations to complete instead of 9000.

XI. CONCLUSIONS

Two approaches were explored to the problem of estimating opponent's strategic intensions, one using semantic nets to estimate the strategy parameters and the second using the concept of disease as a way of maintaining the rate of progress of a genetic algorithm. While the first approach was successful in theoretical simulation trails, the second was less so, where it was not possible to give a convincing demonstration that the mechanism works owing to the time that would be needed to gather statistically significant data; however, there is some evidence that the mechanisms could perform as intended given further work.

REFERENCES

1. Baeck, T. & Schwefel, H.-P. (1993) "An Overview of Evolutionary Algorithms for Parameter Optimization", Evolutionary Computation, 1(1), 1-23.
2. J. Heitkoetter and D. Beasley, eds., 2001. "The Hitch-Hiker's Guide to Evolutionary Computation: A list of Frequently Asked Questions (FAQ)", USENET: comp.ai.genetic. Available via anonymous FTP from rtfm.mit.edu/pub/usenet/news.answers/ai-faq/genetic/ About 110 pages. Also available on the Internet at http://www.faqs.org/faqs/ai-faq/genetic/part1/
3. Kohonen, T. "The Self-Organising Map" Proceedings of the IEEE; 78(9): 1464–1480.
4. D. Al-Dabass, D. J. Evans, S. Sivayoganathan, "Signal Parameter Tracking Algorithms using Hybrid Recurrent Networks", I. J. of Computer Mathematics, Vol. 80, No. 10, October 2003, pp 1313 - 1322 , ISSN 0020-7160 print.
5. W. Press, W. Vetterling, B. Flannery, S. Teukolsky, "Numerical Recipes in C: The Art of Scientific Computing:2nd ed.", *Cambridge University Press*, 1992.
6. Al-Dabass, D, Evans D., and Sivayoganathan, K., "Derivative Abduction using a Recurrent Network Architecture for Parameter Tracking Algorithms", *IEEE 2002 Joint Int. Conference on Neural networks, World Congress on Computational Intelligence*, pp1570-1575, Hawaii, May 12-17, 2002.
7. D. Al-Dabass, A. Zreiba, D. J. Evans, S. Sivayoganathan, "Parameter Estimation Algorithms for Hierarchical Distributed Systems", I. J. of Computer Mathematics, Vol. 79, No. 1, January 2002, pp65-88, ISSN 0020-7160.
8. J. D'Azzo and H. Houpis, "Linear Control Systems Analysis and Design", 4th ed., *MacGraw Hill series in electrical and computer engineering. Control theory*, 1995.
9. W. Press, W. Vetterling, B. Flannery, S. Teukolsky, "Numerical Recipes in C: The Art of Scientific Computing:2nd ed.", *Cambridge University Press*, 1992.
10. Richard Cant, Julian Churchill, David Al-Dabass, "Using Hard And Soft Artificial Intelligence Algorithms To Simulate Human Go Playing Techniques", IJSSST, Vol.2, No.1, pp31-40, June 2001, ISSN:1473-804x Online, ISSN:1473-8031 Print.
11. P. Eykhoff, "System Identification Parameter and State Estimation", *John Wiley & sons*, 1974.
12. J. Beck and K. Arnold, "Parameter Estimation in Engineering and Science", *John Wiley and sons*, 1977.

VOICE INTERACTION SYSTEM FOR VIDEO GAMES USED WITHIN "VIRTUAL SINGER" COMPUTER INTERFACE

Jocelyne Kiss and Karim Abdeljelil
Department of Arts and Technologies
University Marne la Vallée
E-mail: kiss@univ-mlv.fr

KEYWORDS

Tools and systems for video games and virtual reality devices. Interfaces and controllers. Timbre perception and interactivity. Singing simulation. Serious games.

ABSTRACT

The aim of this paper is to develop voice interaction modules increasing the immersive potential of video game contexts, implemented via an interactive virtual singer device. A number of inquiries are followed around the use of pertinent features resulting from human-voice spectrum analysis—including timbre. Connexionist filters provide real-time discrimination on several levels of information from sound-recorded digital data to morphing of the virtual singer device, matching coherent and synchronous voice articulation. This study is extended to cover the use of timbre characterizations intended for implementation on multiplayer platforms, thus referring to the issues of separation of sound sources within context.

INTRODUCTION

The field of video games using standard modes of interaction (keyboards, joysticks or mice) is currently expanding beyond its traditional bases. It now seems only natural to consider this trend as likely to develop along a number of fundamental lines such as – for instance – the increase in the amount of interactivity available from already existing platforms through a more advanced exploitation of their potentials. In particular, the current switch-like nature of start operations might be superseded in order to allow for more adequate immersion in correspondence with greater diversity in the quality of external parameters. In this way, the world of sound, human gestures and new strategies for the integration of non-linear drama might acquire a significant or even prominent role in the design process of such immersive systems. This creative component carries genuine economic impact and is likely to result in a complete rethink of representation and playability modes.

Using sound as a mode of interaction leads to the issue of segmentation and – subsidiarily – to the problem of extracting units of meaning. Similar to what is required for the recreation of human behavior (walking, swimming…etc.) within virtual reality software (Bret, 2000), the creation of a song automaton will require the building of shapes developing in time while related to anteriority. This orientation imposes consideration of morphological and phonetic categories for musical phrases connected to the contents of a virtual scene. Implementation of a voice-based mode of interaction necessarily relies on a specific point of view determining treatment and therefore on a number of partially arbitrary decisions—depending on the data extraction system selected. This much presupposes the operation of a particular representation system for sounds and vocal gestures (Kappas and all. 1991).

One of the main objectives in this paper is to implement analysis techniques designed to capture information specific to the voice of each player, based essentially on the study of spectrum and allowing for the development of interactive strategies within context. Another objective is to focus on the rendering of intent detectable in a player's voice, so as to integrate it into the virtual scene while respecting all multimodal components. This bears on the issue of computerized modeling of human intent and it seems necessary to bring in a global solution for the relevant extraction of intentional data from sound flow - coming from one or several players - in order to create a system of coherence between what is "played" and what is "perceived".

To achieve this, we will assume the point of view of an interaction simulating an adaptive, time-based dimension of improvisations being sung in real time (Pecchinenda, and all., 1997). Significantly, we will rely on completion theories for the modeling of responses from the virtual singer device (for facial emotion modelization we used old Facs reference, (Ekma and Friesen 1977)). A first part will be devoted to the description of the implemented device while expounding its conceptual rationales. A second part will be focused on capturing meaning from sound recording, together with relevant suggestions for methods of exploiting timbre. Finally, we will explore choices open for accompaniments and completions of phrases initiated by players within the framework of designing educational games for the teaching of singing (Scherer, and Kappas, 1988).

COMPUTER INTERFACE DESCRIPTION

It has long been demonstrated that the quality of immersion into any interface heavily depends on the player's modes of involvement (Angel, 2000) regarding potentials being offered to him/her for progress in accordance with his/her own propositions. Our current device is built in this perspective and the use of the player's own voice as the mode of interaction will be the vector used both to trigger actions and to sustain the development of further possibilities from the sound elements provided by the player himself/herself. Its object will therefore be to support the player in the way of

harmony by allowing the avatar to sing with him/her on the lower third or – depending on each case – on the upper sixth. Our device creates a sort of chorus and completes the player's musical phrase while respecting his/ her own initial propositions.

Our essential idea was to design a sort of intelligent "karaoke", initially intended for very young children—including those with psychomotor deficiencies whose handicaps limit their abilities to use mice or styluses. The interactive virtual singer would then allow for playful and entertaining learning of basic nursery rhymes (Scherer and Kappas, 1988)

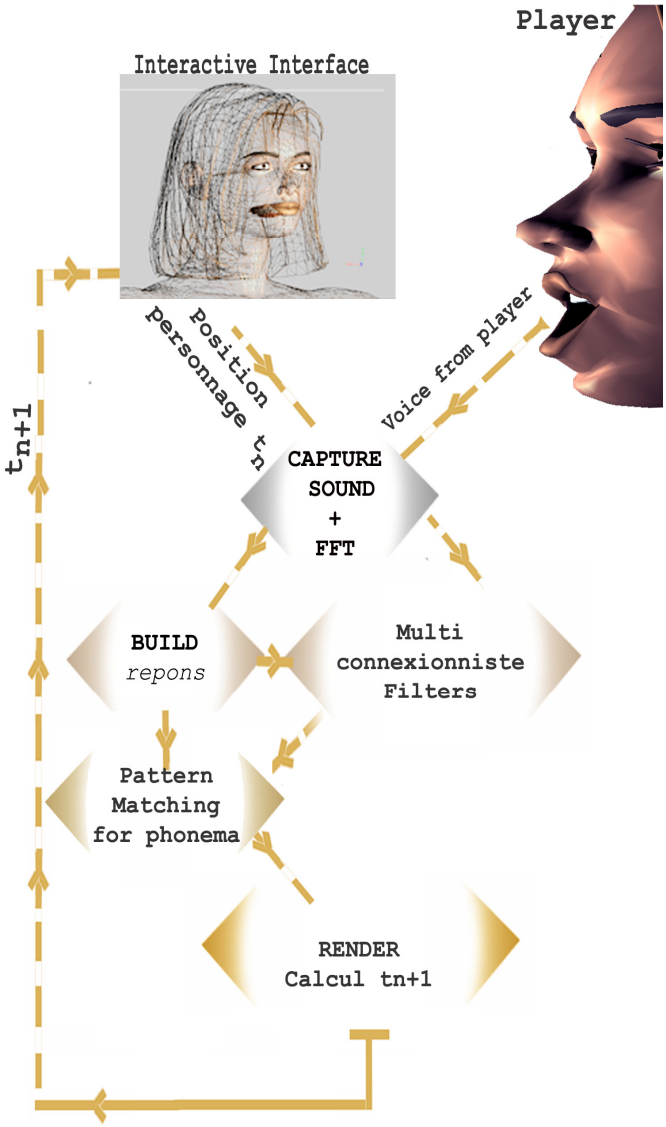


Figure 1. General interactivity treatment diagram

The device is built around a virtual singer model - created using 3DSMAX© for meshing, rendering and animation keys, including problems of facial articulation in relation to phonemes intended for pronunciation. Such data are exploited in real time and animated with Virtools© , (middleware system for virtual reality), -interacting with the voice of any given "player". His/her voice is captured by a microphone and analyzed in accordance with the standards of Fourier's theory (including FFT) before generation of data to be processed through connexionist filters (we used archirecture filters proposed by Kohonen T., and all. 1992) allowing for the recognition of pitch and characterization of timbre at the outcome of learning and calibration processes (further description below). Neural networks used for this phase are mainly built on the back-propagation model. Using the 256 frequencies resulting from FFT and captured at the network's entry point via a hidden layer, real-time discrimination is available over two scales.

A stochastic module based on the theory of conditional probability then animates the singer which follows the player, suggests possible *responses* to be sung (i.e. completions of the player's own phrases) and simultaneously triggers facial movements in correspondence with on-going vocal expression (Scotto Di Carlo and Guaïtella, 1995). This architecture reflects the desire to create an interactive dimension based on the exploitation of a number of significant, built-in components of vocal gesture (phonemes, melodies…etc.). The diagram presented as Figure 1 evidences the in-system management of successive phases used to treat the available interactivity. Articulation on behalf of the virtual singer relies on a pattern-matching method using repertories of pre-recorded sounds and animations used for synchronous playing (De Bonis and Nahas, 1999). Figure 2 evidences the significance of all components of the human voice-tract being simulated for a most realistic kind of articulate singing.

VOCAL INTERACTIVITY

The study of vocal interactivity is based on the knowledge of our modes of reception/perception of sound phenomena. The notion of timbre does appear to play a significant part in the recognition of any sound, together with that of pitch (Houstma, 1995). Timbre can be defined as set of qualitative properties of sound (Schaeffer 1966), such as its time envelope, vibrati, brilliance, spectral envelope…etc. (Plomb, 1970). Their list is not exhaustive—which gives this definition empirical/consensual aspects. For a number of scientists, the perceptive recognition of timbre must be based on 'family groups' (Handel, 1995, and Roads, 1998) sharing close or identical characteristics. This falls in line with previous taxonomic studies intended to discover invariant elements used to achieve speech recognition (Blumstein and Stevens 1979).

For other researchers, such grouping using type categories ought to be supplemented within the framework of proper modeling by the possibility to dissociate timbre from the pitch at which any sound is played, since it would seem that short-term human memory acts in this way (Semal and Demany, 1991). Others still think such classifications modeling timbre are necessarily limited in general value—owing to the considerable qualitative differences existing between the voices of two individuals (Shepard, 1982). Pattern-matching techniques for timbre recognition have nonetheless demonstrated some reliability, however much lengthy calculation time is required (Sethares, 1993). Our own proposition is based on the treatment and recognition of a singing voice intended for video games. Hence our preference for a simplification of the overall problem, i.e.

taking into account only a certain number of parameters and not all listed qualities (D. Herrera-Boyer, and all. 2003) in order to qualify sound while answering the challenge of real-time constraints placed on the interface.

One the essential principles of musical analysis is reference to pitch and duration of sounds. In this perspective, spectrum analysis does seem to provide an indispensable tool but it involves complex calculations unfit for real-time constraints placed on video games. However, this may be avoided using very specific algorithms of the "Fast Fourier Transformation" (FFT) type. FFT supposes clustering of sound samples to the power of 2—generally 256, 512, 1024…etc. This allows for fast calculation of the spectral formula:

$$c_n = \frac{1}{2^\mu} \sum_m f(\frac{m}{2^\mu}) \exp(-\frac{2\pi nm}{2^\mu}) \text{ , } n \leq 2^{\mu-1}.$$

The choice of this segmentation results in often undesirable "side effects", thus making real-time recognition of pitch more complicated. Let us for example record a "perfect A" sung at 440Hz by a player and assume our segmentation mode at every 256 samples with a sampling ratio of 44,110 samples per second. One cycle will take about a hundred samples and segmentation will therefore result in a "truncated" wave of 2 cycles and a half. Fourier treatment will then provide a spectral representation for the first sine amplitudes equal to 0.303, 0.707, -0.579, -0.163,-0.085, … etc. Sound now appears as a mere resultant of sinusoids, unconnected to any perceptible musical concept. Furthermore, this mode of analysis does not allow for characterization of pitch from the examination of fundamental frequency and therefore complicates the extraction of timbre. "Timbre" is meant here as the set of secondary harmonics being superadded to the pitch of any produced sound. In fact, this resulting complexity is the direct consequence of the inadequate nature of discretionary segmentation and clustering being used.

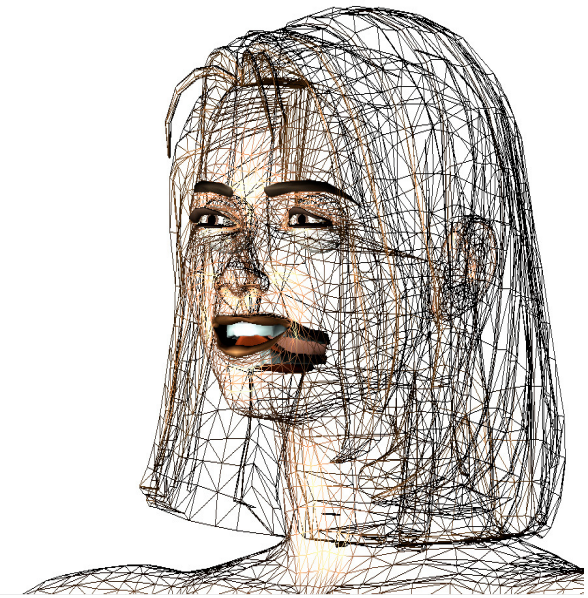


Figure 2. Wireframe presentation of the interactive virtual singer - visualization of articulation-

A proposition to by-pass these undesirable effects is easily formulated via timbre analysis of sound in accordance with a frequency system based on the fundamental note. In this fashion, a "perfect A" analyzed at the frequency of 400Hz will now result in the series 1, 0, 0, 0…etc. (Fourier coefficients in sine)—which naturally constitutes the expected result in terms of characteristic sinusoids. This approach supposes prior knowledge of pitch and real-time FFT adaptation, in order to allow for clean differentiation of the fundamental note from other timbre components. As regards the determination of instant pitch, two natural methods may be used: one based on standard FFT treatment followed by spectrum analysis using differentiation processes based on calculations; the other – more adequate to real-time usage – based on a network of artificial neurones (of the back propagation type) operating from spectral bands to determine pitches regardless of timbre.

In fact, this task may be subdivided into several recognitions carried out by several Artificial Intelligence units specialized for correct interpretation of pitches differentiated from a wide scale. In simple terms, the automaton's learning process will rely on the detection of pitch from standard periods of about 6ms each, bearing in mind that note duration may easily be deducted for more precise extraction of music data. At this stage, any sung "A" would be recognized in real time by the network but its timbre - and therefore its vocal (or instrumental) imprint - would not be identified. The corresponding wave function f on the elementary unit of time is now transformed by scale change into that of $x \mapsto f(\frac{256}{100} x)$ via an interpolation process designed to adjust the first hundred samples to the FFT algorithm at the rate of 256.

Spectrum analysis then evidences the fundamental note to be an "A" and allows for the appearance of secondary components readjusted by reverse scale transformation in order to retrieve real frequencies. Generally speaking, this treatment will adjust sample clusters in accordance with sound pitches recognized for FFT treatment at a fixed rate by means of double scale-change via interpolation. This method allows for simple, real-time reading in terms of secondary components' spectral bands being added to enrich the instant pitch of sounds generated by the user. This "remainder" of secondary fringes will be defined as "timber residue".

“Timbre residue” may therefore be considered from the outset as an instant vocal imprint specific to the pitch of recorded sound. Insofar as timber residue remains of an unknown nature – subject in principle to great variations corresponding to each type of human voice -, we thought it more appropriate to implement a dynamic network of the Kohonen type (i.e. without learning supervision) to achieve effective timbre recognition. In simplified terms, this involves four entry cells capturing four different amplitudes tested against a range of frequencies equally allotted in correspondence with the 256 FFT analyses. The network then performs real-time discrimination in accordance with 64 possible timbres. This interface is only an experimental

example and the underlying models are of course likely to undergo wide-ranging diversifications. Moreover, the instant vocal imprint may allow – under certain hypotheses and constraints – for the separation of sources intended for distinct activities on a multiplayer platform. Let us imagine, for instance, two players in vocal interaction - respectively intended for two virtual characters – using the same microphone. Transmitted signal f will therefore represent the total of two signals f_1 and f_2 coming from the two players' voices. After segmentation and real-time spectrum analysis, the Fourier coefficients on f, $c_n(f) = c_n(f_1) + c_n(f_2)$, will be determined and the problem of separating sources would consist – in this approach – in retrieving the series $(c_n(f_1))$,$(c_n(f_2))$in order to achieve separation by synthesis of approximate values for signals f_1 and f_2. Such is the desired condition for selective interactivity.

In theory, without further indications regarding f_1 and f_2, differentiation is impossible since there is infinite deconstruction for a signal resulting from the total of two signal inputs. However, if sources carry their own characteristics – as is the case with of distinct voice data – the timbre characteristic may be used as an element of answer. Indeed, let us imagine a scale (or more generally speaking a set of sounds) whose notes are subjected to spectrum analysis when sung at constant intensity by one voice 1 and one voice 2, and resulting via averaging in calibrations expressed as Fourier coefficients. Reference series obtained will be $(c_n(f_1,h))$, $(c_n(f_2,h))$with h representing a given pitch. In an instant real situation, intensities on voices 1 and 2 are coefficiented by factors

λ_1 and λ_2 andfrom the reference situation and perceived signal f is characterized after spectrum analysis by the following series of Fourier coefficients:
$(**)\ c_n(f) = \lambda_1(\delta t)\, c_n(f_1, h_1(\delta t)) + \lambda_2(\delta t)\, c_n(f_2, h_2(\delta t))$ with δt representing the time interval where FFT is applied and h_1 and h_2 representing the respective instant pitches assigned to it. The resulting equation system (**) must now be solved. In digital terms, it must noted that this system generally proves incompatible owing to the series $(c_n(f_1,h_1))$, $(c_n(f_2,h_2))$ providing average – non-exact – values.

However, minimizing the mean deviation of quantities (if extant—which is generally the case):

$$\frac{\begin{vmatrix} c_i(f) & c_i(f_2,h_2) \\ c_j(f) & c_j(f_2,h_2) \end{vmatrix}}{\begin{vmatrix} c_i(f_1,h_1) & c_i(f_2,h_2) \\ c_j(f_1,h_1) & c_j(f_2,h_2) \end{vmatrix}}, \quad \frac{\begin{vmatrix} c_j(f) & c_j(f_1,h_1) \\ c_i(f) & c_i(f_1,h_1) \end{vmatrix}}{\begin{vmatrix} c_i(f_1,h_1) & c_i(f_2,h_2) \\ c_j(f_1,h_1) & c_j(f_2,h_2) \end{vmatrix}},$$

will provide intensity values constant enough to allow for partial reconstruction of signals f_1 and f_2, at time $t + \delta t$. Such reconstruction will be termed an approximate or "pseudo-solution".

The main obstacle remaining is then to determine a real-time pseudo-solution and a corresponding performing algorithm. A possible reduction would consist in limiting calculation to what is sufficient by performing random tests on three sets of values i, j, k—in order to eliminate in succession all pitches creating considerable variations. This method affords adequate results but only for limited sets of listed sounds or notes.

AVATAR REACTIONS

It seemed essential for us, in order to create expectation and captivate our young players' attention (Bentley, 2002), to sustain their interest in the avatar's reactions. The introduction of specific tag-endings for musical phrases – i.e. *responses* – thus creates a renewal of dialogue between the player and interface. (And create a developpment, an evolution, like Thompson recommend it, 2002). Tags are introduced as follows: the user sings a sequence of sound events, each of them being recorded and analyzed - just as previously described – for real-time generation of corresponding harmonization; intervals and durations between the various sound events are then retained and a fitting construction to end the phrase is suggested. This method is actually based – however crudely – on the antecedent/consequent model used by classical composers. For instance, a player singing the "C,D,E" sequence will be accompanied by the virtual singer device singing "A,B,C" and ending the musical sentence on its own by suggesting, let us say, "E,D,C". Naturally, improvisation variations are to be adjusted at will and made as much complex as you want them.

This model evidences organization based on a polarization effect. According to music analysts, its duality in both system and symmetry entails reliance on a principle of musical coherence at perceptive level (Sadaï 1980). We therefore considered the part sung by the player to be the "antecedent" for which we had to provide a consequent. In concrete terms, we implemented a module intended to determine the overall key and the main melodic "motives" on the sung segment. The corresponding consequent is then generated in accordance with the model's polarity rules. This amounts to converting some of the motives initially proposed by the player in accordance with the harmonic context set for the ending before concluding the phrase with a perfect cadence—just as recommended by the consequent model.

PERSPECTIVES

Results taken from a young audience have proved conclusive. The current device does allow them to sing lullabies rhymes and improvise musical phrases in an entertaining fashion, playing their own game or in the company of other children. It affords them an opportunity to learn to the notion of time value*, though some aspects of treatment remain somewhat rudimentary—especially those related to phrase-endings

* The time notion as such is often barely apprehended by children with psychomotor deficiencies. Such deficiencies frequently render interfaces based on more traditional modes of interaction too difficult to handle.

proving to be more efficient within the framework of a lullaby than with improvisation without reference. Further research our on behalf will be pursued to provide a more realistic rendering of articulation and various phoneme models are already being tested.

REFERENCES

- Angel, E. 2000. *Interactive Computer Graphics*. Addison-Wesley.
- Bentley P.J. 2002. *Digital Biology. The Creation inside Computers and How it will Affect us*. Headline Review.
- Blumstein, S.E. & Stevens, K.N. 1979, "Acoustic invariance in speech production: Evidence from measurements of the spectral characteristics of stop consonants". *Journal of the Acoustical Society of America*, 66:1001-1017.
- Bret M. 2000, « Virtual Living Beings ». *Virtual Worlds*. Heudin J-C.(Ed.), 119-134
- Nahas, M. and De Bonis, M. 2001, *"Image Technology and facial Expression of Emotions"* P. Coiffet & A. Kheddar (Eds) *Proceedings of the 10th International Workshop on Robot and Human Interactive Communication*, pp. 524-527
- Ekman, P., And Friesen, W.V. 1977. *Manual for the Facial Action Coding System*, Palo Alto. Consulting Psychologists Press.
- Handel, S. 1995, "Timbre perception and auditory object identification". In *Hearing*. B.C. J. Moore, ed., New York: Academic Press: 425-461.
- Houtsma, A.J.M. 1995. "Pitch perception". In *Hearing* B.C.1. Moore (ed.), New York: Academic Press, 267-295
- Kappas, A., Hess, & Scherer, K.R. 1991, "Voice and Emotion". in*, Fundamentals of nonverbal Behavior* R. Feldman & B. Rimé (Eds.), New York : Cambridge University Press : 200-238.
- Kohonen T., Laine P., Tiits, Torkkola K., 1992 "A Nonheuristic Automatic Composing method", in, *Music and Connectionnism*. Massachusetts Institute of Technology. 229-242.
- Pecchinenda, A., Kappas, A., & Smith, C.A. 1997 "Effects of difficulty and ability in a dual-task video game paradigm on attention, physiological responses, performance, and emotion-related appraisal". *Thirty-Seventh Annual Meeting of the Society for Psychophysiological Research*, Cape Cod, Massachusetts. 34. S70.
- D. Herrera-Boyer, G. Peeters, and S. Dubnov, 2003."Automatic classification of musical instrument sounds," *J. New Music Res.*, vol. 32, no. 1: 3–21.
- Plomp, R. 1970, "Timbre as a multidimensional attribute of complex tones". In *Frequency Analysis and Periodicity Detection in Hearing*, R. Plomp & G.1 Smoorenburg, eds. Leiden: Sijthoff: 397-414
- Roads, C. 1996. The Computer Music Tutorial. USA.MIT Press.
- Sadaï, Y. 1980. *Harmony in its Systemic and Phenomenological Aspects*, Jerusalem, Yanetz. And Y. Sadaï, 1992, « D'une logique systémique et phénoménologique de la musique ». *Analyse Musicale.* No. 28: 22-28.
- Schaeffer P. 1966, *Traité des objets musicaux*. Seuil.
- Scherer, K.R., & Kappas, A, 1988. "Primate vocal expression of affective state". Dans D.Todt, P.Goedeking, & D. Symmes (Eds.), *Primate vocal communication*. Berlin: Springer-Verlag. 171 – 194.
- Scotto Di Carlo, N., et Guaïtella, I., 1995, "Facial Expressions in Singing : A pilot study", *Communication au XIII° Congrès International des Sciences Phonétiques*. Stockholm. 226-229.
- Semal C. and Demany L. 1991, "Dissociation of pitch from timbre in auditory short-term memory". *J. Acoust. Soc. Am.*, 89 : 2404–2410.
- Sethares, W.A. 1993. "Local consonance and the relationships between timbre and scale*". Journal of the Acoustical Society of America,* 94, 1218-1228.
- Shepard, R.N. 1982, "Structured representations of musical pitch". In D. Deutsch (ed.), *The Psychology of Music.* New York: Academic Press, 343-390.
- Thompson A. 2002, « Notes on design through artificial evolution: Opportunities and algorithms », in *Adaptive computing in design and manufacture.* Parmee I. C.(Ed.) Springer-Verlag. 17-26.

BIOGRAPHY

Currently in charge of the Multimedia & Digital Art syllabus at the Department of Arts and Technologies of the University of Marne la Vallée, Jocelyne Kiss focuses research on the current debates defining relationships between the various types of sight, sound and other somatic perceptions, as well as their exploitation within immersive-interactive interfaces—for video games in particular. Ms Kiss is the author of numerous papers, interactive installations and plug-ins based on such themes (including « The perception of vocality through a virtual 3D interface within the framework of an interactive opera » in *FIRT-IFTR international conference: The Director in the theatre world*, « Animats and interactive design » in ISIMD 2006, *4th International Symposium of Interactive Media Design*, with Bailly K.; «Computer analyses of the baby movements: Study of the precursory elements of language. » in *Proceedings of the Fifth International conference on Human and Computer HC-2004*; and - with Chu-Yin Chen - « Setting up of a self-organized multi-agent system for the creation of sound and visual virtual environments within the framework of a collective interactivity » in *Proceedings of ICMC 03*)—as well as of reference books on *Modeling of compositional processes related to cognitive sciences* (published in 2004) and on *Singing animats within immersive-interactive interfaces* (to published in 2007). She is also the author of three educational video games with variable ergonomics intended for disabled children.

Session 5

The Game Genre Factor in Computer Games Based Learning

Mats Wiklund
Department of Computer and Systems Sciences
Stockholm University
Forum 100
164 40 Kista
Sweden
Phone: +46 8 161614
E-mail: matsw@dsv.su.se

KEYWORDS
Computer games, learning, education, game genres, empirical study.

ABSTRACT

As the usage of commercial, off-the-shelf computer games as teaching tools are being discussed and empirically studied, the varying properties of different game genres is an important factor that should be taken into account. The possible impact on study results that may be inherent from game genres as such is an issue that needs to be studied in order to assess the potential of using commercial games in a learning situation.

To obtain more information on the impact of game genre on a learning environment, an interview study was conducted. Students undertaking their 10:th and 11:th year of study as part of a test project using commercial off-the-shelf computer games of their own choosing as the main teaching tool, were interviewed about their favourite game genres. This was correlated with their study results in the subject of English (as a second language), for students favouring FPS (First Person Shooter) games versus MMORPG:s (Massively Multiplayer On-line Role-Playing Games).

Results show that students with MMORPG:s as their favourite game genre (with or without other genres in conjunction) received a higher average number of yearly grades in English (as a second language) than students with FPS games as their favourite game genre.

BACKGROUND

As computer games occur in many different shapes and genres, they possess vastly different properties with respect to learning. These differences are manifested as different visual styles, as well as different game tempos and regarding the type and intensity of the player interaction. Independently from this, also the typical feedback mechanisms differs among various game genres. This includes what types of player actions are rewarded with benefitious objects and/or valuable skills, leading to advancement in the game. In effect, these feedback mechanisms are controlling what the player has to strive for in order to succeed in the game. Such differences among game genres results in different genres having different potential as tools for learning.

Among the first observed learning effects regarding computer games are those related to reflexes and hand-eye co-ordination. As remarked by Griffiths, these findings are also accompanied by those pointing out particular aspects of games as having important bearing on using them as educational resources: *"Research dating back to the early 1980s has consistently shown that playing computer games (irrespective of genre) produces reductions in reaction times, improved hand-eye co-ordination an raises players self esteem. What's more, curiosity, fun and the nature of the challenge also appear to add to a games educational potential"* (Griffiths 2002).

The idea to use games as learning tools emerged long before the existence of computer games, however, with *"The modern era of simulation gaming"* (Wolfe and Crookall 1998) including large simulation games such as the RAND corporations logistics simulator for the US Air Force, and the first business simulation being used in college education as early as 1957 (Dickinson and Faria 1997). These and other developments made Duke suggest in 1974 that games may become an entirely new form of communication in education, as noted by Woods: *"He suggested that simulation games might offer a possible answer to the problems of education in an increasingly complex society"* (Woods 2004), in reference to: *"…gaming is a future's language, a new form of communication emerging suddenly and with great impact across many lands and in many problem situations"* (Duke, quoted from Woods 2004).

Further research in the area of specific advantages of computer games as educational tools has pointed out several aspects where games fit very well into key patterns of successful learning. As Gee points out, these aspects need not be related to those features that are perhaps most often noted regarding computer games, such as the graphics: *"The secret of a videogame as a teaching machine isn't its immersive 3-D graphics, but its underlying architecture. Each level dances around the outer limits of the players abilities, seeking at every point to be hard enough to be just doable"* (Gee 2003a). This positive aspect of something being hard, and the danger of making things too easy, is also discussed by Papert: *"What is best about the best games is that they draw kids into some very hard learning … The fact is that kids prefer things that are hard, as long as they are also interesting"* (Papert 1998).

This touches on the Practice Principle, outlined by Gee as one of several principles involved in successful learning situations: *"Learners gets lots and lots of practice in a context where the practice is not boring (i.e. in a virtual world that is compelling to learners on their own terms and where the learners experience ongoing success)"* (Gee 2003b). Among other notable such principles are the Achievement Principle: *"For learners of all levels of skill there are intrinsic rewards from the beginning, customized to each learners level, effort, and growing mastery and signalling the learners ongoing achievements"*, the Ongoing Learning Principle (abbreviated):

"The distinction between learner and master is vague, since learners … must, at higher and higher levels, undo their routinized mastery to adapt to new or changed conditions …", and the Probing Principle: *"Learning is a cycle of probing the world (doing something); reflecting in and on this action and, on the basis, forming a hypothesis; reprobing the world to test this hypothesis; and then accepting or rethinking the hypothesis"* (Gee 2003b).

In the light of these principles, it becomes clear that computer games fit in very well as an educational tool, especially if one also takes into account that many games span across subject boundaries, being able to offer learning in several areas at once. As pointed out in a study by Kirriemuir and McFarlane regarding the roller coaster simulator game RollerCoaster Tycoon: *"The game could be used across a number of subject domains, such as physics (motion and velocity), and business and economics (running a theme park)"* (Kirriemuir and McFarlane 2003).

The usage of unmodified, commercial, off-the-shelf games is not the only possibility, though. The development of a combination of educational software and computer games, often referred to as "edutainment" has been the result of efforts trying to explore the game format and fill it with more traditional, school curriculum oriented material. However, the usefulness of such edutainment software has been questioned in many cases, as observed by Kirriemuir: *"However, when game-oriented entertainment and learning or educational material are combined, the result has often been disappointing; the educational value is debatable or irrelevant, and the gaming and engagement qualities compare poorly to those of pure games"* (Kirriemuir 2002).

A similar standpoint is taken by Papert, viewing this edutainment "offspring" from games and education software as one possessing none of the best features from either "parent": *"Shavian reversals – offspring that keep the bad features of each parent and lose the good ones – are visible in most software products that claim to come from a mating of education and entertainment"* (Papert 1998). More specifically, Kirriemuir and McFarlane points out several reasons for these shortcomings: *"Most edutainment has failed to realise expectations, either because: • the games have been to simplistic in comparison to competing video games … • the tasks are poorly designed and do not support progressive understanding … • the target audience becomes aware that it is being coerced into 'learning', in possibly a patronising manner"* (Kirriemuir and McFarlane 2004). These known issues regarding edutainment makes it interesting to investigate if unmodified, commercial off-the-shelf games may be more useful as educational tools.

Another issue central to using computer games as educational tools, is the role of the teacher. Here, Kirriemuir notes various misassumptions about teaching using computer games in the classroom, such as: *"The teacher will be marginalised, and become partially or fully redundant, by the game. The role of the teacher is reduced to an assistant who turns the computers on and off"* and *"The pupils work individually, boothed, one to a game, in monastic silence. Learning is an isolated and unsocial experience…"* (Kirriemuir 2005). As Kirriemuir points out, these are misassumptions, and if realised such learning environments would be very unfortunate indeed. Instead, if treated by the teacher as a beneficial resource, computer games may take the role of tools that may enhance his/her teaching, the key point being that the games are tools in the hands of the teacher. Being able to use activities occurring within computer games as starting points for educational activities extending out from the games, is one example of how the teacher in a highly creative and active way may create fruitful learning situations. A key point from a study conducted by the British Educational Communications and Technology Agency, BECTA, is that a strong teacher focus is essential: *"The role of the teacher in structuring and framing the activity of the learner remains crucial if learning outcomes are to be achieved."* (BECTA 2001).

Game Genre classification and learning potential

In the absence of a formally defined genre classification scheme, a spontaneously developed de-facto model is commonly used to classify computer games. A number of generally accepted type genres are thus often used to describe a games genre, including the common case that a game may be considered to be a combination of several of the type genres.

Examples of commonly referred to type genres on a high level of abstraction are *action games, strategy games,* and *role-playing games.* Examples on a more detailed level of abstraction are *FPS (First Person Shooter) games, construction games, and MMORPG (Massively Multiplayer On-line Role-Playing Games).* The boundaries between various genres are not always clearly defined, but although the genre boundaries are sometimes fuzzy, these type genres are commonly accepted and serve reasonably well as a reference model for the classification of computer games, at least if mixtures of genres are also taken into consideration.

When considered as potential tools for game based learning, the various game genres of commercial computer games offer different possibilities, in some cases present as a natural consequence of the game concept as such. With game genres that possess a high natural potential as a learning tool, a noticeable learning effect may occur also in those commercially sold games that are not developed for the purpose of being a learning tool. An example of this is several construction/strategy games, from which the player can gain insights in areas such as city planning, elevator algorithms, and the operation of amusement parks, by training on these tasks as they are being simulated in the games.

Another genre with interesting properties from a teaching/learning perspective is MMORPG:s, Massively Multiplayer On-line Role-Playing Games. These games, also referred to as *online games,* are commonly capable of handling many thousands of simultaneous players interacting with each other and the game environment in very large game worlds with complex in-game economies and social structures. Gameplay is constantly ongoing, around the clock, carried out by the players currently connected to the game servers, *persistent worlds.* The possibility to chat online with other players during gameplay is being used frequently.

Online games adds an interesting dimension, increasing the potential for online games as learning tools. As a result of the games online nature, player behaviour and progress can be monitored by game controllers with access to the game servers. If needed, game controllers could also adjust the players situation, either by centrally manipulate settings in the game servers, or by taking physical form in the game world and communicate with the players directly, for instance in a mentor-like manner. Alternatively, others than the those

controlling the game servers might use the game in this way, for instance a teacher appearing in the game through his or her own game account, having in-game meetings with students in a mentor-like fashion.

The studied test project

In Botkyrka, Sweden, a test project using computer games as the primary teaching tool for a class of students in upper secondary education was initiated in the fall of 2003. The project first included students in their 10:th year of education, and now in the second year of the project includes students in their 10:th and 11:th year of education in a mixed fashion. This represents the first and second year of the non-compulsory education in the Swedish school system, normally corresponding to students reaching the age of 16 and 17 if continuing directly from the compulsory school system.

The pedagogical issue of using unmodified off-the-shelf commercial computer games as the main teaching tool was of great interest. The students were free, up to the limitations of the project budget, to suggest game titles to be used. Although the teachers has the right to refuse any suggested game they feel would be too extreme, this veto right had never been used up to the time of the study. The resulting mix of game titles thus reflects the preferences of the students themselves:

Game titles used	Number of regular players
World of warcraft	20
Counter strike	18
Battlefield 1942	15
Age of empires	11
Age of mythology	11
Star wars galaxies	11
Warcraft III	10
Diablo	9
Rise of nations	9
Morrowind	7
Tibia	7
Sims	5
Neverwinter nights	5
Sim City 4	3
Matrix	3

Game titles used in the project, ordered by the number of students having played them regularly during their participation in the project.

With kind permission from all involved parties, we were allowed to perform an independent study interviewing both students and teachers. Previous results from studies on this test project can be found in (Wiklund and Glimbert 2005) and (Wiklund 2005).

RESEARCH QUESTION

As the learning potential of computer games is debated, more information in this area is needed. The usage of unmodified, commercial off-the-shelf games as teaching tools in schools is of special interest, as their edutainment counterparts have been observed to possess deficiencies while pure games are observed to be highly engaging. Since commercial games show highly varying properties though, and thus may be more or less suitable as teaching tools, the game genres factor is a key issue in a learning situation using commercial computer games.

The research issue addressed in this paper is to find out if students using unmodified, commercial off-the-shelf computer games of their own choosing in class, show any notable differences in study results related to their favourite game genres. For the purpose of this paper, the subject of English (as a second language) has been used as an indicator of achieved results.

METHODOLOGY

The empirical contribution of this paper is an evaluation study of an ongoing test project in Botkyrka, Sweden, using commercial, unmodified computer games as the main teaching tool in upper secondary education. The project in question includes students in both their 10:th and 11:th year of education, in a mixed fashion. The interviews were conducted towards the end of the second year of the four year test project, at a time when it was clear to the teachers which grades they would give the students at the upcoming end of that semester.

All 21 students in the project participated in the study through in-depth interviews, as well as the two teachers. The moderate number of students participating in the project is a limitation to the possibility to generalise results to the entire population, thus only conclusions regarding the participants in the project in question are drawn. Also, as all the participating students were male, gender issues are not addressed in this paper. However, as all the students in the project were interviewed, rather than just those choosing actively to participate, the risk of results being biased as a result of personality differences in this area was minimised.

The interviews were conducted individually in a separate room, away from the class room, with no possibilities of anyone else overhearing the conversations. The students retained full anonymity, only being identified by a sequential number untraceable to the specific individual. Each student was informed of this anonymity, and that his or her answers would not be disclosed to anyone else. By taking these measures, the risk of students not daring to answer the questions honestly was reduced as much as possible.

During the interviews, the interviewer followed a fixed form with questions to ensure equal coverage of topics with all students. Only follow-up questions may differ somewhat among the students, depending on the answers given. The information was entered into a database for processing. Key quotes were translated to English for the purpose of appearing in this paper.

RESULTS

A total of 21 students participated in the test project, all of which were interviewed for this paper. Regarding gaming background, all the students reported having played games frequently prior to entering the studied project, with 13 of them (61.9%) belonging to clans or guilds.

Favourite game genres

The two most common game genres stated by the students as their favourite genres were First Person Shooters (FPS) and Massively Multiplayer On-line Role-Playing Games (MMORPG:s). When entering the test program, 11 students (52.38%) stated FPS games being their sole favourite genre, and 1 additional student (4.76%) stated FPS games to be the

favourite genre in conjunction with (offline) Role-Playing Games (RPG:s), making FPS games a favourite genre of 12 students in total (57.14%) when entering the test program. At the end of the period studied, these figures had decreased to 7 students in total (33.33%), out of which 5 students (23.81%) regarded FPS games as their sole favourite genre and 2 students (9.52%) also mentioned other genres in conjunction (RPG:s and adventure games, respectively).

MMORPG:s were the sole favourite game genre for 3 students (14.29%) when entering the test program, and a favourite in conjunction with other genres (strategy and offline RPG:s) for 2 additional students (9.52%), making MMORPG:s a favourite game genre for a total of 5 students (23.81%) at the time of entering the studied program. At the time of study 8 students (38.09%) regarded MMORPG:s as their sole favourite game genre, with an additional 3 students (14.29%) having other game genres (strategy games, offline RPG:s and adventure games) as thir favourite genres in conjunction with MMORPG:s), taking the total number of students having MMORPG:s as a favourite game genre to 11 (52.38%).

Grades

A large number of grades had been given in several subjects specific to the program in question, for instance digital culture, game development, and web design, as well as general subjects such as history and social science. Focusing specifically on the subject of English (as a second language), 10 students (47.62%) had at the time of the study received grades in what is known in the Swedish school system as "English B", corresponding to the 11:th school year level, the second year in the test program). An additional 7 students (33.33%) had received grades in "English A" (10:th year level, the first year in the test program) only. A group of 4 students (19.05%) had not yet received any grade in English as a second language while in the test program.

Game genres and received grades

Correlating the favourite game genres stated by the students with the grades in the subject of English as a second language received by individual students, the results are as shown in the following table:

Game genre	Students	No English grade	English A only	English A+B
FPS only at start of project	11	4 (36.36%)	4 (36.36%)	3 (27.27%)
FPS + other at start of project	1	0	0	1 (100%)
MMORPG only at start of project	3	0	1 (33.33%)	2 (66.66%)
MMORPG + other at start of project	2	0	1 (50%)	1 (50%)
FPS only at end of period	5	2 (40%)	3 (60%)	0
FPS + other at end of period	2	0	0	2 (100%)
MMORPG only at end of period	8	1 (12.5%)	1 (12.5%)	6 (75%)
MMORPG + other at end of period	3	1 (33.33%)	1 (33.33%)	1 (33.33%)

Number of students with MMORPG versus FPS games preferences at the start of the test project and at the time of the study, having received 0, 1, or 2 grades in English studies while participating in the project.

The average number of completed years of English studies, as a function of the favourite game genres MMORPG and FPS games (with or without other genres in conjunction), is shown in the following table:

Favourite game genre stated	Students	Average number of completed years of English studies
FPS, at start of project	12	1.00
MMORPG, at start of project	5	1.60
FPS, at end of period	7	1.00
MMORPG, at end of period	11	1.45

Average number of completed years of studies in English (as a second language), measured as the number of yearly grades received, for students with MMORPG and FPS games as favourite game genres when entering the test project and at the time of the study, respectively.

Teaching methods employed

Interviews with the teachers revealed that their main approach to teaching using unmodified computer games involved using in-game activities as starting points for discussions and assignments of various kinds. This method was applied constantly. Both teachers reported that the students seemed highly motivated and interested in discussing issues in various fields, if the event spawning the discussion/assignment had occurred in one of the computer games.

The in-games activities are thereby leading to a learning process starting in-game and then expanding outside of the game. This is exemplified by one of the teachers: *"When I observed the students gathering* [in the online role-playing game World of Warcraft] *to decide which one of two dungeons to enter, I was thrilled to see that they performed an ordered voting procedure, standing up or sitting down to indicate if they were in favour or opposed to the suggested alternatives. This led me to have several very fruitful discussion with them, going into all sorts of voting taking place in the society, from shareholders of companies to politicians in the Riksdag* [the Swedish Parliament]" (Wiklund and Glimbert 2005). Both the interviewed teachers state that in their opinion such in-game starting points is a key factor when using unmodified, off-the-shelf computer games in a learning environment.

DISCUSSION AND CONCLUSIONS

The method of interviewing the entire class in question, as opposed to ask for volunteers, has the advantage of not just reaching a subset of individuals who might differ from the rest in various ways. In studies performed on volunteers that have actively chosen to participate, great care must be taken when interpreting the results. In such cases it is vital taking into account that the participants are more interested in the subject at hand, or at least more active and willing to take part in a study, than other people in general, even in the same age group, etc. This potential problem has been reduced as much as possible by interviewing not just enthusiastic volunteers, but everyone in the class.

Given the large amount of communication both between players and between players and NPC:s (Non Player Characters) in many modern games, the English language is used extensively also by players from non-English speaking countries. Since geographical distances become unimportant when using the computer mediated communication techniques

inherent in online games, a natural consequence is that players frequently encounter other players from various countries, making chatting in English extremely common. Local servers using local languages (other than English) are perfectly possible, but are in practice used less than the ones with English as the main chatting language. Given the extensive verbal (textual or in some cases by voice) communication between players, it is easy to see how students with other native languages quickly can improve their English speaking capabilities through such computer games.

Regarding communication between player and game environment, game genre and atmosphere can greatly influence the type of English texts encountered, such as quest texts in for example the MMORPG World of Warcraft being of a fairly verbose nature. Another example is games set in historic time, as one student playing the Age of Empires strategy game remarked: *"When you ask the teacher what some tricky medieval English word means, he tells you the Swedish word for that. Then you don't know what that means, either. Then he explains it, and you know a new word both in English and Swedish"* (Wiklund and Glimbert 2005).

During the interviews, 3 students also made spontaneous comments directly related to different game genres and their varying suitability for learning. While all 3 comments pointed out that game genre has an impact on a games potential as a learning tool, 2 did so more strongly: *"You can learn things from some games, like English and about history and stuff. From WoW* [online game World of Warcraft] *and games like that, I mean. And medieval strategy games. But not from the shooters, you don't learn anything there"* and *"They shouldn't allow CS* [FPS game Counter Strike] *and games like that in school, only online games. Because there you have to think more."*. The third comment was slightly more vague, while still attributing some importance to game genre in order to achieve learning: *"I don't think it matters what kind of game it is, I've learnt things from all games! Well, no, perhaps not from all games, not from really stupid games, those that's just reflexes. But from all others"*. Although vaguely defined, it seems clear that at least one game genre, possibly FPS games, is unsuitable as a learning tool in this students opinion.

Using the number of yearly grades awarded as a measurement of the number of completed years of studies in English as a second language, the average among the students having FPS games (First Person Shooters) as their favourite game genre (sole or in conjunction with other genres) at the start of the test project is 1.00 completed years of English studies. Comparing this with students having MMORPG:s (Massively Multiplayer On-line Role-Playing Games) as their favourite game genre (sole or in conjunction with other genres) at the start of the test project, this figure increases to an average of 1.6 completed years of english studies. The corresponding figures at the time of the study is that the average number of completed years of English studies remains at 1.00 for students having FPS games as their favourite genre, while students with MMORPG:s as their favourite game genre had completed 1.45 years of English studies on average.

Although there is a change over time regarding the students in favour of MMORPG:s (a decrease from 1.6 to 1.45 completed years of English studies on average), in both cases this can be compared to the lower average of 1.0 completed years of English studies for those in favour of FPS games.

It is worth noting that the visible difference, although interesting enough in itself, may have at least two different explanations, neither of which has yet been proven or falsified. On one hand, its possible that the observed differences described in this paper may be the result of students more motivated to study harder also having a tendency to favour MMORPG:s (massively Multiplayer On-line Role-Playing Games) to a higher degree than students favouring FPS (first person Shooter) games. On the other hand, its also possible that the described differences regarding study results can be contributed to the fact that those students playing MMORPG:s to a higher extent are receiving more training in, and exposure to, the English language, and as a consequence are both more experienced in, and possibly also motivated to study, the subject of English (as a foreign language). Both possibilities may have interesting implications either regarding games preferences as indicators or measurement tools, or regarding games as teaching tools.

FUTURE RESEARCH

The study described in this paper indicates a varying learning potential being present in using unmodified, commercial off-the-shelf games in class. As the students results, measured as the number completed years of English studies, varies with the students favourite game genres, the learning potential inherent in different game genres is an interesting topic worth more study.

To better understand whether the observed differences in study results are the result of students more exposed to certain game genres receiving specific training through the games as learning tools, or if students more inclined towards studying more often than others also choose certain game genres as their favourites, more research in this area would be valuable.

REFERENCES

BECTA. 2001. *Computer games in education project report.* British educational communications and technology agency, 2001.
http://www.becta.org.uk/research/research.cfm?section=1&id=2835

Dickinson, John R. and Faria, A. J. 1997. "A random strategy criterion for validity of simulation game participation." In *Simulation and gaming*, 28, 263-276.

Gee, James Paul. 2003a. "High score education – Games, not school, are teaching kids to think." In *Wired Magazine*, issue 11.05, May 2003.
http://www.wired.com/wired/archive/11.05/view.html?pg=1

Gee, James Paul. 2003b. *What video games have to teach us about learning and literacy.* Palgrave Macmillan, New York, 2003.

Griffiths, Mark. 2002. "The educational benefits of videogames." In *Education and Health*, vol. 20, No 3, 2002.

Kirriemuir, John. 2002. "Video gaming, education and digital learning technologies." In *D-Lib Magazine*, vol. 8, no. 2, February 2002.
http://www.dlib.org/dlib/february02/kirriemuir/02kirriemuir.html

Kirriemuir, John. 2005. "A survey of COTS games used in education." Presented at Serious Games Summit, Game Developers Conference, San Francisco, March 2005.
http://www.bris.ac.uk/education/research/networks/gern/gdc05.ppt

Kirriemuir, John and McFarlane, Angela. 2003. “Use of computer and video games in the classroom.” Presented at the Digital Games Research Association (DIGRA) conference, November 4-6 2003, University of Utrecht, Holland.
http://www.ceangal.com/papers/42.pdf

Kirriemuir, John and McFarlane, Angela. 2004. *Literature review in games and learning.* NESTA futurelab report series, report 8, 2004.
http://www.nestafuturelab.org/research/reviews/08_01.htm

Papert, Seymour. 1998. “Does easy do it? Children, games and learning.” In *Game developer magazine*, June 1998, p. 88.
http://www.papert.org/articles/Doeseasydoit.html

Wiklund, Mats. 2005. "Behavioural changes in students participating in an upper secondary education program using unmodified computer games as the primary teaching tool". In the proceedings of CGAMES 2005, the 7:th international conference on computer games, 28-30 November, 2005, Angouleme, France. ISBN 0-9549016-2-6.
http://dsv.su.se/~matsw/paper4.pdf

Wiklund, Mats and Glimbert, Lars. 2005. “Students perception of a learning environment and the teachers role while using unmodified computer games as learning tools in upper secondary education”. In the proceedings of CGAIMS 2005, the 6th international conference on computer games, AI and mobile systems, 27-30 July, 2005, Louisville, Kentucky, USA. ISBN 0-9549016-1-6.
http://dsv.su.se/~matsw/paper3.pdf

Woods, Stewart. 2004. “Loading the dice: The challenge of serious videogames.” In *Game Studies – the international journal of computer game research*, vol. 4, issue 1, 2004.
http://www.gamestudies.org/0401/woods/

Wolfe, Joseph and Crookall, David. 1998. “Developing a scientific knowledge of simulation/gaming.” In *Simulation and gaming*, 29, 7-19.

AUTHOR BIOGRAPHY

Mats Wiklund completed his BA degree in computer science in 1994 and his licentiate degree in computer science in 1999. He currently teaches computer games developement courses at Stockholm University, working on his PhD thesis in parallel. Current research areas focus on computer games related commu-nication and learning issues, both within games and through other channels regarding games.

Designing Educational Games

Christy M. Bogard and Dr. Rammohan K. Ragade
Computer Engineering and Computer Science Department
University of Louisville

Keywords *edutainment, game types, software component requirements*

Abstract

Educational games have steadily entered classrooms as a means of challenging advanced students and tutoring those lacking comprehension. However, without adequate educational benefits, instructors are struggling to continually justify the marginal value added of using these programs. It is the goal of this paper to discuss the types of educational games currently available, to outline the elements necessary to make a game entertaining, and to assess the required components for educational value.

Introduction

Students have a higher retention rate of subject material when it is reinforced by additional sources outside of classroom instruction. Instructional simulations and educational games have the potential to provide this additional reinforcement. However, educational software, as it exists today, lacks the components necessary to both entertain students and provide instructors with the necessary tools to justify the use of the game within the classroom. In an effort to bridge this game, this paper addresses what components are necessary to developing an entertaining game and making that game beneficial within the classroom [2, 4, 11].

Status of the Educational Game Industry

While the gaming industry has been growing at an unprecedented rate, expecting to grow by 71 percent to $85.7 billion by 2006, the educational software sector has dramatically lagged, representing only 6.5% of the computer and video game dollar sales. As such, published literature on educational computer games has only begun gaining substantial volume since 2000 [9, 11, 12, 13, 14, 19].

However, even given the substantial growth in literature, leading researchers are divided on how useful computer and video games are. Those in favor of these games claim they can further develop social and cognitive skills, increase in the retention of information, and keep students engaged and motivated in learning. Those against these games claim they can increase youthful aggression, result in social isolation, and because of their addictive nature, cause weight and health complications [1, 3, 6, 7].

Given the lack of consensus on the usefulness of computer and video games within the classroom, the majority of published literature on educational software has focused on the following four categories:

The first category of articles contains general overviews of computer and video games coming to the market. These articles focus on what new games have been developed and how they meet a specific need. In most cases, because the primary intention of the authors is the sell the given product, only a biased evaluation is presented, giving a skewed representation of educational value contained within.

The second category of articles is focused on pre- and post- testing of specific educational software within a controlled environment. Because they resemble short-term, limited case studies often on a single narrow topic, conclusions reached are often difficult to reproduce and even more difficult to generalize in order to make the results beneficial outside their narrow scope.

The third category of research is focused on the effects of gaming on individuals. This is the largest and most

controversial area of research, examining the physiological, cognitive and social effects of playing games on users. These articles are often co-authored by psychologists, focusing more on the benefits or consequences of educational games, not on the educational games themselves.

The fourth and final category is focused on research reviews and meta-analyses. This sector of articles is often authored by educators and aimed at critiquing the components of educational software. Game developers are most interested in this area of research because it provides a glimpse into user specifications for educational software. However, deciphering specific specifications is often more difficult, as educators are primarily focused on what hinders usefulness in the classroom and not what is necessary to make the games beneficial [11, 12, 14].

Current Educational Game Types

Before one can determine clear client requirements, it is critical to have an understanding of the games currently available. Educational games can be divided into five general categories: Drill and Practice Games, Half and Half Games, Discovery Games, Content Games, and Non-Traditional Games. These five games range from a primary focus on educational content to a primary focus on entertainment, respectively [5, 8, 16, 17, 18].

Drill and Practice Games, the first type of game, place focus on continually presenting similar problems centered on a single concept. The student practices over and over until he or she can successfully complete a predetermined number of problems. At such time, the student is rewarded, usually with a miniature activity game or animation. Because students are often relentlessly drilled on the concepts within the classroom, this type of game is typically only entertaining to, and thus effective for, elementary-age youth. Often, these games are presented in "Jeopardy"-like atmospheres, where players create a simple virtual character that gains points or money when he or she correctly answers the posed question and loses points or fined a set amount of money when he or she incorrectly answers the posed question.

Half and Half Games are the second type of educational games. These games, considered the foundation of edutainment, present educational content within an entertaining game environment. Players are rewarded by increasingly more difficult scenarios as they successfully complete the previously presented challenge. Because the game environment is highly interwoven with the educational content presented, the scope of the educational game is often very narrow, making the game too specific to be valuable to a broad audience. One such example is Oregon Trail. Oregon Trail defines survival problems for the game player as they move across the western plains. While players are presented with some differing scenarios as they progress, these scenarios are limited in variety to ensure completion of key educational modules.

Discovery Games expand the Half and Half games by shifting the focus even further to the entertaining aspect of the game. This is achieved by introducing an exploration aspect to the game. Students are encouraged to seek out the solution to the challenge presented through a less structured game environment. Given the increased time required to complete a challenge or reach a suitable stopping point, these types of games are often unsuited to the classroom because most students are forced to leave the task unfinished, an undesirable state. "Where in the World is Carmen Sandiego?" is one such Discovery Game. Students must move throughout the world in search of clues as to where Carmen has fled to with a precious artifact. While the student has vastly more control over his or her interactions, the game provides a myriad of clues to assist a lost player.

Content games expand upon the Discovery Games by shifting the focus primarily to the entertaining environment aspect, making the educational content presented the secondary focus. These games

introduce an increased risk aspect and further exploration aspects by reducing the structured rules of the Discovery Games. However, the reduced structure combined with the shift in focus away from the educational content make these games extremely difficult to use within the classroom setting because success is often based on lucky or random discovery of key pieces of knowledge. An example of a Content Game is the "Riddle of the Sphinx." A player is released into the desert to discover the ancient Egyptian world with only limited instructions. While the student can ultimately complete the objective at hand, it is often difficult to accurately measure one's accomplishments due to lack of guidance.

Finally, Non-Traditional Games are the fifth type of educational game. These games have some clear educational value presented to the student, but weren't originally developed for educational purposes. As such, these games do not easily classify into the four traditional game types aforementioned [16, 17, 18].

Designing an Entertaining Game

Understanding what the five types of games are lends itself to a discussion of what elements make a game entertaining. First, and most importantly, games must have an interactive environment. The player's decisions should drive the game's responses, making the interactive element the distinctive thread separating games from other artistic ventures such as movies, music, or paintings. Thus, entertaining games must present scenarios in which users choose between different options.

Decision-making brings the next element into focus. The game must appropriately respond to different selections made by the user. If the game doesn't produce different responses to alternative selections, then the game lacks true interactivity.

The game also needs an element of achievement, the third critical element. While achievement can take on different meanings with different game contexts, successfully completing progressive challenges indicates a natural advancement through the game in actively seeking the end challenge. With elements of achievement, there also needs to be varying degrees of failure. Not successfully completing a challenge should result in a setback in the journey to the conclusion. However, failures should not result in unconquerable game scenarios.

The next element needed to make a game entertaining is clearly defined challenge. Additionally, the problem presented should be interesting and having a logical solution that can be reached by interacting with the game. The key is finding the appropriate problem level that is not too simplistic as to bore the user to move on to other games and not too complex as to frustrate the user to quit entirely.

An entertaining game must also be fully self-encapsulated, creating an environment in which the user becomes self-absorbed within the game world. This is often called suspension of disbelief, because the user is so engaged in the game that he or she is unaware of one's surroundings.

Finally, an entertaining game should have a personal experience for the user, meaning that while users will have similar experience, there are specific aspects that appeal to each individual user. This is often subdivided into what the user perceives as fun, what the user learns from the experience, and what alternative reality the user supplements with the actual game environment.

These are only a few of the components important to creating an entertaining game, but they represent the basic building blocks of the game. It is also critical to recognize that games are created uniquely in their selected trade-offs in each of these basic elements [4, 5, 10].

Determining Client Requirements

Understanding the types of educational games available and the critical components that make a game entertaining leads to the determination of what additional requirements the clients, or school educators in this context, are seeking. To aid game

developers in determining the specific needs of educators, leading officials have begun evaluating aspects that are critical for a game to have educational value within the classroom. For example, Bringing Educational Creativity To All (BECTa) and Teachers Evaluating Educational Multimedia (TEEM), two leading research organizations in educational computer games, have both developed comprehensive lists of components that are required for games to contribute value within the classroom, but are currently not present. This topic is further elaborated upon in the Master of Engineering Thesis by the first author [20].

First, it is critical for the educational game to record what the student completed during the gaming session. Educational games are valuable in the classroom if it can both increase a student's understanding of a concept and provide an analysis of the student's learning to the instructor. Current educational games on the market only record a student's level of mastery, often given as a percent success rate or subjective description of mastery such as "excellent" or "good." Because of the limited artificial intelligence within the educational games, instructors cannot determine the underlying concepts that a student does or does not understand based on a level of mastery.

In order to provide instructors with the needed information, the game play interface must record what the student was able to successfully accomplish and what the student failed to master. Because learning is a complex process that does not easily fit into precise categories, games should not attempt to determine the underlying misconception, but instead provide the most amount of information possible to the instructor through a record of the student's interactions with the game.

Secondly, educational games should be able to adapt to students with different skill levels. In order to continually challenge a student requires a custom-tailored program that reacts appropriately to his or her demonstrated skills. Advancing question difficulty only after a student has fully demonstrated mastery of the previous challenge level results in boredom when a student has gained mastery but has not yet completed the current level requirements and frustration when a student cannot achieve success at the next challenge level.

In addition to being able to adapt to students with different skills levels, educational games should provide similar, but not identical, repeated experiences. This is especially beneficial when all students do not interact with the game simultaneously, but instead play sequentially. This ensures the latter students are not simply reproducing memorized experiences relayed from the former students. Furthermore, similar but unique gaming experiences promote classroom discussion and enable students to comprehend the experiences of their peers without having exactly the same experience.

The fourth component required to make educational games beneficial within the classroom is providing suitable breaking points during the game play. Using an educational game within the classroom is often inhibited by time constraints and possible interruptions. Providing stopping points allows the student to complete a task while not feeling unsatisfied for having an uncompleted task. Additionally, providing completion points can often reduce unnecessary time repeating previous accomplishments to resume game play.

Another critical component currently lacking is appropriate management tools provided to instructors. Educational games currently on the market lack developed Instructor's manuals that include pertinent information on structure content and underlying game models. For example, game scenarios should mimic realistic expectations and physical properties of the real world, furthering psychological, social, and intellectual development in students. Providing this information allows instructors to analyze games for their educational value as well as their appropriateness to the instructor's classroom.

While most educational games provide limited instructions for the instructors, the educational game play may require elaborate written instructions to be understood by the user. When such instructions are required, the reading comprehension level should match the target audience age. It is essential game designers recognize that an educational game played within the classroom setting must be capable of functioning independently of instructor's involvement, as instructors are often engaged with students not currently engaged with the educational game.

Finally, educational games should foster an encouraging environment that motivates students to continue involvement with the game, such as through satisfaction, desire, anger, absorption, interest, excitement, enjoyment, and pride in achievement. Educational games that do not continually engage the student's interest are often dismissed as futile, quickly rendering any educational value added ineffective.

While BECTa and TEEM have differing opinions as to the priority of these components, both agree that without these components, the costs of using educational software within the classroom will continue to outweigh the benefits. Unfortunately, these components are often expensive to implement. Commercially, these investments are justified by the substantial return on investment through the mass sale of the produced game. For example, Electronic Arts, the leading producer of computer games, reported 2004 revenues at nearly three billion dollars. But educational software cannot produce these high revenues. As such, producers of educational games lack the necessary resources to produce a high quality product comparable to the currently available entertainment computer games [2, 4, 8, 9, 11, 15].

Conclusion

Educational games must have the five/six requirements for entertaining, plus the six software components to provide instructors with the necessary tools to justify the use of the game within the classroom. It is critical to recognize that games are created uniquely by balancing the entertaining and the educational side. This tradeoff will ultimately define the success of the game.

References

1. Anderson, CA & BJ Bushman, "Effects of Violent Video Games on Aggressive Behavior, Aggressive Cognition, Aggressive Affect, Physiological Arousal and Prosocial Behavior: A Meta-Analysis Review of the Scientific Literature." Psychological Science, Vol 12 (2001), 353-359.
2. Arter, Judith A & Jay McTighe, Scoring Rubrics in the Classroom. (Thousand Oaks, CA: Corwin Press, 2001).
3. Bensley, L & J Van Eenwyk, "Video Games and Real-Life Aggression: Review of the Literature." Journal of Adolescent Health, Vol 29 (2001), 244-257.
4. Bringing Educational Creativity To All, "Computer Games in Education Project." (2001).
5. CMP United Business Media. Game Developer: The Leading Game Industry Magazine. (San Francisco, CA: 2005).
6. Griffiths, MD, "Violent Video Games and Aggression: A Review of the Literature." Aggression and Violent Behavior, Vol 4 (1999), 203-212.
7. Harris, J, "Secondary School Students' Use of Computers at Home." British Journal of Educational Technology, Vol 30 (1999), 331-339.
8. Informa Telecoms & Media. Dynamics of Games (5th edition). (London, UK: Informa Media Group, 2005).
9. Interactive Digital Software Association, "State of the Industry: Report 2003-2004." (Washington, D.C: IDSA, 2005).
10. Maidment, Robert & Russell Bronstein, Simulation Games: Design and Implementation. (Columbus, OH: Merrill, 1973).

11. McFarlane, Angela, Anne Sparrowhawk, & Ysanne Heald, "Report on the Educational Use of Games: An exploration by TEEM of the contribution which games can make to the education process." (Cambridge, UK: Teachers Evaluating Educational Multimedia (TEEM), 2002).
12. Mitchell, Alice & Carol Savill-Smith, "The Use of Computer and Video Games for Learning: A review of literature." (London, UK: Learning and Skills Development Agency, 2004).
13. National Governors Association, The Fiscal Survey of States. (Washington, DC: National Governors Association, 2002).
14. Orey, Michael, Mary Ann Fitzgerald, & Robert Maribe Branch, Educational Media and Technology Yearbook 2004, (Westport, CN: Libraries Unlimited, 2004).
15. Roschelle, Jeremy, et. al. "Developing Educational Software Components." Web-Based Learning and Collaboration. (Sept. 1999).
16. Sawyer, Ben, The Ultimate Game Developer's SourceBook, (Scottsdale, AZ: Coriolis Group Books, 1996).
17. Swords, Tara, "At the Top of Their Game." Dell Insight, (Jan 2005), 6-11.
18. Taylor, John & Rex Walford, Learning and the Simulation Game. (Beverly Hills: SAGE Publications, Inc, 1978).
19. Trotter, A., "Budget Crises Leads to Delays for Technology." Education Week, Vol 22, Issue 34 (7 May 2003), 1-2.
20. Bogard, Christy, "Advancements in Educational Games Through Sound Software Engineering Principles." Master of Engineering Thesis, Dept of Computer Engineering and Computer Science, University of Louisville, Louisville, KY. (July 2006).

Session 6

FMMG: A FRAMEWORK FOR MOBILE MULTIPLAYER GAMES

Mario Massakuni Kubo and Romero Tori
Avenida Prof. Luciano Gualberto, travessa 3 nº 380
CEP - 05508-900 - São Paulo – SP
Universidade de São Paulo
e-mail: {mario.kubo, romero.tori}@poli.usp.br

KEYWORDS

Games, Framework and mobile multiplayer games.

ABSTRACT

Mobile devices have increasingly evolved, thus becoming an object of great interest to developers from various fields of study (database, communication networks, engineering, multimedia, virtual reality, and computing, among others.)
In this context, the development of games for mobile devices (especially cell phones) has attracted many, both from universities and industries.
The goal of this paper is to introduce a framework for mobile multiplayer games called FMMG. The FMMG provides game developers with a working consistent environment, which makes it possible for them to work exclusively on the behavior intended for their game. We offer here a detailed description of the FMMG framework structure and of its modules, showing how they are integrated and consequently result in a powerful and efficient tool for the development of Mobile Multiplayer Games.

INTRODUCTION

The branch of Games is growing rapidly, now entering a billionaire market. One of the reasons for such success, besides consumers' high interest, is the diversity of devices with specific characteristics that have been produced through the years, ranging from PC's and videogames to the most recent mobile devices. However, in spite of such growth, important questions still remain. These questions are mainly related with programming languages; hardware; development platforms' heterogeneity; database and application software formatting; interconectivity; management of applications and their systems, taking into account environmental differences; ergonomics; software reuse at different frameworks, architectures and platforms; bandwidth and performance limitations; input/outpt interface limitations; software and hardware incompatibility; development and analysis of data communication protocols' performance in heterogeneous networks; specific domain application support services (games, education, cooperative work, software engineering, business, entertainment, and so forth); among others.
The branches of Games, and especially the branch of Game applications for mobile devices, are extremely stimulating due to the technological challenges and the applicability potential they offer. The fact that this is an emergent multidisciplinary field and that it allows diverse views with various requirements is responsible for the lack of systems that can answer all those questions. Therefore, the branch of Mobile Multiplayer Games makes an extensive research field on frameworks, engines, architectures, networks, and distributed computing systems, for instance. Besides, much research is still to be done in non-multiplayer games, such as: more precise input/output devices, man-computer interface, and graphical computing among others.
Developing Mobile Multiplayer Games is a more challenging task than developing games for PC's or videogame consoles. Besides the difficulties in developing a game, the developer must deal with other intrinsic problems of those devices, such as limitations of memory, resources and processing, screen size, and inappropriate input mechanisms (Brooks 1987) (Bethke 2003). In spite of the differences mentioned above, there are certain elements common to all kinds of games: collision detection; animation; screen rendering; user entries; modeling and animation of objects; artificial intelligence; network communication and algorithms, for instance. Keeping these common elements in mind, it is possible to put them all together in a software component, which should be so generic as to allow it to be used by various kinds of games, but which should, at the same time, be extremely optimized so that it could supply good performance and good software quality, and shorten the development process. Such is the role strongly applied to the concepts of software engineering framework.
The fact that many of the elements mentioned above still lack deeper consideration (with respect to the branch of Mobile Multiplayer Games) together with the lack of an effectively consolidated structure and the inherent problems of multi-user applications motivated the development of the FMMG (Framework for Mobile Multiplayer Games.)
In short, we discuss in this paper the concepts of Game and framework as follows: in section 2 we present some important concepts of Game development; in section 3 we introduce the FMMG; and finally in section 4 we discuss the conclusions from the development of this project.

GAMES

According to Battaiola (Battaiola 2000), "a Computer Game may be defined as an interactive system that allows the user to experience a conflicting situation. Usually there is a background plot to such experience, which defines the game theme, and establishes its rules and objectives."
A Computer Game may also be defined as a system consisting of three basic parts: plot, engine (framework) and interactive interface. The success of a Game is dependent upon the perfect combination of such elements (LaMothe et al. 1994) (Araujo and Bataiolla 1998) (Battaiola et al. 2001).
The plot establishes the theme, the scenery and the objective of the game, which the player must achieve by following a sequence of steps. The task of defining a plot demands not only creativity and research on the subject, but also an amount of exchange with pedagogy, psychology, and other disciplines.
The interactive interface controls the communication between engine and user, displaying a new state of the game. The development of this interface involves artistic, cognitive and technical aspects.
The game engine is its control system, the mechanism that controls the game response according to user actions. Implementing an engine involves several computing aspects, such as the choice of an appropriate programming language, taking into account the easy of use and portability; the development of specific algorithms; the kind of user interface, and so forth.
Games can be divided according to the number of players into two categories: Monoplayer Games and Multiplayer Games.

Multiplayer Games might also include Massive Multiplayer Online Games (MMOG's). MMOG's are games where thousands of players interact simultaneously in a persistent virtual world. These games are held continuously but part of the players may leave the game while others remain.

The merging of MMOG's and mobile Games generates a new world, where it is possible for thousands of users to interact with each other through mobile devices, thus creating mobile massively multiplayer online games (MMMOG's.) For a number of reasons the perfect integration of those systems has not been completed yet. Most games currently available for cell phones worldwide consist of either online or offline single user applications, similar to the PC games released in the 1980's. However, the analysis of the development of both wireless services and markets (e.g. Asia and Europe) reveals that there is a need as well as a commercial potential for MMMOG's. This platform inherits the usual problems of a traditional MMOG's, such as the need of guarantee of robustness, scalability, real-time communication, security and tolerance to failure.

Today we watch the birth of Massively Multiplayer Ubiquitous Games (MMUG's), which comprehend a group of sceneries where players interact with one another and also with a virtual world through various devices, which are adapted to the context of execution, as means to give users the idea that they never leave the game environment completely. There are significant technological and scientific challenges in realizing a MMUG, which will have quite an impact (with strategic value in the near future) on subjects like interactivity, virtual communities, ubiquity and digital convergence.

Based on an adaptations from the Distributed Virtual Environments' requirements proposed by Van Dam (Van Dam 1993) and by Deriggi (Deriggi 1998) it is possible to list Multiplayer Game requirements as follows:

- Fast game refresh;
- Minimum latency (delay);
- Multiple input/output device support;
- Simulation of a significant number of objects with simple, medium or complex behavior;
- Realistic game representation;
- Distribution of long distance data;
- Computing resource and heterogeneous network support;
- Scalability.

Multiplayer Games are distinguishable by five common characteristics (Sementille 1999):

- Space sharing feeling: all players of a Multiplayer Game have the illusion of being placed at a same location. The shared space must show the same characteristics to all players;
- Presence feeling: when a player enters a shared space, each player becomes a "virtual player," called *avatar*. An avatar is a graphic representation that might have the following characteristics: a body structure model, a movement model, and a physical model. Once in a Multiplayer Game, each participant can see other avatars standing in the same space. Also, whenever a participant leaves the game the other players can see his/her avatar leaving. Not all players must be human controlled;
- Shared time feeling (synchronization): it is possible for players to see how they behave in real-time;
- Communication between players: although visualization is the base for an effective Multiplayer Game, many of these games also attempt to allow some kind of communication between players. Communication may occur by means of movement, animation, text and audio, and it adds greater sense of presence and reality to any simulated game;
- Means of sharing: the elements mentioned above effectively offer a high quality videoconference system. The real power of a Multiplayer Game derives from its players' ability to interact with one another and also with the game itself. In a combat simulation or Game, for instance, participants have to shoot or collide into each other. Interactions like these must be modeled in a realistic way. Players should be able to choose, move, handle, and even give other players objects that exist in the game.

The importance of Multiplayer Games lies in the combination of graphics with the possibility of having a number of players sharing information and handling objects inside a Game. The presence of many autonomous players in the same game is what distinguishes a Multiplayer Game from a Traditional Game.. Multiplayer Game forms would be more appropriate to Games that demand telepresence rendering, that is, the illusion that all players are present at the same place, even if they are actually physically appart from each other.

Multiplayer Game Components

Multiplayer Games are made of four basic components: graphic displays, communication and control devices, processing system, and communication network. These components work together in order to supply a sense of involvement to the players.

- Graphic displays: give players a view of the graphic structure of the Game. Some examples of graphic displays are: monitor screens, mobile devices' screens, glasses (Head Mounted Display - HMD) and CAVEs (Cave Automatic Virtual Environment - CAVE);
- Communication and control devices: game players have to handle objects and communicate with other players in the Game. Such tasks are performed through several input devices. The most ordinary, however not very effective, input devices include mouse, joysticks, keyboards, and also mobile devices' controls. For more precise handling tasks, a glove (dataglove) may be used. Textual communication might be distracting to players, holding them from total involvement in the Game. Therefore, in more sophisticated Multiplayer Games, verbal communication is possible through multimode interaction;
- Processing System: Multiplayer Games demand a considerable processing capacity. The processing unit receives events from a player's input devices and computes the effect of these entries over both the player's location and other objects in the Game. The processor must determine how and when to notify other players about the changes caused by a local player, and vice-versa. Finally, the processor must render the graphic display and keep updates from the point of view of the player in the Game. Therefore, one of the challenges for Multiplayer Games designers is to allocate the processor's timing to the tasks required to sustain the illusion of presence of players;
- Communication network: in a Multiplayer Game players need a communication network to exchange information about changes of position or placement of objects, for instance. The network is also responsible for the synchronization of shared states in the Multiplayer Game, such as time and visibility. It is also important that the network allow textual and audio communication.

FMMG – FRAMEWORK FOR MOBILE MULTIPLAYER GAMES

A framework is a set of software components organized and designed for the construction of a specific domain's applications.

It should be noticed that a Game framework is neither a set of functions nor an API serving a certain objective. It is not a finished product either. The framework must be integrated with the game design implementation (Azevedo 2005) (Pessoa 2001) in order to form a complete Game. Therefore, the goal of the framework is to make it possible for its users (the game designers) to focus only on game designing.

Developing mobile device Games is a complex task. In addition to the difficulty of developing the Game itself, developers must deal with various inherent problems of these devices. The use of game frameworks hence captures the developer's expertise at a given platform, supplying a set of components designed and optimized for those devices. Besides, the use of frameworks also aims the reusability of the code, of the intrinsic elements of the Game, and of the platform's characteristics.

In this context, FMMG (Framework for Mobile Multiplayer Games) is intended to be as generic as possible so that it can be used as the basis for any kind of Mobile Multiplayer Game application, thus providing the program portability between development platforms for mobile devices based on object oriented programming language.

FMMG is meant to provide game developers with a set of classes to help them implement Mobile Multiplayer Games. In the future an engine may be implemented or aggregate to the framework. By supplying a working consistent environment, FMMG makes it possible for game developers to work exclusively on the behavior intended for their game.

All other functionalities – possible incompatibilities in the operational environment functioning (each equipment, even working at a same platform, might have different functionalities and functioning), for instance – are to be solved by the FMMG.

With the resources the framework supplies, developers will be free to implement their Game the best way they can, regardless their (lack of) knowledge about the environment where the framework is being used.

The FMMG architecture was defined according to identified requirements, and it was meant to be an open architecture framework; that is, developers have access to the source code and may change it.

The MVC (Model-View-Controller) project standard was adapted (Krasner and Pope 1988) to best suit this framework. The Model class hence corresponds to the Model, the View class corresponds to the View itself, and the Controller class corresponds to the Controller itself. A summary of the description of each interface's role as shown in Figure 1 is found bellow:

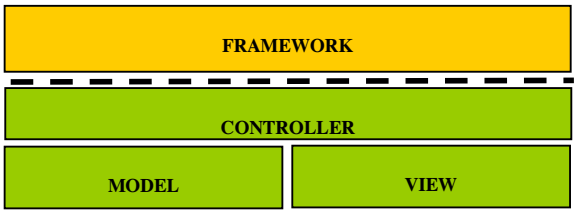


Figure 1 – MVC Infrastructure

Controller

- Defines the application's behavior.
- Maps the user's actions in changes in the Model interface.
- Selects and controls the active View.

Model

- Encapsules the application's state.
- Point of access to functionalities.
- Notifies changes in the View.
- Responds to consults of the View.

View

- Renders the application.
- Receives update requests from the Model.
- Sends the user's entries to the Controller.

Controller

Controller is an abstract class that must be implemented by the Game developer and informed to the framework. Through this class the framework is able to inform the Game about events taking place at the cell phone. The Controller class has two methods used by the framework to communicate with the Game, as Figure 2 shows:

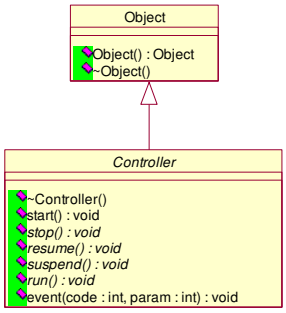


Figure 2 – Controller class

The goals of these methods are:

- start(): method executed just once at the beginning of the Game. This method should be used to allocate all basic resources the Game will need while running.
- stop(): method executed just once at the end of the Game. This method should be used to deallocate all resources that had been made available to the Game by the start() method.
- suspend(): method executed whenever the cell phone has to perform one of its functions (e.g. receiving a voice call, a shot message, or low battery warnings.) This method should be used to put the Game on hold, so that the resources used by the Game are made available for the cell phone's operational system to execute other functions.
- resume(): method executed whenever the operational system ends the functions that caused a Game interruption (suspend()). This method should be used to reallocate all resources needed by the Game so that it starts running again.
- run(): method executed at each time fraction (depending on the number of frames per second configured in the framework.) This method is the entrance to the execution of the Game. It must be implemented so that after its execution, it returns the control to the framework (there can be no infinite loops.) This kind of implementation should be respected, because the operational environment used in the framework implementation must support cooperative multitasks, and if the control is not returned to the environment, the device might not work as expected and the keys pressed by the user will not be recognized. The run() method is the main loop of the Game, however, such loop is realized/controlled by the framework.
- event(): method that will be executed whenever an event that might be of interest to the application occurs in the operational environment.

The implementation of the Controller class is responsible for the maintenance of Game states. It defines what action and what view

must be executed. This class may be called Macrostate Machine of the Game.

Model

Model is the abstract class that must be implemented to represent action in a state of the Game. All intelligence that must be implemented to a certain state of the Game is encapsulated during the implementation of the Model interface, as shown in Figure 3. Basically, the Model class uses the same methods as the Controller class, for the same ends. It may be called Microstate Machine of the Game, since it is responsible exclusively for the implementation of the "intelligence" of a macrostate of the Game. A macrostate can be composed of various microstates, which are under the responsibility of the Model class implementation.

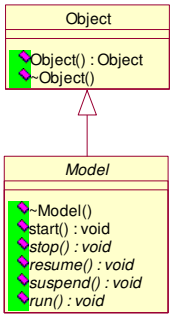


Figure 3 – Model Class

These methods have the following purposes:start(): this method should be used to start state actions or Game actions.

- stop(): this method should be used to deallocate all actions that had been made available to the Game by the start() method.
- suspend(): method executed whenever the cell phone has to perform the suspension of its functions in the action of the Game state.
- resume(): method executed whenever the operational system ends one of the functions that caused a Game interruption (suspend()) in the action of the Game state.
- run(): method executed at each time fraction. This method is the entrance to the execution of the class.

View

The abstract class View must be implemented to represent graphically a state of the Game. It uses basically the same methods as the Controller class, except for the run() method, here replaced by the draw() method. Whenever activated, the draw() method is responsible for the screen update, as shown in Figure 4.

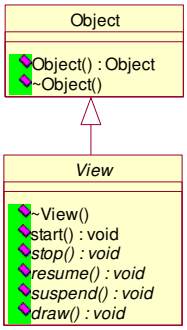


Figure 4 – View Class

Here, the methods have the following ends:

- start(): this method should be used to start the Game View.
- stop: this method should be used to deallocate the View that had been made available to the Game by the start() method.
- suspend(): method executed whenever the cell phone has to perform the suspension of its functions in the View of the Game state
- resume(): method executed whenever the operational system ends one of the functions that caused a Game interruption (suspend()) in the View of the state of the Game.
- draw(): method executed at each time fraction. This method both presents and renders the Game information.

Functioning of the Standard MVC Framework

The infrastructure we propose for the use of the MVC standard in the framework is based on the use of Controller, Model and View described in previews sections of this paper and also on the use of State structures and DataObject class. The State structure is dependent of the player's entry or action, and it determines the current occurrence of a Game. The DataObject class is responsible for the data structure of the state to be manipulated in the Game. Figure 5 shows the structure of the MVC standard used by the framework.

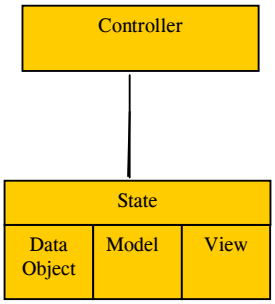


Figure 5 – Structure of the MVC standard used by the Framework

Communication between the Model class and the View class is established through the implementation of a DataObject class, as shown in Figure 6. The DataObject class contains all needed information to execute both the Model class' and the View class' methods.

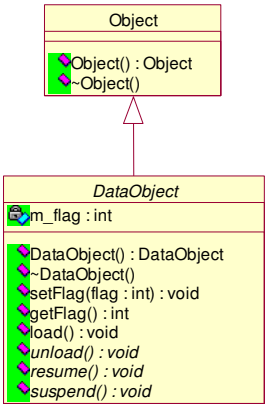


Figure 6 –DataObject Class

The Controller class, where the macrostates of the Game are controlled by the changeState() method, are requested to represent it based on the following states:

- DataObjectState.
- ModelState.
- ViewState.

Figure 7 shows the code to the changeState().

```
…
Controller::changeState( int state )
{
        switch ( state )
        {
                case state_1:
                   stateDataObject = new state
                   state_DataObject( ) ;
                   stateAction        = new
                   State_Model( stateDataObject )
                   ;
                   stateView          = new
                   state_View( stateDataObject ) ;
        }
}
…
```

Figure 7 –changeState method

The run() method of the Controller class is called upon each framework iteration, and based on its macrostate control, all classes representing that macrostate are activated, as shown in Figure 8.

```
…
Controller::run( )
{
        stateAction->run( ) ;
        stateView->draw( ) ;

        if ( stateDataObject->EndOfState( ) )
        {
                changeState( nextState( ) ) ;
        }
}
…
```

Figure 8 – run method

Such framework infrastructure results in freedom to develop a Game in any desired way, and at the same time, it offers the possibility to reuse the state implementation of one Game into another. This is possible because state implementation occurs through the implementation of that three-class set: DataObject, Model and View. Consequently, there is a development gain, since Game parts (SplashScreen, Menu, Rank, About and Help) can be reused, and thus developers may concentrate only on the development of the GamePlay state of the Game, as Figure 5.9 shows. Also, the time spent in the development process is reduced, since different developers can work separately on each state of the Game, which will be put together in the end, forming a complete Game. In this way, a weakly coupled structure is formed.

The MVC State based infrastructure for Games can be added to other State structures, according to the game developer's needs.

A Game can make use of the following MVC-standard-based states, as shown in Figure 9.

- SplashScreenState –responsible for the initial opening of the Game.
- MenuState – responsible for the menu structure of the Game.
- GamePlayState – responsible for the execution of the Game itself.
- RankState – responsible for the display of a ranking list of a Game.
- HelpState – responsible for the display of the Game help.
- AboutState – responsible for the information about the Game.

The View and the Model classes are responsible for calling other class structures belonging to the framework component modules. These modules contain the game development functionalities.

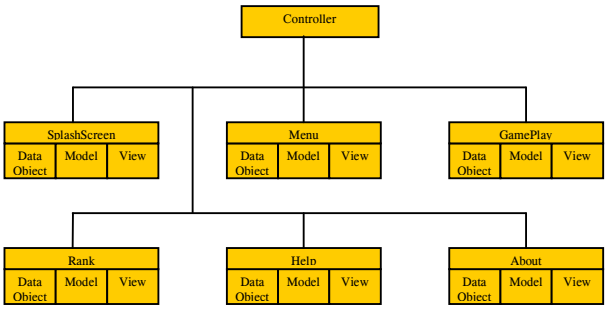


Figure 9 – An exemple of the use of the MVC standard for Games

Framework Structuring

The analysis of various frameworks and Game engines (Pessoa 2001) (Madeira 2001) (Bernades et al. 2004) allowed us to identify which component modules would be needed in our framework. A description of the integrate architecture of frameworks or Game engines for PC's proposed by Pessoa (Pessoa 2001) and Bernades (Bernades et al. 2004) is found bellow:

- Object Manager: responsible for the interaction among objects and between objects and scenery. There may be various object managers, one for each group of similar objects of the Game.
- World Manager: manages the current state of the Game, interacting with other managers to inform what has to be done.
- Main Manager: works as a bridge for information exchange and supplies a single access point to the Game functionalities.
- Rendering: image generating components with a determined quality standard, and speed enough to maintain the Game interactivity.
- Network communication: components responsible for the communication among players in a Game.
- Sound and Music: components that emit sounds or music from valid format files in the platform of development of a Game.
- Artificial Intelligence: components that implement computer-controlled characters' intelligence in a Game.
- Collision: components that detect collision of objects in a Game.
- Physical Simulation: independent components characterized by the capability of simulating sets of objects with definable binds and levels of liberty in real-time.
- User Entry: components responsible for capturing and transmitting commands received via input devices.
- Multimedia: components responsible for the execution of video and other kinds of media.

It is important to notice that the component architecture described above refers to an engine or framework implementation for PC Games, thus being a too demanding architecture in terms of memory and processing to be implemented for cell phones.

Modules Proposed for a Framework

The frameworks and engines studied in this project allowed us to identify the main modules for the definition of FMMG. The module proposal to form the framework architecture for mobile devices consists of a basic module and management modules.

The management modules' and the basic module's classes are capable of interacting with each other, depending on the Game needs, as Figure 10 shows:

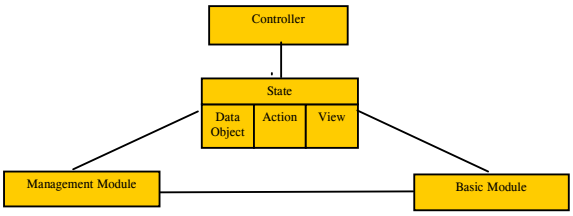


Figure 10 – Module Communication Structure

A list of the management modules is found bellow:

- LogManager – utilitary class that helps depuration/debug processes of a Game;
- ResourceManager – manages the resources (images, strings, sounds, etc) that may be used by a Game;
- SoundManager – manages the music and sound effects of a Game;
- FileManager – manages the files used by a Game;
- FontManager – manages the sources available to a Game;
- NetManager – manages network communication.

The classes listed bellow belong to the basic module of the framework. They sustain the management modules of FMMG components and can be used by developers in the construction of a Game, encapsuling API's available in the framework operational environment.

- BitMap – Image class implementation for a Windows bitmap file;
- Camera – runs down a scenery (vertically and horizontally) and records/captures all scenes reached by its "lense" and visual grasp;
- Canvas – responsible for the actions over the physical screen of the cellphone;
- Circle – creates circles;
- Film – any scene sequence made in a recording period of a camera;
- Font – represents chosen fonts;
- Image – a basic class that represents an image;
- Label – textual representation of information in the Game;
- Layer – data structure that represents a layer in a scene. It is used so that the object order can be evaluated (it separates what is in the front from what is at back)
- Projector – projects the camera-generated film on screen;
- Rectangle – creates rectangles;
- Scene – data structure that represents a Game scene. A scene corresponds to the screen size and is made of various layers;
- Scenery – data structure that represents the scenery of the Game. A scenery is made of several scenes;
- Shape – basic class that represents a visual object of the Game;
- Tile – manages a tile in an image. It represents an image frame inside an image archive containing various frames (used in the composition of maps and in animations);
- Window – creates a window on the cell phone screen. The cell phone screen can be divided into several windows to facilitate the development of a Game where too much information must be displayed, but not refreshed all at once;
- Timertask – abstract class that must be implemented for use with the Timer class. TimerTask should be implemented to schedule an execution.
- Timer – schedules a task for execution at a certain time, without framework control;
- TiledImage – builds a repetitive-pattern image. For instance, a single color image of size 200x10 can be easily built from a 10x10-sized image repeated 20 times horizontally. Vertical repetition is also possible;
- TileContainer – stores the various tiles forming a larger image (or a screen.), also grouping images and making the developer's work easier;
- System – makes the framework functionalities available for the Game. Through this class the developer can access source, network and file managers, log, and key checking functions;
- String – represents a text string;
- Stack – represents a stack where information can be stored;
- Session – this class is supplied by the framework, and it represents the Game session, where information is either stored or accessed at any stage of the Game;
- Queue – abstract class that represents the functionalities a line must have;
- Poolingtask – implements TimerTask and is used to call Controller run() methods, respecting the frame-per-second information specified to the framework. This class represents the loop of the Game;
- Pool – represents an Object pool. This class is used for temporary object storage;
- Object – basic class for all framework objects. All classes must extend from Object;
- List – represents a chained list;
- File – represents a file in the cell phone file system;
- Device – supplies both information and access to cellphone devices;
- CircleQueue – implements the Queue interface so that a circular line is generated;
- Context – an abstract class that represents the context of the environment where the framework is implemented.

Communication Network Structure proposed for FMMG

Network communication is a solution based on the Client/Server model for real-time games.

The Client/Server paradigm in a Multiplayer Game implies the establishment of direct one-way communication (only from client to server, or only from server to client), with no direct message exchange between clients. The server is thus responsible for the transmission of the necessary information to clients.

The advantages of the Client/Server model are:

- All information will pass through the server, so that the latter can execute an additional process before retransmiting it;
- The server may be in charge of the filter of selection of messages sent to clients, producing message traffic economy;
- Control is centered at the server, which simplifies the coordination of activities between clients;
- Low connectivity level – clients have to communicate exclusively with the server, and consequently there must be only one connection per machine, with simpler communication protocols;
- A client address directory may be kept at the server, and the client only has to know the server address;
- Software version control is simplified, since the main copy is kept at the server, being available for access to all clients.

The disadvantages of the Client/Server model are:

- There may be delays when trasmiting information, since

all information must be first sent to the server and then replicated to clients;
- System scalability depends on the server's capacity of precessing and retransmiting information;
- It does not deal well with geographic dispersion, since the geographicly distant client may not have communication options other than one specific server;
- If the server fails, the whole system becomes nonfunctional.

Components are meant to make the development of Multiplayer Games easier. The network communication module uses the object/property/event paradigm. Each property change is acknowledged as an event. Each and every player must be able to receive the updates made by the other players, so that all of the multiple players can execute and take part in the same Game. To attain this situation, the same properties must be shared by the player who modifies them as well as by the ones who receive them.

Internally, events make updates in the shared properties, which are then sent to any player. It is also possible to specify actions to be executed in response to an event, limiting data transfer only in the shared properties, and consequently reducing bandwidth use.

Summing up, the network communication module manages the connection, the login/logout notification, and the data sharing and processing of the Game.

The server must supply the following capacities:

- A simple sinchonization point for data consistency;
- Preprocessing of Game information;
- Message filter and techniques of bandwidth use reduction at the server.

The component uses two server layers attempting to increase scalability and also to allow the simultaneous processing of related tasks, either in client administration or in the data management. The two servers that make this possible are:

- Server Manager: manages the initial point of contact in the connection of a client in a Game. The system can have only one single server manager.
- Simulation Manager: stores and mediates shared data access in the Game. The system can have several simulation managers.

Both server manager and simulation manager can be located either in the same or in different machines.

Since these servers are separate programs, they can be executed simultaneously, making simulation administration hence transparent to the client. That is, any Game being executed will not be interrupted upon the connection of a new client to the server manager – something that would not be possible if the two servers constituted a single program.

With the Client/Server model, as long as all messages are routed by a central process, system scalability depends on two factors:

- The ability to adjust the power of the central process hosting machine, and
- The ability to share the work amount between machines.

In a simple server configuration, a single machine manages the work amount both at the server manager and at the simulation server. To increase the processing power of the server, we can either increase the machine speed or use multiple machines to host the server manager and the simulation managers.

In a multiple server configuration, the server manager's and the simulation managers' work amount is distributed among several machines. The processing power can be extended with the speed increase of the machines.

The UDP protocol is used for intercommunication within the network; therefore, some of the unnecessary overheads imposed by the TCP protocol are annulled.

A consequence of using the TCP protocol is that acknowledgements are sent to each package received by the client. Each acknowledgement is sent in its own package and cannot be packed with other data. If no acknowledgement is received for a sent package, the latter is sent again, as it is understood the package must have been lost, but not replaced. The high level of packages distributed with the use of the TCP protocol may affect performance in a shared simulation.

The component reduces package overhead, packing the acknowledgement with other data and resending lost packages only when the update has not been replaced by a more recent one. To attain this, the component overwrites the UDP protocol, which is an unsafe service, and incorporates its own acknowledgement functionality to it.

In the component, if no other data is being sent, acknowledgements are delayed one second before they are sent. Such delay reduces package traffic in two different ways. Firstly, it allows the packing of more acknowledgements, provided that more updates are received. Secondly, it increases the chances of annulling the transmission of old information in lost packages. In addition, acknowledgements are packed with other data in order to reduce the average number of sent packages. Also, the component allows the specification both of the shared-property update frequency and of the simulation server update rates, thus reducing bandwidth use.

The shared-property update frequency indicates the frequency with which updates are lined for transmission. Thus, if a client makes 20 updates, it is possible that only 10 of them will be lined, depending on the update frequency and on the type of Game.

The update rate of a connection indicates the frequency with which packages will be sent to the simulation server and vice-versa. This may reduce the number of sent packages, since only the most recent update of a given property will be sent.

Considering that clock systems used by clients in a Mobile Multiplayer Game are hard to synchronize, the component internally determines a global simulation time referred to each client. The global simulation time derives from the estimated latency time, that is, the time span between the moment the client makes a change and the moment this change reaches the simulation server.

The Server Manager manages the initial point of contact in the connection of a client to a Game. Based on the information given by component configuration files, the manager authorizes the connection of the client to the Game, and sends it the simulation server's location. Once the client is authorized, the server manager directs the client to an appropriate simulation server.

The simulation server is also responsible for licensing and for the system security, with no impairment of the system. Finally, the server manager includes a graphic interface that shows the simulation servers and the reference points they are responsible for, as well as the clients connected to each of them.

The primary function of simulation servers is to store and mediate Game property access, keeping the most recent copy of Game data. It also allows the checking of a property's current values, the deletion of objects, and the disconnection of clients.

Servers are especially designed to forward game data packages via sockets, thus working as a gateway, with clients receiving data packages.

CONCLUSIONS

We introduced in this paper the FMMG, detailing both its framework structure and its modules, as means to show how they are integrated, consequently resulting in a powerful and efficient tool for the development of Mobile Multiplayer Games.

The objective of the FMMG is to provide game developers with a set of classes for game development, where the FMMG applications will be application-logic oriented; that is, the logic of

a certain domain might be ported to other platforms and seen through various means, hence assuring greater reusability.
In the future, another objective of the FMMG will be to make it possible to port Games to a J2ME (Java 2 Platform, Micro Edition) environment, with just an adaptation of the framework implementation in the operational environment.

REFERENCES

Brooks, F. P. 1987 – "No Silver Bullet: Essence and Accidents of Software Engineering," Computer Magazine, 20(4), pp. 10-19.

Bethke, E. 2003 – "Game Development and Production," Wordware Publishing.

Battaiola, A. L. 2000– "Jogos por Computador – Histórico Relevância Tecnológica e Mercadológica, Tendências e Técnicas de Implementação," Proceedings of XIX Jornada de Atualização em Informática (JAI) – V. 2, pp. 83-122, SBC.

LaMothe, A., Ratcliff, J., Seminatore, M., Tyler, D. 1994- "Tricks of the Game Programming Gurus," Sams Publishing.

Araujo, R. B., Battaiola, A. L. 1998 - "Jogos 3D Interativos Multiusuários na Internet: Estado Atual, Perspectivas e Desafios," Internal Report of DC/UFSCar.

Battaiola, A. L., Domingues, R. G., Feijo, B., Scwareman, D., Clua, E. W. G., Kosovitz, L. E., Dreux, M., Pessoa, C. A., Ramalho, G. 2001 - "Desenvolvimento de Jogos em Computadores e Celulares," RITA, V. 3, N. 2.

Deriggi Jr., F. V. 1998 - "Suporte de Comunicação para Sistemas Multiusuários de Realidade Virtual," Masters Degree Dissertation, UFSCar - Brazil.

Van Dam, A. 1993 - "VR as a Forcing Function: Software Implications of new Paradigm," IEEE´93 Symposium on Research Fronties in Virtual Reality, San Jose, CA.

Sementille, A. C. 1999- "A Utilização da Arquitetura CORBA na Construção de Ambientes Virtuais Distribuídos," Doctoral Thesis – Physics Institute of São Carlos - USP, São Carlos.

Azevedo, E. 2005 – "Desenvolvimento de Jogos 3D e Aplicações em Realidade Virtual," Elsevier Publishing.

Pessoa, C. A. C. 2001 – "wGEM: Um Framework de Desenvolvimento de Jogos para Dispositivos Móveis," Masters Degree Dissertation, UFPE.

Krasner, E. K., Pope, S.T. 1988– "A Cookbook for using the Model-View- Controller User Interface Paradigm in Smalltalk-80," Journal of Object Oriented Programming, p. 26-49.

BIOGRAPHY

Mario Massakuni Kubo is currently developing his Electrical Engineering Doctorate research in the field of Computing and Digital Systems Engineering at the University of Sao Paulo. Master of Computing Sciences (2000) at the Universidade Federal de Sao Carlos, Specialist both in Projects Development and Management and in Information Systems (1997), and graduate Data Processing Technologist (1996) at the Universidade de Lins. At present working as a project manager at software development companies and also as a university professor.

Dr Romero Tori is Associate Professor at University of Sao Paulo (USP), a Full Professor at SENAC University, and general manager of Interlab (Interactive Technologies Laboratory) at USP. Romero was one of the pioneers in researching Computer Graphics in Brazil, and he's been teaching Computer Graphics in computer engineering undergraduate and graduate programs since 1984. Romero's approach for teaching computer graphics is based on game technology and in customized software tools, including a Java 3D game engine, called "enJine", and an interactive learning tool, called "Interlab 3D", both of them developed at his research lab. Recently Romero was the General Chair of VII Symposium on Virtual Reality and the Program Chair of WJogos (Brazilian Game Technology Symposium).

Investigation into Mobile Development Tools and Technology for Mobile Games and Application

Aly Uweso Abubakar Salim and Dr Quasim .H. Mehdi
Games Simulation and Artificial Intelligence Centre (GSAI)
School of Computing.
University of Wolverhampton.
A.Salim@wlv.ac.uk

KeyWords :

Adhoc mobile grid, mobile grid, mobile devices, protocols, communication,

Abstract

Mobile devices have come a long way with the advancements in terms of processors, memory etc. This has brought about flexibility for development of platforms and different applications far more superior to older ones used and has prompted research into better methods of deployment and use of mobile device capabilities. This paper looks at different technological advancements in progress and also proposes a plan for future work evaluates current and future developments..

1. Introduction

As the mobile devices continue to become more sophisticated with time and capital investments being poured into the industry to enable production of better devices there is more room for play in terms of production of more sophisticated software (games and multimedia applications) geared towards mobile devices [2]. Mobile games and applications have also come along way from the days of black and white snake ages. There has been a significant improvement in terms of graphics and playability all within the limitations associated with mobile devices in terms of hardware capabilities and communication. Games as we know them have three different parts physics, graphics and game AI. When it comes to mobile device games then the use of these tools has to be limited due to the inherent limitations faced by developers unlike their PC and purpose built game stations like the Playstation, Xbox, etc. This has to some extent been looked at with the development of tools that could help design more appealing games like the Gapi draw [11] and Opentrek [11] platforms for graphics and networking. The gapi draw in particular does go a long way in improving graphics and thus overall look of the games. As for the physics capability there are dots of it but not to the extent that a player would like and lets face it as the generations of mobile users change the more demanding they become when it comes to the overall quality of the games they purchase for their mobile devices be it FSP's, strategy, sports or even the puzzle games. A survey done by Nokia has shown figures for people that love playing mobile games while waiting for their means of transport is getting higher [3]. This type of games requires different degree of capabilities in terms of physics, graphics and game AI engines. The PCs or PS2 and Xbox games have different engines that deal with the complex nature of each and thus like pieces of a puzzle once fitted together the game developed is a sight for sore eyes in most cases. The game AI has been developed in such away that could literally increase the amount of AI that is currently available on mobile games. The rule based development environments require powerful computers with fast processors but this may not necessarily be untrue [1]. This requirement may be needed only during the development stages after which they can be deployed to less powerful devices. Once the knowledge base is developed, it can be extracted and combined with an MEE (Mimosa Execution Environment) a light weight interpreter for the mimosa language targeted at a particular subclass of mobile devices [1]. There are some MEE's constructed for different languages for devices running different operating systems including Symbian, WinCE etc [1].

The development of game engines for mobile phones require more critical evaluation of the strengths and weaknesses of mobile devices and this should be taken into account when developing these engines [2].

2. 2.Development Platforms

There are different platforms for game and application development available for mobile devices including J2ME, Windows Mobile, Brew, and Jamdat. Windows mobile is a development platform for smart phones and PDA's. It addresses different aspects with regards to developing applications and games for mobile devices (10). There are two instances for windows mobile SDK one for smartphones and the other for PDA's [10]. The use of Windows CE is restricted to PDA's and deals with the development platform that supports windows CE

(.NET CF platform). This is due to the fact that .NET CE still needs about 2MB space for it to be installed and run on mobile devices which is mostly seen in smart phones and PDA's [12]. It is designed for windows based CE compliant PDA's and smartphones [14]. As compared to J2ME it is not open-source which does prove to be more cumbersome when it is used in research purposes [13].

J2ME is a collection of configurations, profiles and optional packages fitted like a puzzle into a compliant development platform for applications on mobile devices. It is branched into two paths CDC (connected device configuration) and CLDC (Connected limited device configuration) profiles where the former caters for more hi-end tiered mobile devices while the latter supports more average mobile devices with more stringent memory and processor capability. Thus CDC is a superset of CLDC. J2ME consists of three layers a virtual machine, configuration and profile. The java virtual machine is placed at the lowest level and interacts with the operating system available it exists in two types that are already mentioned above namely CDC and CLDC [12]. There are different components are used for mobile wireless devices as can be seen in Fig 1 [4].

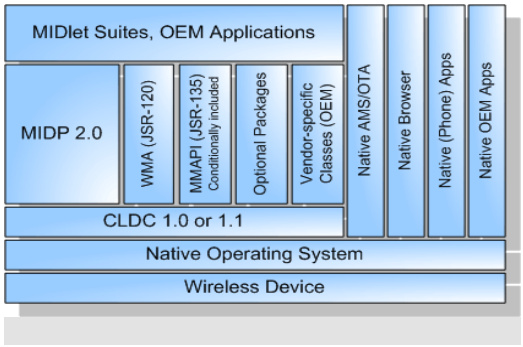


Fig 1: Mobile phone software components [4]

Thus the CLDC incorporates MIDP (Mobile Information Device Profile) profile. MIDP has so far been reviewed and the latest version out is the MIDP 2.0 with game API which is an asset as far as mobile game programming is concerned and has refined a lot of the problems associated with MIDP 1.0 providing more features for the mobile game programmer to work with including collision detection, sprites, tiled backgrounds, layers and layer management [13]. This addresses some of the challenges which have been experienced with MIDP 1.0, for example, Collision detection in MIDP 1.0 was not addressed but it is included in MIDP 2.0.

A TimerTask had to be run and used to check every half a second whether the ships had collided or not. If detected then the timer would stop as shown in listing below:

```
Public class CheckWinTimerTask extends
        TimerTask {
Public void run() {
  If (( ship.friend.x < ship.foerightx)&&
     ( ship.friend.x > ship.foeleftx)&&
     ( ship.friend.y < ship.foebottomy)&&
     ( ship.friend.y > ship.foetopy)){
    System.out.println("CAUGHT");
    Ship.caught = true;
   }
  If ( ship.caught == true) {
        fspTimer.cancel();
        foe.TimerTask.cancel ();
        checkWinTimerTask.cancel ();
   }
  }
}
```

The collision detection in MIDP 2.0 can be performed a lot easier with less coding as shown below:

```
Private void checkShips () {
   If ( ship.collidesWith (ship2, true)) {
      Ship.undo ();
      Stop ();
    }
}
```

It is free to download and use in its entirety and has a very wide support base though it has its own limitations [4].

Brew is a Qualcomm ingenuity targeted at its CDMA (code division multiple access) technology. It supports C, C++, Java and XML. It however needs brew based chipsets for it to work thus restricting it to brew chipset phones only [15]. Its applications are compiled to machine code which enhances its speed. As far as the data storage is concerned brew allows bit level manipulation which increases its efficiency in this area. Brew does also have some costs which a developer incurs before gaining some much needed tools for development [15].

There are different graphics platforms being used to create graphics for mobile games and multimedia applications that are required by different hardware configurations of these devices [11]. Also there are different types of graphic APIs for handheld graphics development including Java based graphics APIs, Frame-buffer APIs and Graphics hardware APIs. They all seem to have influenced the developers to think about moulding their products towards the devices they are trying to create them for thus creating more gruelling work [11]. The development platform for graphics extends to different mobile devices and operates on palm, symbian as well as windows mobile [11]. The GAPI platform is a mixture of frame buffer API and Graphics API. It is a platform that can be used to develop high

performance graphics and can be used on a normal pc and the product implemented onto the mobile device without downscaling and at the same time the developers need not worry about device specificities and can concentrate solely on their products logic. It has so far been used in quite a few research projects including Tilt and Feel which explores the use of mobile devices with a tilt sensor and a vibrotactile transducer [18]. It has also been used in education where it has been used for teaching students software development on mobile devices along with the OpenTrek platform for networks. Thus this shows that it is an enabler platform that can be used to pioneer new mobile device applications and concepts to be tested on mobile devices [11].

3. Grid Computing and Mobile Devices

Grid computing is rooted to the high performance computing area. There have been three approaches so far towards providing alternatives to the massively parallel processor systems. These include Local area Networks of work stations (NOW) thus providing low cost high performance computing e.g. Beowulf systems. The second approach is the use of geographically diverse supercomputing resources via high speed networks bringing together gigaflop capable centres to form teraflop capable virtual super PCs yielding huge amounts of performance for applications that require this. The third approach is most like peer to peer file sharing where included with this file sharing is computation, bandwidth and storage [8].

There has been a lot of research now being funded in terms of research into mobile devices on formation and usage of grid services and resources. This is a pool of untapped enrichment that needs to be focused on. So far there has been progress within this area with research being able to break boundaries of enabling the mobile devices to tap into the resources available from computing grids. This has been tried and tested with much success as seen in different research work like GridLite a framework for managing and provisioning services on grid-enabled resource limited devices. It is a framework that has been tested at the HP laboratories and is used to minimise resource constraints of mobile devices using the intelligent grid infrastructure [5]. It is however pointed towards smart phones and PDAs thus cutting out the rest of the mobile phones that are around. Condor grid computing research has initiated research into integrating mobile devices into the grid. Condor is on its own is a grid architecture that runs on a set of heterogeneous computers. Each PC executes two daemons including a scheduling daemon and a starter daemon for launching new jobs. Thus helping tap into the grid infrastructure (6).

Another addition is the proxy-based clustered architecture which is seen as a way of integrating mobile devices through an interlocutor (intermediary) preferably a laptop which provides a link between the devices and the grid infrastructure as shown in Fig 2 [8].

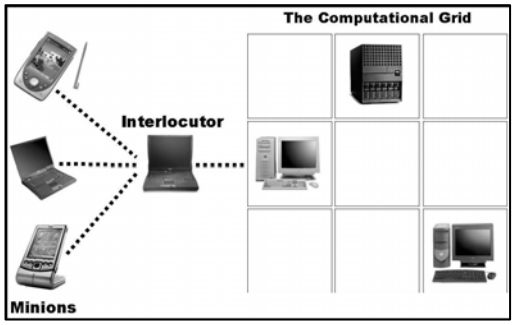


Fig 2: A Broad view of the proxy-based clustered architecture [8]

Mobile OGSI.NET is the first venture in terms of exploiting the possibilities of a native mobile grid. The use of OGSI standards from Globus in creating an infrastructure for mobiles to co-operate and share resources and at the same time is able to tap into the computer grid infrastructure has been implemented. The facts that it has been set for windows CE compliant handsets mostly pocket PCs. It presents a glimpse into the future to come. Figure 3 shows how the relationship of OGSI.NET with device hardware and software layers.

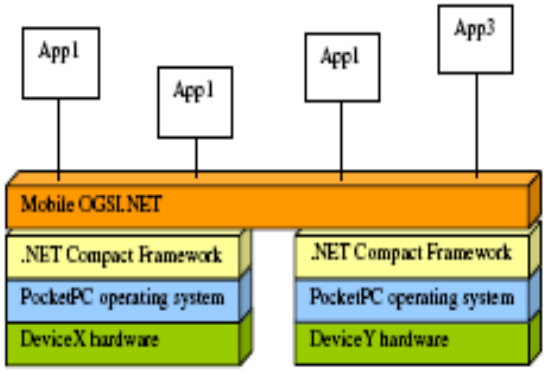


Fig 3: Mobile OGSI.NET and its relation to other device hardware/software layers [9].

This is also shown that the ability to create and assimilate a mobile grid infrastructure is not out of reach but just needs more research and refinement [9].

A lot of research on mobile devices and grid integration has been done using Globus and condor grid infrastructure. There is another infrastructure which has by far been left out called Plan 9. Plan 9 is a distributed operating system that us able to create and maintain per process distributed environments independent of the physical location of resources [16]. Thus unlike globus Plan 9 provides allows for grid creation without changes being made or middleware being created. Thus the two of them

show two different approaches to making of grids [16]. It being an operating system that is grid enabled it offers good principles in terms of authentication processes, data management processes, network management and also security principles which could aid in building a strong more reliable and secure grids in the future [16].

A lot more work has been done towards adhoc mobile grid applications specifically MoGrid a project done in brazil for peer 2 peer resource discovery on mobile grids thus the MoGrid application. This has focused more on adhoc networks aiming to provide collaborative support towards applications that run purely on adhoc networks [17].

The utilization of technology available towards formation of mobile grids may unleash a better more efficient game playing environment for mobile device owners. If harnessed then the potential for mobile games and applications becoming even more enhanced in terms of physics, graphics and AI. It would help in easing the load of processor power in terms of being able to play or run more intense games or applications that may need more processing resources than the device can offer. Thus the use of other devices within its reach comes in handy.

4. Communication in Mobile devices

There are a quite a few established communication methods when it comes to mobile devices. The client/server network models seem to be favoured over others. This can be seen by the use of a central point of resource coordination which still poses restrictive and centralised approach on wireless network infrastructure [17]. Thus client server models serve as a major basic network communication model used in construction of wireless networks as far as mobile grids are concerned. Another model that has been used so far is the peer 2 peer model that provides flexibility, less restrictive and decentralised approach towards mobile networks. This can be seen in MOGRID project that pioneers the use of a peer 2 peer discovery protocol layer that allows the distribution of grid tasks in decentralised and dependant on different factors among mobile devices [17]. This project is still in development and further tests are being conducted with regards to its capabilities.

The use of communication standards however provides another challenge as different devices allow different types of wireless communication for example WLAN, Bluetooth, GPRS, WAP, 3G etc. There are some studies still being undertaken as to sourcing out the best possible. Till this moment though each type has its own strengths and can be used according to different situations arising.

5. Future Proposal

Thus there has been a lot of development on mobile grids but along the principles of condor and Globus grid software which has so far been quite successful. However there are other avenues that can be exploited as far as creating mobile grids is concerned which may prove a lot more beneficial too in terms of security, data management, resource management etc. this has led to the idea of investigating the workings of plan 9 and inferno. Plan 9 being a distributed operating system that is grid enabled which may give key contributions toward forming tighter secure networks over mobile grids and reducing the risks involved in climbing onto a grid and letting resources from ones mobile device be utilised. The proposed idea will be investigated with an aim of developing an adhoc independent mobile grid protocol that can be used for different resource intensive tasks. The use of different mobile devices to make up the grid is the main focus for this project. In this work, we aim to explore the use of plan 9 as a grid enabled platform. Experiment with its architecture, principles, process handling, and network communication capabilities etc., will allow for the further development of an adhoc grid enabled OS that could be used as a grid protocol for mobile devices modelled under the Plan 9 and inferno architecture and processes.

So first the exploration of Plan 9 will commence where appropriate scenarios are drafted in to test its capabilities and give an insight to how it would support the creation of adhoc mobile grids.

There will be more research into inferno which draws from plan 9 and is supposed to be a more compact platform and has similar qualities to plan 9. This will help in assessing its suitability as a portable operating system or application on mobile devices.

Networking will be looked at in different aspects and with regards to different scenarios thus the use of blue tooth, 3G, WLAN, WIFI among others will be debated by assessing different situations including areas with little or no wireless capability, areas of confined space and on open ground etc. This will therefore affect the choices of which protocols will be included for communication between the devices.

Another aspect will be addressed is the security principles that are offered by plan 9 and assess their suitability to mobile device grid security. Thus a suitable test bed will be used to assess it and the results will therefore shape the direction in which the security aspect of the adhoc mobile grid will be decided.

6. Conclusion

The research so far has proved an invigorating experience and with new mobile devices being unveiled every quarter there is room for more advancement in terms of mobile applications and

games. This paper goes to an extent to show the research being done thus far in the field. As for the development of mobile grids there are still more improvements being developed due to the increased capabilities being seen on new devices.

The use grid computing for mobile devices has been a good turning point as far as use of these supercomputer highly intensive resources is concerned. This in time will bring about the enhancement of games and multimedia applications thus bridging the QOS gap between mobile devices applications and desktop computers. A new dawn seems to have come of age since the breakthrough into grid computing for mobile devices was realised. The realisation of how application graphics, features, AI components especially in games, physics capabilities can be improved. Thus with the future proposal that has been briefly outlined in this paper there is hope of even further accomplishments in the mobile grid computing field with benefits being realised for mobile multiplayer games and multimedia real time applications.

References

[1]. Hall L, Gordon A, James R, Newall L. 2004 "A lightweight Rule-Based AI engine for Mobile Games, *Proceedings of the 2004 ACM SIGCHI International Conference on Advances in computer entertainment technolog"*, vol. 74. pp 284 – 289.

[2]. Bancroft M., Cant R., Langeniepen C., Al-Dabass D,. 2005. "A game engine for Mobile phones. *Proceedings of international conference for computer games*." Pp. 89 – 95.

[3]. Ritter H, Voigt T, Tian M, Schiller J. 2003 "Experiences using a dual wireless technology infrastructure to support ad-hoc multiplayer game. *Proceedings of the 2nd workshop on Network and system support for games*." Pp 101-105.

[4]. Ortiz E. 2004. "A Survey of J2ME Today."http://developers.sun.com/techtopics/mobility/getstart/articles/survey/(Accessed 10th March 2006).

[5]. Kumar R, Song X. 2005. "GridLite: A Framework for Managing and Provisioning Services on Grid-Enabled Resource Limited Devices".*http://www.hpl.hp.com/techreports/2005/HPL-2005-146.pdf* (Accessed 5th February 2006).

[6]. González-Castaño F, Vales-Alonso J, Livny M, Costa-Montenegro E, Anido-Rifón L. 2002. "Condor grid computing from mobile handheld devices. *ACM SIGMOBILE Mobile Computing and Communications Review*."vol. 6. pp 18-27.

[7]. Millard, D., Woukeu, A., Tao, F. B. and Davis, H. 2005." Experiences with Writing Grid Clients for Mobile devices. In *Proceedings of 1st International ELeGI Conference on Advanced Technology for Enhanced Learning"*.

[8]. Phan T, Huang L, Dulan C. 2002. "Challenge: Integrating mobile wireless devices into the computational grid. *Proceedings of the 8th annual international conference on Mobile computing and networking*."pp 271 – 278.

[9]. Chu C. D., Humphrey M. 2004. "Mobile OGSI.NET: grid computing on mobile devices.*Fifth IEEE/ACM International Workshop on Grid Computing.*"pp. 182 – 191.

[10].Microsoftsite http://www.microsoft.com/windowsmobile/about/default.mspx. (Accessed 10 march 2006).

[11]. Sanneblad J., Holmquist L.2004."The GapiDraw platform: high-performance cross-platform graphics on mobile devices *Proceedings of the 3rd international conference on Mobile and ubiquitous multimedia MUM."vol.83. pp.47 – 53.*

[12]. Janecek A., Hlavacs H. 2005. "Mobile and wireless games: Programming interactive real-time games over WLAN for pocket PCs with J2ME and .NET CF. *Proceedings of 4th ACM SIGCOMM workshop on Network and system support for games NetGames*."pp. 1 – 8.

[13]. Williams C., Burge M. 2004."Special session on mobile computing #2: MIDP 2.0 changing the face of J2ME gaming. *Proceedings of the 42nd annual Southeast regional conference*."pp. 37 – 41.

[14]. O'Hara R., 1997. "Microsoft Windows CE: A new handheld computingplatform. *Proceedings of the 1997 ACM symposium on Applied computing.*" Pp. 295 – 296.

[15]. Coulton P., Rashid O., Edwards R, Thompson R." Creating Entertainment Applications for Cellular Phones. *Computers in Entertainment."* vol. 3. issue 3. pp. 3 – 3.

[16]. Pike R., Presotto D., Dorward S., Flandrena B., Thompson K., Trickey H., Winterbottom P. 1995. "Plan 9 from Bell Labs. Computing Systems." vol. 8. pp. 221 – 254.

[17]. Lima L., Gomes A., Ziviani A., Endler M., Soares L., Schulze B.2005. "Peer-to-Peer Resource Discovery in Mobile Grids. *Proceedings of the 3rd international workshop on Middleware for grid computing MGC*."vol. 117. pp. 1 – 6.

[18] Oackley, I., Angesleva, J., Hughes, S., and O'Modhrain, S.: Tilt and Feel; Scrolling with Vibrotactile Display, in Proceedings of EuroHaptics 2004, Munich, Germany.

RECURSIVE INTEREST MANAGEMENT FOR ONLINE GAMES

Pawan Kumar and Qasim Mehdi

School of Computing and Information Technology
University of Wolverhampton
Wolverhampton, UK WV1 1SB
{pawan.kumar, q.h.mehdi}@wlv.ac.uk

KEYWORDS

Multi-player online games, interest management, HLA, data distribution management, multi-dimensional routing spaces

ABSTRACT

Performance and scalability in multi-player online games and distributed simulators mainly depends on the effectiveness of the deployed interest management schemes. These schemes aim at providing message-filtering mechanisms that reduces the communication overheads. However, in order to do so, they incur computational costs that are quite significant and are not suitable for scalable real time systems. In this paper, a recursive algorithm for interest management is presented that can be applied for systems that use multi-dimensional routing spaces for interest management. The algorithm's simulation shows that it is more efficient and scalable than existing approaches.

1. INTRODUCTION

Networked games allow multiple players at geographically dispersed location to share and play in a common virtual world. Typically, a player's node will contain some subset of the shared virtual world whose state is influenced and maintained by the player. In order to have a mutual consistent view of the virtual world, events or messages are exchanged between player nodes (either directly or indirectly through a server) [1]. However, an update occurring at one node is likely to have an immediate significance for only a subset of other nodes in the system. The techniques that exploit this *interest* of each node to minimise the number of messages sent are referred as *interest management* (IM) schemes.

Interest management systems have been incorporated in several large-scale distributed simulators [2,3], collaborative virtual environments [4,5,6] and multiplayer online games [7,8,9]. These have been incorporated mainly to allow systems to scale seamlessly and efficiently. The scalability in these systems is primarily related to the number of nodes that can participate and the computational complexity of the model that is being simulated (e.g. in a game it could be the number of entities the game has, etc). Without the IM system, it would entail every update or state changes at one node to be communicated to all the other nodes. This could significantly increase the bandwidth usage, message sent per second and computational requirements at processing these messages. However, incorporating IM system would try to minimise the above at the expense of computational costs for it's processing and thus affecting the real-time requirements of these systems and degrading performance. Thus, performance and scalability of these systems mainly depend on the effectiveness of the deployed IM scheme in these systems.

Several IM schemes have been adopted in which a node expresses its interest to some subset of the world. Only information that is pertinent to the node gets forwarded to the node. The filtering techniques used to achieve this can be *grid-based* [4,5,10], where the world is partitioned into an n-dimensional grid cell. Each node subscribes to some set of cells and updates are sent only between nodes whose subscriptions fall into the same grid cell. The advantage of this scheme is in its simplicity by statically partitioning the world in advance and each grid cell can be associated with a multi-cast address. However, issues of cell granularity can either result in imprecise filtering (for coarse grained) or significant overheads of leaving and joining multicast groups (for fine grained). Another filtering mechanism is based on *class based* filtering [3,11], where nodes express interest through subscription to some set of classes and send/receive updates to only those classes. This scheme is quite powerful and is used along with other filtering schemes [3]. However, in its entirety it is not sufficient for scalable systems. Further, there is *region-based* scheme [3,6,9], where simulation entities or nodes specify interest areas in the form of publisher and subscriber regions where intersection between them represents a potential communication between the entities. The regions can be specified as homogenous multi-dimensional routing space [3,9] or as auras [6]. The region-based scheme is more powerful and general as it allows each node to specify, create and modify the regions at run time. However, this expressivity does come with a significant overhead that could lead to a time complexity of $O(n^2)$. In this paper, a recursive algorithm (order O(nlogn)) for region matching is proposed where regions are defined using multi-dimensional routing space (Figure 1). Several other techniques for IM includes use of N trees [12] for massively multiplayer online games, sphere of influence [13] for distributed agent simulation and use

of message oriented middleware technologies [14] for networked games. However, their discussion is beyond the scope of the paper.

The remainder of this paper is organised as follows. Section 2 provides a background of existing approaches to region matching for multi-dimensional routing spaces. In section 3, the recursive algorithm for region matching is presented along with its complexity analysis. In section 4, simulation results and performance evaluation is provided. Finally, section 5 details conclusion and future work.

2. BACKGROUND AND RELATED WORKS

Much of the works of interest management using multi-dimensional routing spaces has been undertaken in the context of High Level Architecture (HLA) [3,16] and similar concepts have been applied in multiplayer online games middleware [9]. HLA is a standard interface specification that provides a general infrastructure and services for distributed simulation. Two of its services namely, *Declaration management* (DM) and *Data distribution management* (DDM) offers IM facilities. While the DM service provides IM using *class-based* approaches, the DDM service provides IM based on *region matching* where publisher and subscriber regions are specified using routing space information. A *routing space* is defined as a collection of dimensions that are used to define regions. Typically, a region comprises of set of *extents* where each extent has a *bounded range* defined along each dimension of the routing space as shown in (Figure 1). For IM, node's specifies object attributes or interactions that it wishes to publish or subscribe (using DM) and associate regions (extents) with those subscriptions (using DDM). When publisher regions overlap with subscriber regions, connectivity is established between the two nodes and finally the information is transmitted. Thus for successful implementation of IM, it is required that region matching and connectivity be established as efficiently as possible. Depending on the context of usage, several algorithms have been established for DDM. These are discussed next.

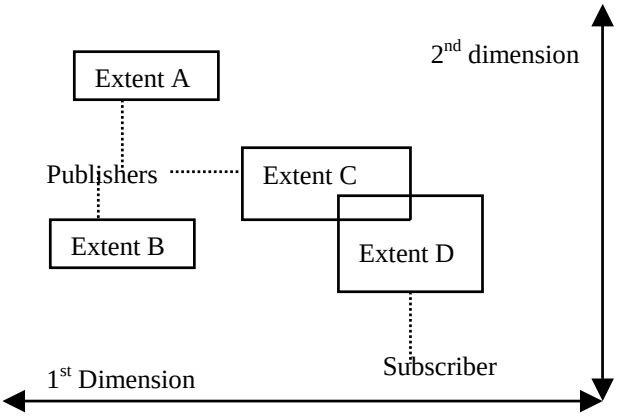


Figure 1: Extents in 2-dimensional routing space

2.1. Brute Force Approach

Brute force approach for region matching simply checks each of the publisher extents with each of the subscriber extents for an overlap. This results in a complexity of $O(n^2)$. The advantage of this approach is in its simplicity and performs well in situations when most of the publisher extents are overlapping with the subscriber extents, as probability of getting a matching pair early is high. However, this algorithm does not scale well except in situations where high number of regions overlap.

2.2. Grid Based Approaches

Grid based approaches can be utilised for DDM implementation as it completely eliminates the complexity associated with brute force approach. In this, routing spaces are partitioned into grid of cells and extents are mapped to some subset of the cells. Whenever, publisher extents and subscriber extents overlap with the same grid cell, they are assumed to be overlapping with each other. A prior application specific knowledge of the world being simulated is required for efficient realization of this approach. One such implementation partition the world into grid cells and statically assigns multicast address for each grid cell [10]. This approach is simple and scalable than the brute force approach. However, issues of cell granularity, as discussed in [15], can significantly affect the performance and resource usage of the IM system. A large cell size could lead to imprecise filtering, as extents may not be overlapping even though they are mapped to same grid cell. This would not only deliver additional data across the wire but also requires additional processing at the receiver to filter irrelevant data. On the other hand, a finer grid cell would require additional resources for maintaining data structures for each cell. Further, if multi-casting is used, then this could lead to thrashing in network layer where extents are modified frequently and are joining and leaving several multicast groups. In addition, there can be practical limitation of available multi-cast addresses [15] that could significantly affect the scalability of the systems. Thus, application specific knowledge, network bandwidth, resource availability are the key metrics for deciding an optimal cell size. In [17], multi resolution grids were used that were statically defined and were allocated multicast addresses. One section of the world uses coarse resolution while another section uses a fine resolution.

In addition to static approaches, more recent works uses dynamic grids [18] and hybrid approaches [19] as an extension to grid based approaches. In dynamic grids, multicast addresses are dynamically allocated to the cells that have at least one publishing extent and one subscribing extent. This results in efficient utilization of multi-cast address. Hybrid approach, on the other hand, combines brute force with grids. In this, world is partitioned into grid cells and then for all extents in each grid cell, brute force algorithm is used for region matching. This approach is more scalable

than a simple brute force approach and produces less irrelevant messages than grid based approaches. However, both still suffer from optimal grid size and application specific knowledge for their efficient implementation and have similar drawbacks as with static grids.

2.3. Spatial Partitioning Based Approaches

Like grid-based approaches, [16] suggested the use of spatial partitioning or hierarchical approaches for decomposing the world. In these, world can be partitioned using persistent spatial data structures such as quad-trees or oct-trees. These data structures provide efficient queries for searching. Depending upon usage, a node retrieved from a tree can have a multi-cast address associated per node or can have multiple extents that uses brute force for region matching. Further, extents can be stored at leaf nodes or at middle nodes and tree's depth can be statically fixed a priori or be allowed to grow dynamically. In addition to these issues, there are significant costs associated for storage, maintenance and balancing of these complex trees. Further, proper partitioning requires application specific knowledge.

2.4. Sort Based Approach

Recent work of [20] proposed a sort-based approach for region matching algorithm. In this, for each dimension in a routing space, a list of endpoints (representing coordinates of extents in that dimension) is created, sorted and examined in ascending order to obtain the overlap information in that dimension. Extents overlap if an overlap is found in all the dimensions. However, despite making improvements to their original algorithm and incorporating additional data structure for optimisation and intermediate storage of overlap information, the overall complexity still remained quadratic [20] and is therefore not scalable.

3. RECURSIVE ALGORTIHM FOR REGION MATCHING

All above discussed approaches have weaknesses and are not suitable for interest management in systems that demand performance and scalability. Our research aims at finding new approaches for interest management that uses multi-dimensional routing spaces and demand performance and scalability. Here we propose an efficient and scalable recursive algorithm for region matching that can be applied in these systems.

3.1. Problem Definition

Given

$$Sp = Space,$$
$$D(Sp) = Set\ of\ Dimensions\ for\ S$$

A region ***R*** comprises of set of extents and can be associated as a publisher or a subscriber. Therefore problem of region matching requires finding overlapping publisher and subscriber extents. Let

$$P = Set\ of\ Publisher\ Extents$$
$$S = Set\ of\ Subscriber\ Extents$$
$$R(x) = Set\ of\ Ranges\ for\ Extent\ x$$

Then we have

$$\forall P_i \in P : \exists R(P_i) \ \wedge \ \left|R(P_i)\right| = \left|D(Sp)\right|$$
$$\forall S_i \in S : \exists R(S_i) \ \wedge \ \left|R(S_i)\right| = \left|D(Sp)\right|$$

The problem here is to identify overlapping publisher and subscriber extents to establish connectivity between publishers and subscribers. The extents overlap if

$$\exists P_i \in P, \exists S_i \in S : Overlap(R(P_i), R(S_i))$$

Where $Overlap(R(x), R(y))$ is a predicate that is true if and only if

$$\forall i, j \ R_i \in R(x), R_j \in R(y) : R_i overlaps R_j \wedge i == j$$

i.e. each range of one extent overlaps with the corresponding range of the other extent.

For scalability, it is required that the above overlapping information of extents is obtained using minimal use of computational resources.

3.2. Algorithm Description

The algorithm for region matching takes the best of the existing approaches for region matching. In particular, this work extends the recent sort based approach to recursively do region matching and combines brute force approach when clusters of overlapping extents have been found. In order to understand recursive algorithm, a simple scenario is examined. Figure 1 shows four extents in a 2D routing space. Extents A, B and C belong to region that acts as publisher whereas extent D belongs to region that acts as subscriber. The first step in the algorithm identifies grouping of overlapping extents in one of the dimensions of the routing space. Figure 2 depicts this scenario. Here all the extents are projected to one of the dimensions. As seen, two groups can be obtained, one containing extents A and B and another containing extents C and D.

To algorithmically achieve this grouping information, a mechanism is required for evaluating the lower bound and upper bound points of each of the extents in that dimension. For this, a list of all endpoints in that dimension is *created* and *sorted* from lowest to highest

value. Figure 3 shows the list before and after sorting of Figure 1 extents in 1st dimension.

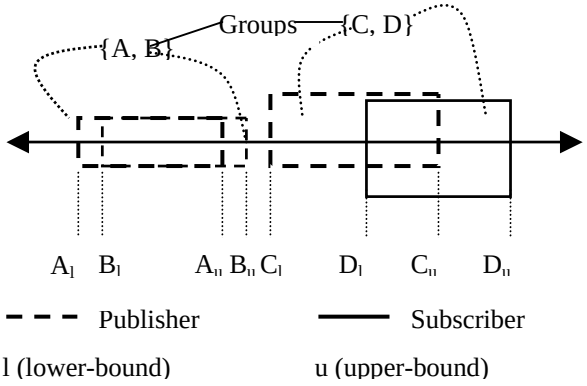


Figure 2: Extents projected to one of the dimensions

List = { A_l, A_u, B_l, B_u, C_l, C_u, D_l, D_u }

(a) List before sorting

List = { A_l, B_l, A_u, B_u, C_l, D_l, C_u, D_u }

(b) List after sorting

Figure 3: List of endpoints along one of the dimensions

Once sorting has completed, the algorithm proceeds to find the group of overlapping extents in that dimension. A container *group* comprising of two sets: one for holding references of publisher extents and another for holding references of subscriber extents, is used for storing the group of extents that are currently overlapping in that dimension. The algorithm scans for groups by sweeping along the dimension and examining endpoints as depicted in pseudo code below.

```
Input: Gp (group)
Gp.P = Publisher Extent Set
Gp.S = Subscriber Extent Set
L = sorted endpoints list along a dimension
count =0;
---
For each endpoint e in the list L
{
   if (e == lower bound){
      ++count;
       extent = getExtentRef(e);
       if (extent == publisher)
             Gp.addPublisher(extent);
       else // subscriber
             Gp.addSubscriber(extent);
   }
   else {// e is upper bound
        --count;
        if (count == 0){
             groupCompleted(Gp);
        }
        useGroup(Gp);
        Clear(Gp);
   }
}
```

Figure 4: Pseudo code for finding groups of overlapping extents along one dimension

For the above scenario, the pseudo code in Figure 4 will form two groups as shown in Table 1. These would be used further for recursive matching.

Step	End point	count	Group information (Gp)	
			Gp.P	Gp.S
1	A_l	1	{A}	{}
2	B_l	2	{A, B}	{}
3	A_u	1	{A, B}	{}
4	B_u	0	Completed	
5	C_l	1	{C}	{}
6	D_l	2	{C}	{D}
7	C_u	1	{C}	{D}
8	D_u	0	Completed	

Table 1: Iterations of grouping algorithm for the scenario of Figure 1

A special consideration to sort function is required while sorting the endpoints to consistently group the extents. This is depicted in Figure 5. Here extent A's upper bound coincides with the extent B's lower bound. Clearly the order in which they get sorted is important for proper grouping. If A_u occurs before B_l then extent A and B would incorrectly be placed in separate groups whereas if B_l occurs before A_u then both A and B occupies the same group. Therefore for proper grouping, the sort function is implemented as a binary predicate as depicted in Figure 6.

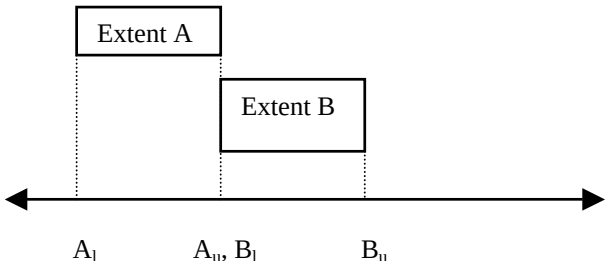


Figure 5: Extents projected along one dimension where sorting affects grouping

Input: $x \in l, y \in l$ $(l = list\ of\ endpoints)$

Output: Boolean

$$x < y\ if \begin{Bmatrix} x < y \\ (x == y) \wedge (x\ is\ lower) \wedge (y\ is\ upper) \end{Bmatrix}$$

Figure 6: Binary predicate used for sorting

The group formed in the previous step contains a reduced set of publisher extents and subscriber extents that may be overlapping along one of the dimensions. Thus a mechanism is needed to determine whether this reduced set of extents overlap or not. This is determined through recursion. Here, the groups found

in one dimension are sent along other dimensions to be examined in those dimensions. The same process is repeated for this group. A list of endpoints along that dimension is created; sorted and examined whether further sub-groups can be created or not. This process continues until there is no subdivision of the group or else the group size reaches a *threshold*, both of these used as a criterion for stopping recursion.

For our example scenario, the first group comprising of extents A and B will be sent along 2nd dimension that will result in two subgroups one containing A and the other containing B (Figure 7a). On the other hand, the second group comprising of extent C and D do not subdivide any further (Figure 7b).

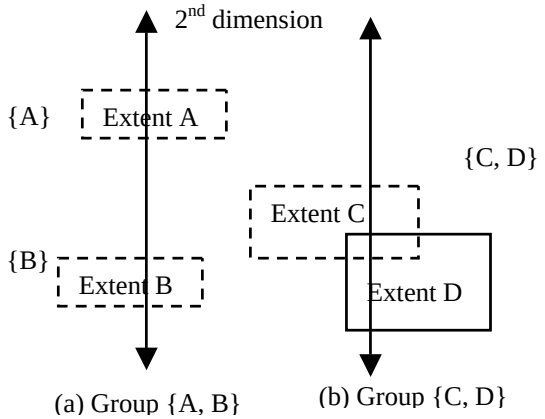


Figure 7: Groups of Figure 2 sent along second dimension

Here only three groups were created at the end of algorithm processing i.e. {A}, {B} and {C, D}. Their sizes are quite small and it is easy to identify overlapping extents. With large number of extents, the sizes of these groups can become significant and thus every recursive call will attempt to reduce the size. However, recursion has its own overheads of function calls and could lead to stack overflow. Thus for efficiency and to overcome recursion overheads, a *threshold* specifying the maximum group size is used to stop recursion and extents within the group are finally examined using brute force approach. The brute force algorithm efficiently considers each of the extents ranges for a *non-overlap* using separating axis along range's dimension (Figure 8). The steps of the complete recursive algorithm are depicted in Figure 9. Steps 1-7 of the algorithm are quite straightforward and are based on the above discussion. When a sub group is formed and completed (step 8), a recursive call is made so that it can be examined along the next dimension. All groups with size greater than *threshold* are examined in all the dimensions before recursion can be stopped. Since dimension is an abstract concept, this algorithm can be extended to any n-dimensional routing space and thus unlike other approaches, it is not limited to 2D or 3D routing spaces.

$Input: P \in Gp.P, S \in Gp.S$

$Output: boolean$

$for\ each\ \ d\ \ in\ D(Sp)$

$\quad if\ ((P_u^d < S_l^d) \vee (S_u^d < P_l^d))$

$\quad\quad retrun\ false;$

$return\ \ true;$

X_y^d : Extent ***X***'s endpoint in dimension ***d***

Figure 8: checking for non-overlap using separating axis

Input: Gp (Group of all publisher and subscriber extents)
Output: list (list of Overlapping pairs)

1. If (Gp.size < *threshold* || invalid dimension ***d***)
2. Do brute force
3. Else
4. Examine current dimension ***d*** ∈ ***D(Sp)***
5. Create list ***l*** of endpoints in ***d***
6. Sort ***l*** using binary predicate of Figure 6
7. Find sub-groups of Gp using grouping algorithm of Figure 4
8. If Gp is sub-divided, then do steps 1 to 8 for each sub-group in other dimensions until
   a) The sub-group does not get subdivided and
   b) All dimensions have been examined for the sub-group

Figure 9: Steps of recursive algorithm for region matching

3.3. Algorithm's Complexity Analysis

The algorithm discussed above uses combination of sorting, brute force approach and recursion to achieve region matching. The overall aim of this algorithm is to reduce the inherent O(n^2) complexity of matching problem for scalability and performance. In order to provide complexity analysis, we assume there are ***n*** publisher and ***n*** subscriber extents and thus the group size is ***2n***. The first step in the algorithm checks the group size with the threshold ***t*** and uses brute force approach when the group size is less than or equal to ***t***. Thus, maximum complexity of brute force approach is of order ***O(t*t)***. Since sub groups comprises of extents that are possibly overlapping, the brute force algorithm yields best performance in this case.

Further, if the group is to be sub divided, it is examined along one of the dimensions. This requires creating a list of endpoints and sorting them. The list creation is a linear time operation of order ***O(m)*** for average sized group (***m<2n***) and sorting can be achieved in order ***O(mlogm)*** using a quick sort.

For an average case where most of the extents are non overlapping, we believe that every recursive call reduces the average group size ***m*** significantly and thus the algorithm's overall time complexity is of order ***O(nlogn)***. This is evident from the experimental simulation of the algorithm as discussed in section 4.

The worst-case performance of the algorithm arises mainly in two cases. Firstly, when all the extents are overlapping and secondly, when recursion goes very deep. In the first case, the algorithm examines all dimensions and finds only one group in the end that is sent to the brute force. This process is of order ***O(3nlogn + n*n)*** or equivalently ***O(n*n)***. In the second case, the recursion gets very deep mainly because of asymmetrical sub-division of the group such that in worst case there is only one extent in one of the sub-groups and the remaining n-1 extents in the other. If a similar sub-division occurs on every recursive functional call for the larger sub-group, then there could be n function calls in total having a time complexity of order ***O(n)***. Further, each function call creates and sorts the list of endpoints that has a time complexity of order ***O(nlogn)***. Thus worst case complexity is of order ***O(n*n*logn)***. However, in practice such cases are very rare and based on the average case performance, this algorithm can be applied for region matching where there are large number of extents.

4. Algorithm Simulation and Performance Evaluation

In order to see the effectiveness of the algorithm, we simulated the algorithm in C++ using visual studio .net on windows XP running on Pentium 4 3.2 GHz processor. We implemented DDM service interface of the HLA standard as defined in the draft specification [3]. This involved creation and defining classes for a routing space, dimension, region, extent, range, etc. Further, singleton *object factory* was created that acts as one-stop shop for creation of all the regions, extents and ranges. The factory maintains set of pooled buffers for regions, extents and ranges where each of these buffers maintains *free-list* and *used-list* for efficient creation and deletion of the associated types. In addition, spaces were used as indices to retrieve the pooled buffer for regions and extents whereas spaces and spaces dimension were used as indices to retrieve range buffers. All these optimisations have been incorporated so that the region-matching algorithm can have access to the buffers and associated objects in constant time.

For evaluation, a comparison of the algorithm's performance with the brute force algorithm is made. In addition, the following performance criteria were considered for evaluation: the computation time, scalability factor i.e. total number of extents supported, variability in threshold i.e. the affects of varying threshold size on the performance of the algorithm and the size of the routing space i.e. how algorithm behaves from average case scenarios to worst case scenarios. Further, random distribution of publisher extents and subscriber extents were used throughout the experiment and average of five runs of the algorithm for each data set is used to obtain the timing information. The routing space of 3 dimensions is used throughout the experimental simulation and the sizes of extents ranges were generated randomly using the same random number generator for all the scenarios. Three scenarios were considered for performance evaluation. For each of these scenarios, the extents were varied in numbers from 50 to 5000 whereas threshold is varied from 5 to 30. Charts 1-4 below details the simulation results.

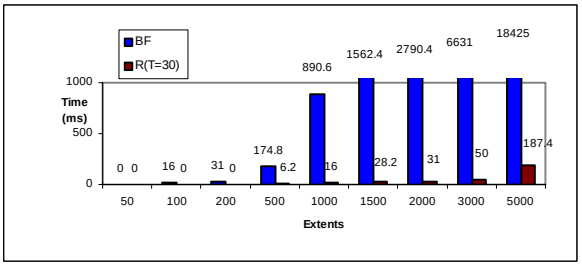


Chart 1: Performance comparison of a recursive algorithm (Threshold t=30) with brute force for 3 dimensional routing space of size 100000*100000*100000 (Average case scenario)

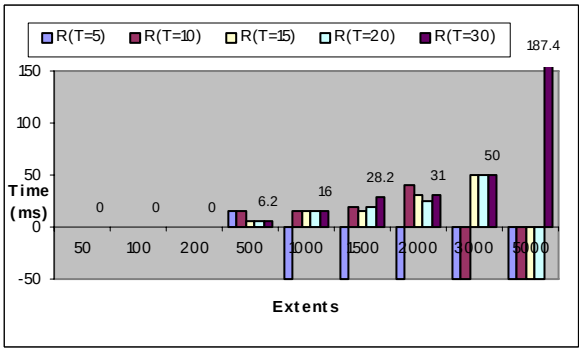


Chart 2: Recursive algorithm's simulation results for variable thresholds (T=5-30) for 3D routing space size: 100000*100000*100000 (Average case scenario)

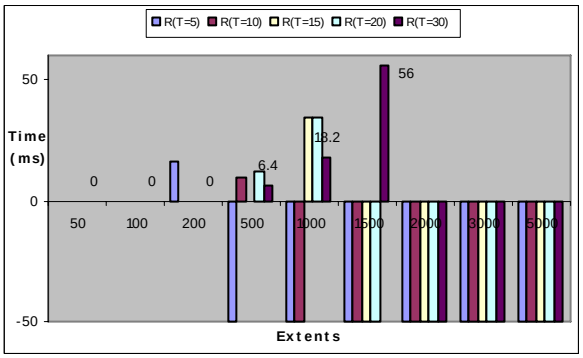


Chart 3: Recursive algorithm's simulation results for variable thresholds (T=5-30) for 3D routing space size: 10000*10000*10000 (Average to worst case scenario)

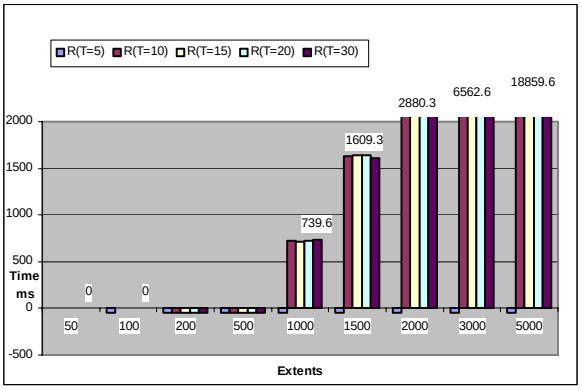


Chart 4: Recursive algorithm's simulation results for variable thresholds (T=5-30) for 3D routing space size: 1000*1000*1000 (Worst case scenario)

The first scenario for evaluation of the algorithm was based on the average case. For this, an expansive routing space of maximum 100000 units for each of its dimension was considered (chart 1 and 2). Here, extents were randomly distributed with ranges having sizes between 1 to 20 units. This forms an average case scenario as not only the extents are randomly distributed but also the size of extents is very small as compared to the world. This would form an ideal scenario for a massively multiplayer online game. Clearly from chart 1, it is evident that recursive algorithm outperforms the brute force approach. As hypothesized in the pervious section, the recursive algorithm does the region matching in logarithmic time as compared to the brute force approach that rises exponentially as the number of extents increases. Further, chart 2 details the recursive algorithm's behaviour with different choice of threshold. The ***negative*** bars in the chart reflect occurrence of stack overflows. Clearly, some interesting observation can be made form the chart. The algorithm performs smoothly up to 500 extents for any choice of threshold. However, as the extents were increased in numbers, the overheads of recursion dominate and result in stack overflows for small threshold values. This is true as more groups are being found with more number of extents and thus smaller threshold values do not break the recursion. Further two important observation made are: *when the number of extents is raised to 5000, the algorithm only succeeds with a threshold of 30 and whenever the recursion succeeds, the choice of threshold hardly have impact on the algorithms performance*. These two observations reflect key points with respect to scalability and performance of the algorithm. As the number of extents increases, the value of threshold should be increased appropriately and if the recursion succeeds, then this increase in threshold would not affect the algorithm's performance. A metric or a heuristic will be required that can achieve this seamlessly. A simple mechanism would be to catch the exception and then increase the threshold. This is yet to be incorporated in the current implementation of the algorithm.

In addition to the above scenario, we simulated the algorithm's performance for two other cases. One that reflects an average to worst-case scenario i.e. a routing space of maximum of 10000 units for each of its dimension (chart 3) and the other that reflects a worst-case scenario i.e. a routing space of maximum of 1000 units for each of its dimension (chart 4). In both these scenarios, extents were randomly distributed and the size of the their ranges varied between 1 to 20 units, as was the case with average case scenario. Because of this, the sizes of extents are quite large as compared to the size of the world and therefore there is a high probability that most of the extents would be overlapping with each other. However, such large extent sizes do not exist in reality for most of the games. Charts 3 and 4 details the simulation results for these scenarios. In chart 3, the algorithm performs smoothly for up to 1500 extents with higher end threshold values and thereafter recursion overhead dominates and result in stack overflows (*negative bars*) when extents are in between 1500 to 5000 (near worst-case). This happens mainly because of asymmetrical sub division of groups and because of possibly large numbers of extent's ranges are overlapping in each of the dimension. Further, it is observed (not in the chart) that at threshold values of 60 and 200, the algorithm succeeds and performs exceptionally well when the number of extents is 2000 and 3000 respectively. This information details some interesting facts. Comparing these results with average case scenario (chart 1), it can be seen that algorithm succeeds in both the cases when proper threshold is used. Further, with increase in the number of extents, the threshold value required increases in small steps (linearly) for scenario in chart 1 whereas for scenario in chart 2, it makes larger jumps (increases exponentially). Therefore, for proper running of the algorithm either a heuristic need to be developed that take this information into account for setting the value of the threshold or lookups could be used if the number of extents and world size are known a priori and remains fixed.

Lastly, chart 4 details the algorithm's performance for the worst-case scenario. Here it can be seen that algorithm performs well for only up to 100 extents. After that, recursion overheads dominate for extents between 200 and 500. However, some interesting results are obtained in the cases when extents are in numbers between 1000 and 5000. Here, the recursion succeeds (unlike the previous two cases) to actually do the region matching and the performance of the algorithm degrades to the order of brute force (see chart 1 and chart 4). This happens mainly because most of the extents are overlapping in this small world size and very large groups are being formed that cannot be sub divided and thus after going through all the recursive calls, the algorithm reverts back to brute force approach to complete the region matching. However such a scenario is impossible and has be created only for evaluation purposes.

5. Conclusion and Future Work

In this paper we presented a recursive algorithm for region matching that can be applied for interest management systems that uses multi-dimensional routing spaces. The concept of routing spaces for interest management has been adopted in standard distributed simulators and online games platforms. For scalability and performance, these systems require efficient solutions to interest management and the algorithm presented in this paper has potential to be used in these systems. The simulation results show promising results for region matching as compared to more traditional brute force approach. The results were even more impressive when large numbers of extents were involved. However, proper mechanism is required for setting the values of the threshold and we are researching techniques that can be used for that purpose. Further, for complete scalable interest management system, connectivity is to be established between nodes whose extents overlaps in order to do information exchange and therefore real evaluation of the algorithm would entail consideration of the network bandwidth, connectivity mechanism, allocation of multicast address, etc to do information exchange as done in real time online networked games. For this, we plan to integrate the algorithm in federated simulations development kit (FDK) [21]. FDK is an open source implementation of HLA standard developed at Georgia Institute of Technology. In our previous work [22], we discussed approaches for integrating FDK with game engines like Ogre3D so that it can be used for distributed agent simulation in online games. The algorithms integration in the FDK will not only improve the current HLA architecture but also provides middleware services for scalable online games.

References

[1] Singhal S, and Zyda M. 1999. *Netwroked Virtual Environments: Design and Implementation*. Addision Wesley

[2] Morse K. 2000. “An Adaptive, Distributed Algorithm for Interest Management”; *PhD Thesis*, University of California, Irvine

[3] US Defence Modelling and Simulation Office. 1998. High Level Architecture (HLA)- Interface Specification, version 1.3

[4] Macedonia M, Zyda M, Pratt D, Brutzmann D and Barham P. 1995. “Exploiting Reality with Multicast Groups: A Network Architecture for Large-Scale Virtual Environments”; *IEEE Computer Graphics and Applications*, 15(3): 38-45

[5] Miller D and Thorpe J A. 1995. “SIMNET: The Advent of Simulator Networking”, *Proc. of IEEE*, 83(8): 1114-1123

[6] Greenhalgh C and Bendford S. 1995. “MASSIVE: A Distributed Virtual Reality System Incorporating Spatial Trading”, *Proc. of 15th International conference on distributed computing systems (DCS 95)*, IEEE Computer Society, 27-35

[7] Epic Games 1999. *The Unreal Networking Architecture*. World Wide Web, http://unreal.epicgames.com/Network.htm

[8] Yu A and Vuong S T. 2005. “MOPAR: A Mobile Peer-to-Peer Overlay Architecture for Interest Management of Massively Multiplayer Online Games”, *in proc. of International Workshop on Network and Operating systems Support for Digital Audio and Video*, pp: 99-104

[9] Liu E, Yip M and Yu G. 2005. “Scalable Interest Management for Multidimensional Routing Space”, *in proc. of the ACM symposium on Virtual Reality Software and Technology*, pp: 82-85

[10] Tan G, Ayani R, Zhang Y S and Moradi F. 2000a. “Grid-based data management in distributed simulation”, *In Proc. of 33rd Annual Simulation Symposium*, pp: 7-13, 16-20 April, 2000

[11] DIS. 1995. IEEE 1278 Standard for Distributed Interactive Simulation

[12] GauthierDickey C, Lo V and Zappala D. 2005. “Using n-trees for scalable event ordering in peer-to-peer games”, *In proc. o of International Workshop on Network and Operating systems Support for Digital Audio and Video*, pp: 87-92

[13] Logan B and Theodoropoulos G. 2000. “The Distributed Simulation of Multi-Agent Systems”, *in proc. of the IEEE-Special Issue on Agent Oriented Software Approaches in Distributed Modelling and Simulation*

[14] Morgan G, Lu F and Storey K. 2005. “Interest management middleware for networked games”, *In proc. of the ACM symposium on Interactive 3D graphics and game*, pp: 57-64

[15] Abrams H, Watson K and Zyda M. 1998. “Three-Tiered Interest Management for Large-Scale Virtual Environments”, *in proc. of the ACM symposium on Virtual Reality Software and Technology*, pp: 125-129

[16] Van Hook D, Rak S and Calvin J. 1996. “Approaches to RTI Implementation of HLA Data Distribution Management Services”, *in 15th Workshop on Standards for the Interoperability of Distributed Simulations*

[17] Rak S, Salisbury M and MacDonald R. 1997. “HLA/RTI Data Distribution Management in the Synthetic Theatre of War”, *in proc. of Fall Simulation Interoperability Workshop*, Orlando, Florida, USA

[18] Roy A. 2000. “Dynamic grid-based data distribution management in large scale distributed simulations”, *MS Thesis*, Department of Computer Science, University of North Texas

[19] Tan G, Ayani R, Zhang Y S and Moradi F. 2000b. “A Hybrid approach to Data Distribution Management”, *In Proc. of 4th IEEE International Workshop on Distributed Simulation and Real-Time Applications*, pp: 55-61, San Francisco, CA

[20] Raczy C, Tan G and Yu J. 2005. “A Sort-Based DDM Matching Algorithm for HLA”, *in ACM Transactions on Modeling and Computer Simulation*, pp: 14-38

[21] FDK- Federated Simulations development kit. Available: http://www.cc.gatech.edu/computing/pads/software.html

[22] Kumar P. 2005. “Towards Integrating Ogre3D with FDK”, *in proc. of 7th International Conference on Computer Games (CGAMES)*, Angouleme, France

Session 7

An Optimised Implementation of the A* Algorithm using the STL

Bryan Duggan, Fred Mtenzi
School of Computing,
Dublin Institute of Technology,
Dublin 8, Ireland.
{bryan.duggan,fred.mtenzi}@comp.dit.ie

Abstract

In this paper we present our work on developing a fast, memory efficient and student friendly implementation of the A* algorithm for use in teaching using the game Dalek World. We first present our criteria for evaluating an implementation of the A* namely that it should be easy for students to understand, support multiple heuristics and demonstrate some of the optimisation issues which are relevant to implementing path finding. We compare three candidate A* implementations using various data structures to hold the open and closed lists. We conclude that our own implementation based on the STL `map` and `priority_queue` data structures is both the easiest to learn from and also the fastest.

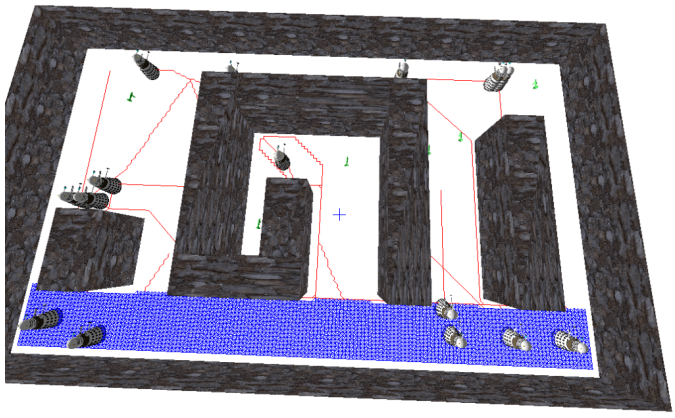


Figure 1: Dalek world with paths and part of the navigation graph drawn

1 Introduction

Our previous work [3] describes Dalek World, a framework we developed to teach a course in games programming at the School of Computing - Dublin Institute of Technology. In Dalek World, daleks need to navigate to ammunition spawn points, when they run out of ammunition. Figure 1 shows a top down view of Dalek World with shortest paths between daleks and ammunition drawn.

Path finding in the majority of commercial computer games is achieved using the A* algorithm [4]. In the second semester of our course, students learn how the A* algorithm is implemented in the game FarCry. Because the A* algorithm is so widely used in path planning, it is essential for students studying games programming to have an understanding of not only path finding [6], but also of the issues involved in creating a memory and speed optimised implementation. On our degree program, students learn the concepts of the A* algorithm in the third year AI Algorithms course. In fourth year, students learn how to implement the A* algorithm using C++ in Dalek World. This paper describes our work creating a "student friendly" (and optimised) implementation of the A* algorithm for Dalek World. Our criteria for implementing the A* algorithm in Dalek World were:

1. The implementation must be easy to understand. To support this, the C++ code should closely reflect the pseudo code and require as little "'extra"' knowledge as possible.
2. The implementation must support plug-in heuristics. This is so students can evaluate the algorithm for speed and efficiency using different heuristics, such as the Euclidean distance, the Manhattan distance and the distance squared.
3. It must support Dalek World sized navigation graphs. The world size in Dalek World is configurable, but a world size of 160 x 100 is used on the course. This results in a search space of 16000 nodes. Partitioning, triangulation and other techniques are not used at this point.
4. It must be fast and memory efficient as it will be called often and must not cause the program to "hang" as searches are carried out. As a benchmark, our goal was to make the implementation as fast as our adaptation of Buckland's implementation [2].

Our first attempt was to adapt Buckland's implementation from "Learning Game AI by Example" [2]. This implementation is both fast and flexible, however by the authors own admission, the implementation is difficult to follow. We therefore implemented our own version of the A* using various data structures from the STL to hold the open and closed lists. Our implementation is fast and memory efficient and in fact in most cases, it outperforms our adaptation of Buckland's implementation over the same search space and using the same heuristics. It also much easier to understand.

The rest of the paper is organised as follows. Section 2 presents a brief introduction to the A* algorithm. Section 3 describes our adaptation of Buckland's implementation and critiques it from the perspective of the criteria outlined above. Section 4 presents our implementation. We also present our performance data and compare our implementation with our adaptation of Buckland's. Section 5 presents our conclusions.

2 The A* Algorithm

The A* algorithm is a graph search algorithm. Pseudocode for the A* algorithm is presented in Figure 2[1]. First the search space (the world in this case) must be represented as a graph of navigable locations (nodes). Each *node* in the graph is connected to other nodes via an *edge*. An edge represents whether it is possible to travel directly from one node to another. Figure 1 shows a partial navigation graph generated using a flood fill algorithm in Dalek World.

The A* algorithm keeps two lists of nodes. An open list contains all the nodes found that have yet to be expanded and a closed list contains all nodes that the algorithm no longer needs to consider. The algorithm works by first adding a node representing the start position to the open list. The algorithm proceeds into a loop, popping off the node with the lowest f score from the open list until either the destination position node is popped or the the open list is empty. Each node that is popped (that is not the destination node) is expanded. When a node is expanded, each adjacent node is first checked to see if it is navigable. If it is navigable, the adjacent node is then checked to see if it is on the closed list. If its is not on the closed list, it is checked to see if it is on the open list. If it is on the open list, the f, g and h scores are calculated. The h score is the heuristic distance from the current node to the destination node. The g score is the cost from the source node to the current node and the f score is the sum of these two. If the node on the open list has a higher f score, it's parent and scores are updated. [2] refers to this as *edge relaxation*. Otherwise the nodes scores are calculated and it is added to the open list. Once the destination is found, the loop terminates and the path is generated by iterating from the destination to the start node via parent nodes.

[1] Adapted from [5], [2], [7]

```
function findPath(Node start, Node end) {
  List open, closed;
  open.empty();
  closed.empty();
  start.f = start.g = start.h = 0;
  open.push(start);
  while(! open.isempty()) {
    bestNode = open.popLowestFScoreNode();
    if (bestNode == end) {
        \\Iterate from end to start via parents
        \\Construct the path;
    }
    closed.push(bestNode);

    parent = bestNode;
    foreach adjacentNode in bestNode.adjacentNodes() {

      if (adjacentNode.isTraversable()) {
        if (! closed.exists(adjacentNode)) {
          if (! open.exists(adjacentNode)) {
            adjacentNode.h = heuristicCostTo(end);
            adjacentNode.g = parent.g + costTo(adjacentNode);
            adjacentNode.f = adjacentNode.g + adjacentNode.h;
            adjacentNode.parent = parent;
            open.push(adjacentNode);
          } else {
            // Edge relaxation
            openNode = open.find(adjacentNode);
            adjacentNode.g = parent.g + costTo(adjacentNode);
            if (adjacentNode.g < openNode.g) {
              openNode.g = adjacentNode.g;
              openNode.parent = parent
            }
          }
        }
    }
}
```

Figure 2: A* Pseudocode

3 Buckland's A*

Our first attempt at implementing the A* in Dalek World was to adapt Buckland's A* implementation from "Programming Game AI by Example" [2]. In his book, the author first describes Dijkstra's shortest path algorithm and continues by incorporating a heuristic to choose the next node to expand. This is the feature of the A* algorithm that gives it its efficiency over Dijkstra's algorithm. The author describes his implementations as follows:

> *"The implementation of Dijkstra's shortest path algorithm can be gnarly to understand at first and I confess I've not been looking forward to writing this part of the chapter because I reckon it's not going to be any easier to explain."*

Buckland's implementation is indeed hard to understand and is implemented in no less than eleven C++ classes. It is however fast and flexible due to several

factors:

1. It uses templates for nodes and edges, therefore can be used with graphs of any type, (not just navigation graphs).
2. The heuristic is also a template, meaning that any heuristic can be used with the algorithm.
3. It uses its own data structures rather than the STL for the open and closed lists, including an implementation of an indexed priority queue for the open list. It has been suggested that greater efficiency will be achieved if the STL is not used for the data structures (our implementation suggests this is not necessarally the case).

On the other hand:

1. It is difficult to understand and requires eleven classes and many hundreds of lines of code to implement (see table 1). We suggest therefore it is not a good implementation for learning purposes.
2. In his demos, Buckland pre-generates the graph. This is the approach we follow in our adaptation of Buckland's implementation also. It uses a flood fill algorithm to pre-generate all the nodes. In Dalek World, this results in a graph of 16000 nodes. Were the world to be bigger, the graph would consequently increase. For example for a world of size 500 * 500, the graph would be 250000 nodes. We suggest therefore, that it would be more memory efficient were an alternative approach to be adopted that did not require that the graph be pre-generated.
3. The flood fill algorithm used first adds every single node to the graph, even those that are not navigable and then tags non-navigable nodes as invalid. We suggest that this is also inefficient as many nodes are added and stored unnecessarily.

Table 1: Classes used to implement Bucklands's A$*$ in Dalek World

Class	Description
SparseGraph	A sparse graph that uses node and edge templates
GraphNode	Base class for a graph node
NaveGraphNode	A node in a navigation graph
GraphEdge	An base class for an edge connecting two nodes
NavGraphEdge	An edge connecting two nodes in a navigation graph
GraphHelper	Generates a graph using the flood fill algorithm. Also draws the graph and generates a Path from the Graph_SearchAStar class
Graph_SearchAStar	This class implements the A* algorithm
PriorityQ	A heap based priority queue
PriorityQLow	A 2-way heap based priority queue implementation
Path	A container for a vector of waypoints
Heuristic_Euclid	Implements the Euclidian distance heuristic

Table 2: Classes used to implement A* in Dalek World

Class	Description
Node	A navigation graph node
PathFinder	Implements the A* algorithm
Path	A container for a vector of waypoints

4 Our A* Implementation

Having adapted Buckland's implementation, we felt that an alternative "student friendly" implementation had to be developed, which was equally as efficient but perhaps sacrificed some of the flexibility in order to be easier to understand. We therefore developed our own A* implementation using the STL. Our implementation has several features that make it a better fit for teaching:

1. It is implemented using just three classes. (See Table 2).
2. To use the implementation to calculate a path requires just as little as two lines of code. This compares with eleven for our adaptation of of Buckland's implementation.
3. It is implemented using the STL to hold the open and closed lists and therefore does not require the coding of any additional data structures.
4. It supports configurable cost and heuristic functions.
5. It does not require a pre-generated graph. Instead, nodes are added to the graph as required as the algorithm proceeds.

Our implementation creates graph nodes on the fly as required. Thus it does not require the graph to be pre-generated. A node is represented using an instance of the Node struct in Figure 3.

We need to access nodes on the open list by retrieving the node with the lowest f score, and also by position vector (to check if a node is on the list). We need to access nodes on the closed list just by position vector.

We first used the STL `list` template data structures to hold the open and closed lists. A list has a member function `sort`, which sorts the elements by a configurable sort order. To sort the list it is necessary to provide a class functor which compares two nodes. To retrieve nodes by position, `list` has a member function `find_if`. Using this function with a comparison functor argument, it is possible to retrieve a matching element, (to check if a node exists on the list).

Using STL `lists`, our implementation was as much as ten time slower than our adaptation Buckland's im-

Figure 3: A graph Node

```
struct Node
{
  D3DXVECTOR3 pos;
  float f;
  float g;
  float h;
  Node * parent;

};
```

Figure 4: Data structures used by our implementation

```
priority_queue<Node*, vector<Node*>, NodeLessFunctor>
_open;
map<D3DXVECTOR3, Node*> _openMap;
map<D3DXVECTOR3, Node*> _closed;
```

plementation. Profiling of our implementation revealed that our implementation spent up to 70% of its execution time sorting the open list and retrieving elements from the open and closed lists. This confirms [8]'s assertion that the representation of the underlying search space used will have an impact on the performance and memory requirements of a path finding system.

We then optimised the implementation in two ways. The first optimisation we performed was to use an STL `priority_queue` to hold the open list. A priority queue is a data structure in which only the largest element can be retrieved (popped) [1]. An STL `priority_queue` can be set up using a class functor to compare elements. This facilitates program defined sort order. The class functor we used is shown in Figure 5. Using this class results in the priority queue being sorted in ascending f score. Because elements are inserted *in order* into a priority queue, there is no need to sort the data structure. The second optimisation we used was to keep a *second* copy of the open list in an STL `map` data structure using node position vectors as a keys. As entries in both data structures are pointers, there is a minimal memory overhead involved in keeping a second copy of the lists in a second data structure. This eliminated the need for the sequential search. Declarations of all the data structures used are presented in Figure 4.

Figure 6 compares search times for our adaptation of Buckland's implementation with our implementation using the STL `priority_queue` and STL `map` data structures. To generate this data, twenty searches were performed using various start and end points in Dalek World. The program was then executed ten times (performing the same searches each time) and the times taken by each A* implementation were averaged. We observed that in fourteen of the twenty searches, our implementation either outperformed our adaptation of Buckland's A*, or was at least as fast. Overall our algorithm was faster by 34% that in fourteen of the twenty searches, our implementation either outperformed our adaptaion

Figure 5: Class functor used to sort the `priority_queue`

```
class NodeLessFunctor
{
    Node * p;
    public:
    NodeLessFunctor() { p = NULL;}
    NodeLessFunctor(Node * p) : p(p) {}
    bool operator()(Node * f1, Node * f2)
    {
      return (f1->f > f2->f);
    }
};
```

of Buckland's A*, or was at least as fast. Overall our algorithm was faster by 34%.

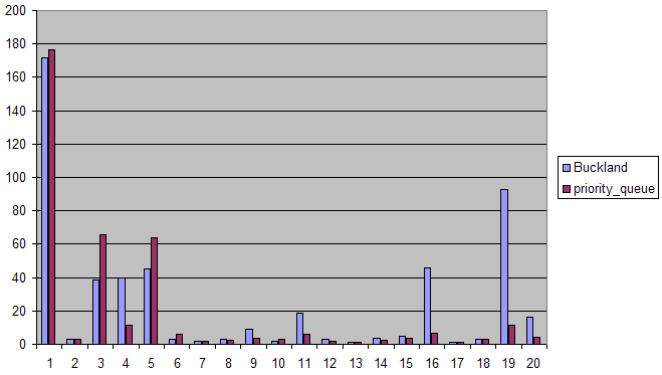


Figure 6: A comparison of average search time for our adaptation of Bucklands A* algorithm with our implementation based on the `priority_queue` and STL `map` data structures

Figure 7 compares search times for our implementation using the STL `priority_queue` and STL `map` data structures using three heuristics, the Euclidian distance, the Manhattan distance, the distance squared and using no heuristic. Again, twenty searches were carried out using various start and end points, the program was executed ten times and the average search times were profiled. As expected, using no heuristic, the number of nodes expanded was greatly increased and therefore the time taken by the search to complete was higher. What is interesting to observe from this data is that using the Euclidian distance heuristic is often fastest, despite the fact that the computational overhead for this approach would seem to be greatest. We speculate that this may be because to calculate the Euclidian distance, the DirectX API call `D3DXVec3Length` is used and this call must be hardware accelerated.

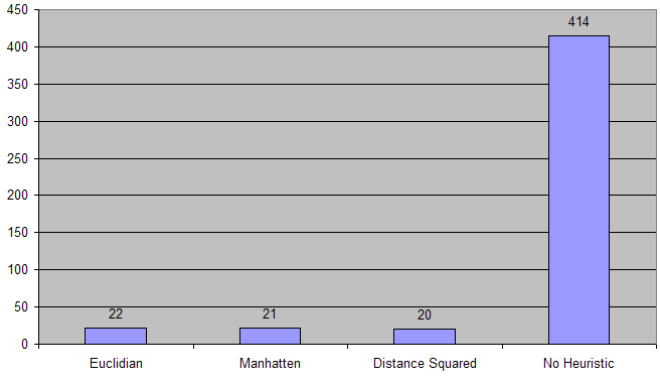


Figure 7: A comparison of search times in milliseconds using different heuristics

5 Conclusions

This paper presented our implementation of a student friendly yet efficient implementation of the A* algorithm. We first adapted Buckland's implementation of the A* algorithm in our game framework Dalek World. We conclude that this implementation is both flexible and efficient. However, we further conclude that due to the complexity of the implementation, it is not ideal for use in a teaching environment. We then described our own simplified implementation of the A* algorithm, generating graph nodes on the fly and using the STL `list` template data structure to hold the open and closed lists. We conclude that based on data gathered from search executions, this implementation is also not suitable because it is too slow. Finally we presented our optimised implementation of the A* algorithm using the STL `map` and `priority_queue` data structures to hold the open and closed lists. We presented data demonstrating that in Dalek World, it is on average the fastest of the three implementations. We can therefore say that it is possible to create a highly optimised implementation of the A* algorithm using data structures from the STL, if appropriate data structures are chosen. We also conclude that using three classes as opposed to eleven, our implementation is more suitable for learning the A* algorithm.

On the other hand, we acknowledge that our implementation lacks the flexibility of Buckland's implementation. We also conclude that due to the fact that our implementation queries the world repeatedly to check if a node is navigable, it is susceptible to optimisations in this area.

All the code used for teaching and testing this paper can be downloaded from http://www.comp.dit.ie/bduggan/games.

6 About the Authors

Bryan Duggan is a lecturer in the school of computing at the DIT in Kevin St. He hold a first class honours degree in computer science and software engineering from the University of Dublin (studied at the DIT) and a masters degree in Information Technology for Strategic Management (DIT). He is presently working on a PhD with a working title of "Modeling Creativity in Traditional Irish Flute Playing". He lectures on games programming and music technology in the DIT School of Computing.

Fred Mtenzi received his B.Sc degree from University of Dar es salaam - Tanzania in 1989. He also received his M.Sc and PhD from University College Dublin - Ireland graduating in 2000. Currently, he is a lecturer in the School of Computing at the Dublin Institute of technology - Ireland. His research interests include design and implementation of heuristic algorithms for combinatorial optimisation problems, security issues in mobile devices, games and healthcare systems and design of power aware protocols for Mobile Ad Hoc Networks.

References

[1] Leen Ammeraal. *C++ for Programmers.* Wiley, 2000.

[2] Mat Buckland. *Programming Game AI by Example.* Worldware Publishing Inc., 2005.

[3] Bryan Duggan. Learning games programming using "dalek world". May 2005.

[4] FarCry. *FarCry AI Bible.* Crytek, 2003.

[5] Mario Grimani and Mathhew Titelbaum. Beyond a*. *AI Game Programming Wisdom 2*, 2005.

[6] IGDA. *IGDA Curriculum Framework.* Crytek, IGDA.

[7] Patrick Lester. A* pathfinding for beginners. accessed from http:www.policyalmanac.orggames-astartutorial.htm, 2005.

[8] Paul Tozour. Search space representations. *AI Game Programming Wisdom 2*, 2005.

Personality Profiling Agent using Computer Games

Sidi O. Soueina[1], Ahmed H. Salem[2], Kondala Rayudu Addagarla[3], Adel S. Elmaghraby[4]
[1]Computer Science Department, Sullivan University, Louisville, KY 40205,
[2]Computer Science Department, Hood College, Frederick, MD 21701,
Suryacom Consulting Services, Lexington, KY. 40511,
[4]Computer Engineering and Computer Science, University of Louisville, Louisville, KY 40292,
ssoueina@sullivan.edu, asalem@hood.edu, rayudu@suryacs.com, Adel@louisville.edu

Abstract - This paper indirectly lays out the theoretical foundation for reliable personality profiling and detection methodologies and their application in education. We begin by discussing two things: the shortcomings of ways classical personality profiling methods (e.g., Enneagram and BMTI) are applied and the need for seamless personality traits detection means. Based on a formal relationship between personality, skills, and fixations, we introduce a framework of an interaction model for personality recognition, using a computer word-find game, and we discuss issues pertaining to the game player's subconscious controlling the eye's scanpath (fixation and saccades) on words that closely relate to the personality type without the player's awareness.

Keywords - Personality Profiling, Educational Games, Eye Movement, Cognitive Science.

1. INTRODUCTION

Computer gaming techniques have been used to enhance the delivery of online educational material. Learning Strategies (LS) [1][2][3] are educational delivery methodologies used by teachers and instructors in online and regular class environments. Examples of learning strategies are experimental learning, collaborative and active learning. In order to facilitate learning, these strategies incorporate different methods of information delivery, for example to break down complex theories and problems into simple parts, or to metaphorically introduce complex problems and theories in ways to which the students could relate . Learning strategy nonetheless does not systematically employ or *measure* specific personality traits and learning capabilities of the students.

Although empirically simple, the consideration of regular personality profiling techniques brings different issues. In classical Personality Profiling (PP) methodologies [4][5][6][7] for example people are presented with questionnaires. Typically, the subject student would choose a level of similarity close to his or her personal behavior and emotions using a scale, e.g., of 1 to 3, where 1 is the least likely of being similar and 3 the most. The subject's Personality Type (PT) is classified, depending on the highest similarity score of any of the known personality types (PTs). The main problem with scale-based questionnaires, however, remains reliability. The non-measurable level of 'frankness' brings about the issue of the candidness of the selections made by subjects that can be classified as a major hindrance against accuracy of the selections and thus of personality typing as a whole. The person being tested is subject to the influence of her of his own *ego* manipulations (see definition 1) whose effect is mainly, invisible even to herself.

We are investigating seamless methods to detect personalities and intellectual traits in ways that are immune to the influence of the ego. In this paper we are introducing a seed model and a computerized word find game, where subjects are asked to find words hidden in a field of randomly displayed characters. Here we are motivated by the assumption that the subconscious does automatically identify the words that closely relate to the personality type of the person, even if those words are not uniformly displayed in front of the person in a legible format. We engage a discussion of issues pertaining to eye-movement and its relation to the cognitive mind, and we draw possible research directions from that.

In the following section we will introduce one of the basic personality profiling methodologies and its brief history. Formal definition personality types and their relationship to mental traits and skills are also presented in this section. In section 3, we introduced the game model. In section 4 we discuss motivations pertaining to the eye movement and its relationship to personality. We conclude with a discussion and new research directions.

2. PERSONALITY AND SKILL

The Enneagram is a psycho-spiritual typing theory developed in the first Millennia by Sufi Philosophers [8]. Its original aim is "to understand the self and put the *ego* under control in order to reach enlightenment". This paper is obviously not focused on the spiritual dimension of the Enneagram. The Enneagram has long been tested scientifically, and it is that aspect which we will discuss. The Enneagram clusters the human behavior into 9 sets, all according to hidden conscious and subconscious plans, called the Enneas (or *intentions*). Every set of intentions tends to cause individuals who experienced certain pivotal events and lived in certain familial or societal environments, to behave in certain recognizable patterns and thus to develop specific identifiable mental and social skills. These experiences are directed by certain features of the phenomenon called the *ego* (see definition 1). Most interestingly, nonetheless, is that the personality types do cause the person to develop certain mental qualities more than others. For examples, the fear of being deprived is what causes the personality Type 7 (TP7) to be creative and thus to be able to generate more new ideas in

comparison with other personality types. Personality Type 5 (PT5) are known for a behavioral patterns named *withdraw*-and-*observe* which drives their intellectual qualities to higher performance causing most of them to be scientists[1].

2.1. The Nine skills set

Personality can thus be defined to be a set of human beliefs emotions and behavioral patterns that have developed over time. Universally there is no reconciliation on a fixed number of personality types. It seams however that every personality shares a central motivator or manipulator called the *ego* and a pattern of repetitive behavior called the *fixations* (Definition 2). The intensity of behavior or skill levels in dealing with events can be measured by the level of *fears & desires* which originally gave birth to the fixations (Definition 2). Given an event set of events e a personality of a human can be define as following:

$$P = \sum_{i=0}^{i=m} f_i + S_i$$

such that f_i is the set of fixations or actions that generated feelings of safety or happiness and s_i is the set of the new skills gained during the trials and later in life. **Figure 1** depicts the main components of personality as follows:

***Definition* #1. The Ego.** The ego is not defined in the typical societal sense. Rather it is being defined as the psycho-emotional phenomena that *manages* one's intentions and that at the same time generates human behavior to guarantee certain materialistic and emotional needs. The intentions are not always apparent nor understood by the subject person, or other observing humans. The ego is being developed throughout the history of the personality but especially during early childhood where an event or a series of events made the subject *fixated* on certain behaviors or beliefs.

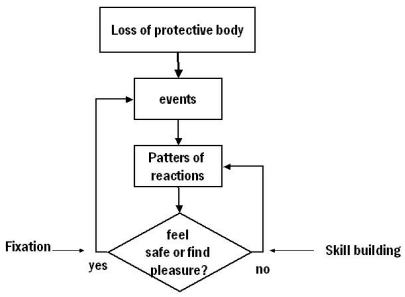


Figure 1. Seeking of pleasure and avoidance of fears lead to forming a Fixation and skills building, or personality.

Definition #2. *Fixations.* Every ego has a set of fixations or traits which are mainly behavioral patterns motivated by the avoidance of fears or seeking of pleasures. Grounded in reality or not, a fixation is a set of actions and thoughts that generate a dose of satisfaction that makes the ego "feel safe" ones again. The fixation is an addictive behavior. We are willing to fight to maintain our fixations intact, for through them, we feel safety or pleasure.

Fears & Desires. The loss of the *protective body*, such as the mother, at an early life stage, drives the self to seek protection and safety. A feeling of safety is achieved through fixating the behavior on a particular pattern that make us feel safe or feel happy after performing it, which in turn makes us tend to repeat those behaviors.

Definition #3. Skills & Traits. A trait, such as an intellectual trait, describes the capabilities of the person gained during skill building. It is an expressive phenomenon that is a combination of skills and fixations. A trait can be defined as an object containing *action-type, subject-actions, state, scale and a life-span*:

- Trait-type: defines a general or intellectual trait.
- Subjects-Actions: specific predefined variables representing actual actions.
- State: is an instance of a subject-actions set.
- Life Span: A trait is born when its subject-actions reach a significant repetitive threshold.

For the sake of this example, we define an Object trait in BNF as following:

```
Object :: <trait-type><Subject-actions><Scale> <Life-span>
<action-type> :: <String>
<Subject> :: <act-type>*
<act-type> :: <String>
<Scale> :: <action> | time| state>*
<action> :: <String>
<time> ::  <String>
<State> :: <String>
<Life-span> :: [<Subject-action>*<time>*]
<Subject-action> ::<Subject> <State> <Scale>
<String> :: [A – Z]
```

2.2. Personality Types and applications

The non-spiritual applications of the Enneagram are numerous [9]. They range from Business and politics to therapy. In [10] we introduced the foundation of an Enneagram based adaptive architecture for a Multimodal in-Car navigation system where the traits of the 9 Personality Types (9PTs) were to be identified by the system through, among other means, a Dialog Manager, Speech Recognition and Facial Recognition modules. To date however we feel that the Enneagram application in education has not yet received the attention it deserves. A goal of such a study would be to reveal the learning personalities of students; match students with specific subjects of study or specialties, maximizing by that the odds of success, and of course, tailoring the teaching to students' capabilities. In this paper we discuss this issue and present a seed solution pertaining to this topic. Since the target is to advice present of future students on fields of study, we are introducing a slight modification in the *naming* of the 9PTs based on every type's primary intellectual traits (see a list of the 9PTs in **Table 1**).

[1] See conclusions and discussion for a suggested research direction to quantify this causal relationship.

Although more conclusive research needs to be done to reliably extract the different traits, thus far we can empirically associate the 9PTs with different subjects of study solely based on the Intellectual Traits we are naming through the identification of the know Intellectual qualities of every type. Note however that each intellectual trait we suggest is not a sole representative of the type career recommendation, but of all closely matching groups of specializations.

Enneas	Gen-Traits	Int-trait	careers
1	Rational, Idealistic, Principled, self-controlled & perfectionist …	Rational Thinking	Engineering
2	Possessive, Caring, interpersonal, Generous, pleasing…	Preventive Thinking	Strategy and managerial
3	Adaptive, driven, image-conscious …	Pragmatic thinking	Design
4	Sensitive, Withdrawn, Expressive …	Imaginative	Art
5	Intense, Cerebral, Perceptive, isolated…	Problem Solver	Scientist
6	Loyalist, engaging, responsible, Anxious, Suspicious …	Preventive	Unions
7	Enthusiastic, Busy, Fun-Loving, Spontaneous, versatile…	Creative thinking	Inventor, CEO,
8	Challenger, Self-Confident, confrontational…	Decisive thinking	Conflict resolution, political science
9	Peacemaker, Receptive, agreeable …	Reconcile	Peace activism

Table 1. A basic format of the general traits of the 9PTs vs. the main Int-traits and few examples of their recommended subjects of study and careers.

2.3. Intellectual Traits

In traditional Enneagram classification, General Traits and Intellectual Traits each develop in a lifetime of conservation of seeking basic desires and avoiding basic fears. In the following section, taking the original 9PTs in consideration, we loosely define the concept of intellectual traits and discuss the rational of the desires and fears that groomed them. See section 5 for a discussion on a suggestive study on the relationship between basic fears, basic desires and skills.

Assumption #1. *Skill are basic behavior of every personally developed through the struggle to balance the basic fears and basic desires, motivated by the loss of protected body.*

Assumption #2. *Every general traits and intellectual trait is a skill.*

Rational Thinker: The *World View* of the rational thinker is that the world is an ill perfect environment, and his/her job is to make it perfect. The *Basic Desire* is to be right and the *Basic Fear* is the fear of being condemned. The balance of the fears and desires makes the Rational Thinker more and more skills at bringing about sound solutions to every day problems.

Rational Thinker. *World View*: His/her conviction that the world depends on them in order to function. Their *Basic Desire* is to be loved, and *Basic Fear* is to be unloved. *Basic character* are generosity, friendliness, pridefulness, reassuring. *The balance* of fears and desires make preventive thinker develop care giving and compassionate *skills*.

Pragmatic Thinker. *World View*: The motivator believes that the world gives great consideration and respect for a champion and that is why, they try to avoid failure at all costs. Their *Basic Desire* is to be admired and *Basic Fear* is the fear of being rejected. The Balance of basic fears and basic desires give the change for the pragmatic thinker to develop skills in generating grounded solutions in all circumstances to satisfy all parties.

Imaginative Thinker *World View*: Something's missing in themselves and others have it. *Basic Desire*: to understand self and *Basic Fear*: is the fear of being defective. The imaginative thinkers creative skills are developed through scanning wide range of option and alternatives.

Problem Solver Thinker. *World View*: The world is invasive and confusing. I need privacy to think. Basic Desire: to understand the world Basic Fear: is to be overwhelmed by the world. This type is insightful, theoretical, detached, eccentric, and intense. They can be extremely brilliant and inventive, but also can be alienated. Their problem solving skills are developed due to isolation and desires to understand the world.

Preventive Thinker. World View: The world is a threatening place. Their Basic Desire is to be secure and Basic Fear: is the fear of being abandoned. They primary motivation of this type is to find security, resolve their paranoia. Their preventive skills are developed through experience in always trying to avoid danger.

The Inventor Thinker. World View: The world is full of opportunity and options. I look forward to the future. Their Basic Desire is to be happy and Basic Fear is fear from of being deprived and gluttony which are the main motivator of this type to generate new ideas.

Assumption 3. The subconscious identifies phenomena that closely relates to the personality type of the same person more than others, e.g., PT9 first reorganizes (discovers) words that relate to PT9 traits such as “Peacemaker” and “Receptive” before it does other words.

Assumption 4.*The eye of a game player is directed by the cognitive mind of the player to identify with artifacts, e.g., words that most related to his or her personality using a weight*

Decisive Thinker. World View: believe that the world is an unjust place. I am strong and I ‘defend the innocent’. Their Basic Desire is to be self-reliant. Basic Fear is the fear from submitting to others. The fears and desires of the decisive thinker makes them develop leading attitudes and skills.

Reconciler Thinker. World View: It's best to keep the peace. Basic Desire: to find union and peace Basic Fear: fear of separation. The reconciler’s efforts to unite separated parts stems from his fears of separation.

3. PROFILING AGENTS AND WORD-FIND GAME

In section 1 we briefly introduced the classical dilemma of having to profile the personality through the questionnaires. The main problem in this approach is that we are dealing with the *ego* as defined in section 2 above. We mentioned that the ego is capable of unconsciously piloting the subject to place untruthful answer to some of the questions in the questionnaire. It makes thus sense to find a method(s) allowing for an ego-independent traits extraction. One of such method is to observe the subject interact in an environment, far from ego stimulus that might be related to the ultimate goal of personality identification.

Computer games are a good alternative canvas because they gear the person towards his or her natural state of interaction to deal with solving the game rather than self evaluation. The user does have to willfully intend to play in un-orderly manner for the test to fail. We built and tested a world-find game (Figure 3) based a on a multi-agent model of three agents (Figure 2).

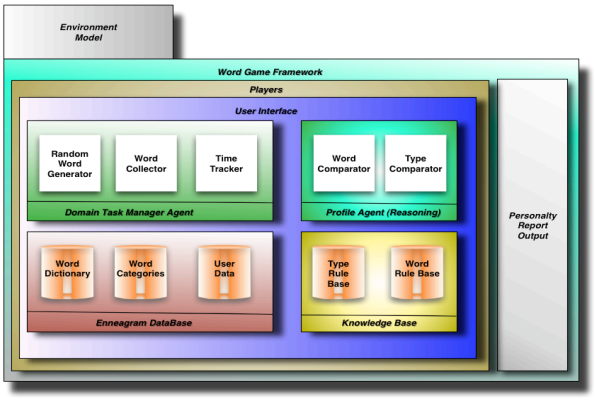


Figure 2. an agent-based model of the world find game

The Domain Task Management Agent (DTMA) is a generic agent responsible for managing required data for personality profiling based on the domain application in use. The inference capability of such an agent is standard, i.e., by changing the domain database; in our case the Enneagram database, other domain database such as the MBTI can be used. The DTAM has three main parts, which use different databases, namely the word dictionary, world category and user data. Second, the Word Categorizer, which divides and organize the words in the Enneagram dictionary into 9 Types based on the personality type. Finally the User Database records the user interaction wit the game and identifies the user, subject personality type. This is later on used by the profiler agent to generate the Personality Report.

The profiling agent is an intelligent agent that uses detailed database of *rules* describing conditions for identifying personality types and the types of words that match them. The user interface of the game is depicted in Figure 3. The game starts by showing the user a group of words for two minutes. During these two minutes the player will only concentrate on the words and try to memories these words. These words are nineteen words, which is a description of the personality type. After the words are displayed to the players for two mints the game interface is displayed and asking the players to find ten words from the 19 words that were shown earlier, these 10 words are in the game matrix. The game has a timer that allows the player "Subject" to play the game for 20 mints. Once the 20 mints is passed the game notify the player that the time is up and the game will generate a report that indicates the player or personality type the game provide a detailed report regarding the time that took the player to find the words and also it generates a histogram report regarding the personality probability type.

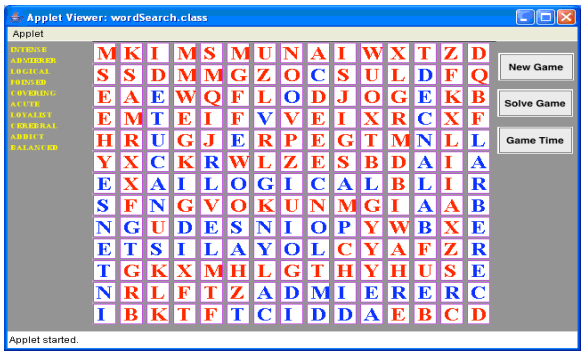


Figure 3. The user interface is a simple web-based word-find-game (WFG) that lets the subjects find words in a matrix of 15 by 13 characters consists of an array of 15 main game matrix, as shown in the following figure.

The first object in Figure 4 is the Random Word Generator. The purpose of this object is to randomly select words from the Word Categorizer database and to select 10 words from the displayed words that the subject or the player needs to memorize at the game start. The random words generator assures that the randomly chosen words are one from each personality type.

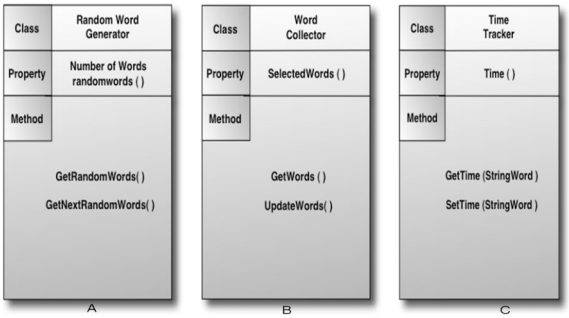


Figure 4. Domain/Task Management Agent.

The second object in Figure 4 is the Word Collector and its task is to collect the words from the random word generator object and place it in game grid interface. Lastly, the time tracker tracks the total of all game time and the time acquired

by the player to discover the words. Further the time tracker also brings game to an end after a particular set time.

4. EYE MOVEMENT RATIONAL

In the previous section we presented the game and claimed that the players' personalities can be identified merely by classifying the words the found and matching them into one or more of the 9PTs. In this section we will briefly introduce and discuss pertaining scientific research. The Human Eye moving on subjects exhibits two main phenomena, namely *fixations* and *saccades* [11]. The fixation is the amount of time the eye rests on a subjects while the saccades are fast movements that occur between fixations. Studying these two phenomena has made Eye-Movement part of the core research in many areas ranging from medicine [12], engineering [13] and entertainment [14]. In HCI (Human Computer Interaction) for instance scientist use eye-tracking in usability in order to measure the effect of colors and shapes, on how the users navigate through a website [15]. Our interest in eye tracking however is more related to cognitive science. Our goal is to study the personality through eye movements tracking while the subject is performing other activities mental and physical activities, such as playing a game. Our ultimate goal is to analyze the eye movements, actions in order to find credible evidence depicting the relationship between the general traits and intellectual and personality type.

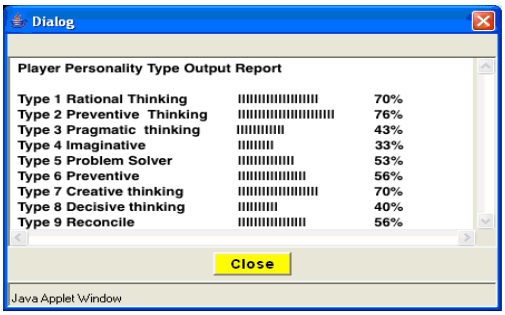


Figure 5. Personality Type Report.

5. CONCLUSIONS AND DISCUSSION

The fact that the ego is very manipulative and in control of most of the human activity and capable of sabotaging both the conscious and unconscious, makes the issue of directly probing the ego about the fixations, e.g., by expecting answers to written questions, rather an inaccurate way of discovering the fixations, traits and skills. It seams, therefore, that observing the ego in action is the only way for identifying the personalities / mental capabilities. This however brings forth many challenges and research directions. Mainly, how can we quantify and model the human fixations in order to discover personality traits and mental capabilities? What are the types of *artifacts* and external subjects can serve to devise an appropriate framework to study and discover personality traits. We intend to investigate other implementation models that will help us bring forth a demonstrable relationship between the subconscious and the actions performed. Most urgently however, we would like to perform a case study whereby the profiling results of this game are compared with a conventional Enneagram test, and further, using an eye tracking device, to study the significance of the eye's movements during the game. Furthermore we would like to define a model the relationship between the basic fears, basic desire that lead the birth of skills.

REFERENCES

[1] Tom Atkinson et. al. "GOALS: Graduate Online Active Learns Strategies". Journal Of Computing Sciences in Colleges Vol. 17, issue 3. February 2002.

[2] J. A. Polack-Wahl et. al. "Learning Strategies and Undergraduate Research". Proceedings of the 37th SIGCSE technical symposium on Computer science education SIGCSE '06. ACM Press, March 2006.

[3] John Rieman A field study of exploratory learning strategies ACM Transactions on Computer-Human Interaction (TOCHI), Volume 3 Issue 3. September 1996.

[4] [4] The Enneagram Inst. "Enneagram Research, Development & Applications" Retrieved from "www.enneagraminstitute.org" Retrieved March 2006.

[5] Linda V. Bernes "The 16 Personality Types, Descriptions for Self-Discovery". ISBN 0966462475.

[6] Don Richard Riso et al. "Personality Types : Using the Enneagram for Self-Discovery" Houghton Mifflin . ISBN 0395798671.

[7] Palmer, Helen "The Enneagram in Love and Work". Harper, 1995.

[8] Darrell M. Dodge. "Possible Applications for Therapy and Personal Growth for Speech-language Pathologists and People Who Stutter Retrieved from "http://members.aol.com/dmdodge/dw/ennestut.htm in 2006.

[9] Clarence Thomson, "The Enneagram Applications" C Thomspon et. Al. Editors ISBN 1555521037.

[10] Sidi O. Soueina "An Enneagram Based Model for Personality Based Adaptive Systems." Proceedings of AAMASS'03 Melbourne Australia.

[11] Dario D. Salvucci et. al. " Identifying Fixations and Saccades in Eye-Tracking Protocols". Proceedings of the 2000 symposium on Eye tracking research & applications ETRA '00, November 2000. ACM Publishers.

[12] Julie A. Jacko et. al. "Using eye tracking to investigate graphical elements for fully sighted and low vision users. Proceedings of the 2000 symposium on Eye tracking research & applications ETRA '00. November 2000.

[13] M. Sodhi et., al. "System and & applications I: On-road driver eye movement tracking using head-mounted devices" Proceedings of the 2002 symposium on Eye tracking research & applications ETRA '02. March 2002.

[14] Roland Arsenault et. al.," Eye-hand co-ordination with force feedback". Proceedings of the SIGCHI conference on Human factors in computing systems. ACM Press April 2000.

[15] Bing Pan et., al. "The determinants of web page viewing behavior: an eye-tracking study" Proceedings of the 2004 symposium on Eye tracking research & applications ETRA '04. March 2004. ACM Press.

[16] Daesub Yoon et. al. "Mental imagery in problem solving: an eye tracking study". Proceedings of the 2004 symposium on Eye tracking research & applications ETRA '04. March 2004 ACM Press.

[17] Christina Merten et al. "Gestural input: Eye-tracking to model and adapt to user meta-cognition in intelligent learning environments". Proceedings of the 11th international conference on Intelligent user interfaces IUI '06. January 2006.

Session 8

MUTI-LEVEL SB COLLIDE: COLLISION AND SELF-COLLISION IN SOFT BODIES

Jaruwan Mesit Ratan K. Guha Erin J. Hastings
jmesit@cs.ucf.edu guha@cs.ucf.edu hastings@cs.ucf.edu
Department of computer science, University of Central Florida
4000 Central Florida Blvd., Orlando, Florida 32826.

Keywords

soft body simulation, animation, spatial subdivision, spatial hashing

Abstract

In interactive 3D graphics collision detection of soft bodies in real time is a significant problem. It is time consuming because soft bodies are composed of possibly thousands of moving particles. Each time step all particles rearrange in new positions according to their behaviors and collision must be detected for each particle. To optimize collision detection in soft bodies, we introduce a solution called Multi-Level SB Collide. The method relies on the construction of subdivided bounding boxes, box hash functions, and contact surfaces. Multi-level SB collide applies multi-level subdivided bounding boxes (AABBs) into a box hash function and uses contact surface method to detect collision. This contact surface can be used to detect both collision with other objects and self-collision in soft bodies. Experimental results show that multi-level SB Collide is an accurate and efficient method for real-time collision detection in soft bodies.

1. Introduction

Collision detection is a fundamental issue in most all forms computer animation. Many algorithms are based upon types of bounding volume hierarchies. Some notable examples are: bounding spheres (Hubbard 1995, James and Pai 2004), axis-aligned bounding boxes (AABBs) (Bergen 1997, Teschner et al. 2003), oriented bounding boxes (OBBs) (Gottschalk et al. 1996), quantized orientation slabs with primary orientations (QuOSPOs) (He 1999), and discrete-oriented polytopes (K-DOPs) (Klosowski et al. 1998). For polyhedral objects, CLOD with dual hierarchy (Otaduy and Lin 2003) has been also proposed whereas Separation-sensitive collision detection (Erickson et al. 1999) was presented for convex objects.

Another method for collision detect is spatial subdivision. Numerous variations have been proposed such as: octtree (Moore and Wilhelms. 1988), BSP tree (Naylor et al. 1990), brep-indices (Bouma and Vanecek, Jr 1991), k-d tree (Held et al. 1995), bucket tree (Ganovelli 2000), hybrid tree (Larsson et al. 2001), and uniform spatial subdivision (Teschner et al. 2003). Some additional subdivision methods specifically suited for collision detection in large environments are I- COLLIDE (Cohen et al. 1995) and CULLIDE (Govindaraju et al. 2003).

However, most of the methods that have been mentioned are to solve collision detection between rigid bodies and other rigid bodies, or between soft bodies and rigid bodies, which is slightly less complex problem. In this paper we present a new efficient algorithm for collision detection among soft bodies, which is extended from Mesit et al. 2004. The algorithm is based on the classical methods of subdivided bounding volume and spatial hashing. The major contribution of this paper includes: 1) subdivided bounding box method, 2) the specialized hash function, and 3) the contact surfaces.

The paper will proceed as follows: first, an overview of the proposed algorithm, next analysis of the algorithm, then a set of the experiments with the algorithm and results, and finally conclusions and discuss possibilities of further work.

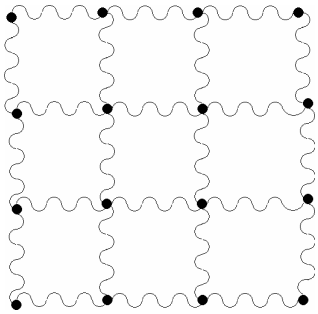


Figure 1: *The skeleton of a soft body object where the vertices are moving particles interconnected by force springs.*

1.1 Introduction to soft body simulation

A soft body simulation is usually a set of particles, each of which has its own position, velocity, and force. Particles are connected to neighbors by linear springs. Spring length, elasticity, and damping factor are defined by physical properties of the soft body. Figure 1 shows a skeleton of cloth and figure 2 presents two more types of springs called shear and bend to simulate a soft body. The shear and bend properties are required to allow the soft body to stretch in and out.

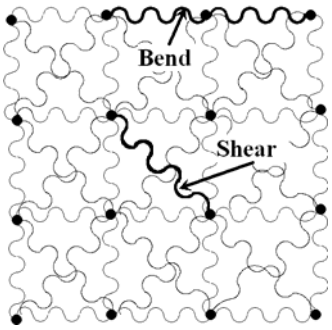


Figure 2: *Shear and bend properties used in soft body simulation*

1.2 Springs and Dampers

Springs are the structures that connect between two vertices. The function that we use to stretch or compress is spring force by Hook's law. This relates to the "rest length" of the spring and a "spring constant". The spring constant determines the stiffness of the spring. Damper is related to velocity of vertices, since each vertex has its own force and velocity. Damper acts against this velocity. If there are two vertices connected with springs, dampers slow the relative velocity between those two vertices. Figure 3 shows compressed spring and stretched spring.

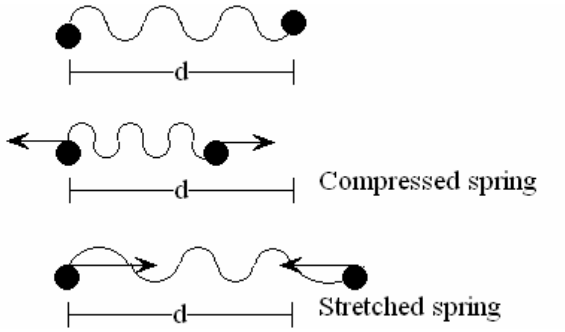


Figure 3: *Spring structure for soft body simulation with rest length = d*

2. Algorithm Overview

For collision detection with soft bodies in a large environment, we introduce three steps of the Multi-Level SB Collide algorithm:

1) *Multi-level Subdivided Bounding Box* (Multi-level SB): all soft bodies are surrounded by a bounding box (AABB) for tracking. Two points, a minimum and a maximum, define the bounding box. Then, the bounding box is subdivided to n-levels of sub division. We use 3-level subdivision for this simulation.

2) *Box Hash Function (BHF)*: we apply the box hash function to each point that we use to create the subdivided bounding box. One subdivided bounding box has 8 points. Each point is hashed and given hash index by box hash function. List of subdivided bounding boxes is put in hash table related to hash index. Then, we compute contact surface from vertices that belong to subdivided bounding box in the list of hash table.

3) *Collision for Flexible Models (CF)*: finally we detect collision for the models. If distance between points in the flexible models is less than collision tolerance, collision for flexible models is detected.

2.1 Multi-level Subdivided Bounding Boxes (Multi-level SB)

The beginning of our algorithm is to do tracking with multi-level subdivided bounding box. We are tracking our soft bodies by using AABB bounding box. Minimum point and maximum point

are calculated to bound our soft body. The bounding box is subdivided in to 2^n regions, where n is level of subdivision. Figure 4 shows 2-level of subdivision in which the second mid point is created from the first mid point.

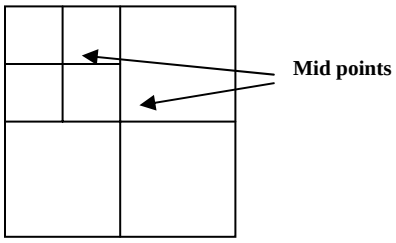


Figure 4: *Mid points for subdivision*

These objects are subdivided by using the midpoint. Then, we find depth, height, and depth of each subdivided bounding box as shown in figure 5.

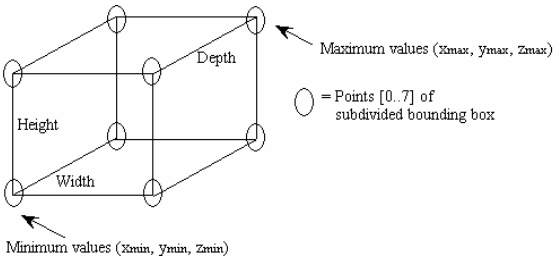


Figure 5: *8 points for a subdivided bounding box*

2.2 Box Hash Function (BHF)

We use **Box Hash Function (BHF)**. The idea is to use **BHF** and hash to 8 points that we use to create the subdivided bounding box. To facilitate hashing, initially a hash table is created which is based on grid size. The formula is as follows:

Index = (floor(Grid.min.x / length) * xprime+
floor(Grid.min.y/ height) * yprime +
floor(Grid.min.z / width) * zprime)
% bucketsize;

where length is grid length,
height is grid height,
width is grid width,
Grid.min.x, Grid.min.y, and Grid.min.z are minimum points of each grid,
xprime, yprime, and zprime are any prime number for x, y, and z.

Next, we apply **BHF** to 8 points of our subdivided bounding box shown in figure 6.

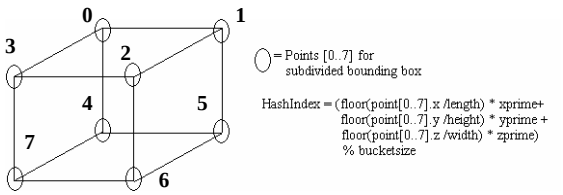


Figure 6: *Points of subdivided bounding box are hashed by hash function*

Since there are 8 points, [0..7], in one subdivided bounding box, **BHF** can be written as:

HashIndex = (floor(point[0..7].x /length) *
xprime+
floor(point[0..7].y /height) *
yprime +
floor(point[0..7].z /width) *
zprime) % bucketsize

where xprime, yprime, and zprime are any prime number for x, y, and z
length, height, and width are length, height, and width of grid cell.

When we have the hash index, we can put the subdivided objects to hash table. At this step, we create one hash table and we have the list of subdivided objects that have chances of collision. Figure 7 and 8 show that subdivided box 2 (SB2) and 3 (SB3) are in the same hash index which is index 6. Then, contact surface will be calculated.

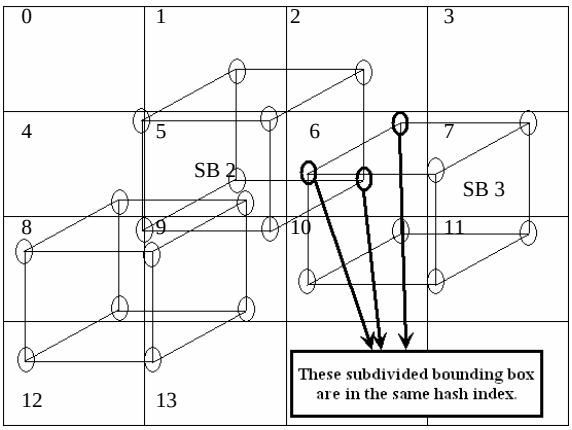


Figure 7: *The combination of grid and box hash function*

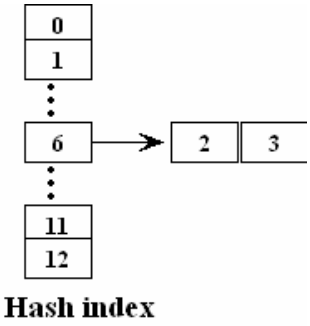
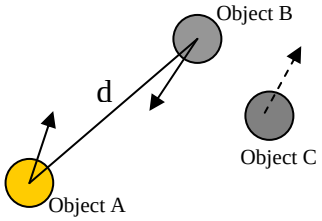


Figure 8: *Example of box hash table. Index 6 points to subdivided box 2 and 3*

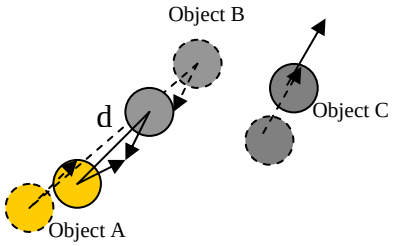
2.3 Self-collision and Collision for flexible models (SCF and CF)

In this step we find the distances of vertices in the hash table and then we compare with a collision tolerance. This step returns true or false. True is colliding while false is not colliding. Contact surface can be determined as follows:

Given position of object A and B, d is the distance between them. If they are in same hash table, then we find for d. if d < collision tolerance, then they collide.



Time t = n, d > collision tolerance, they don't collide.



Time t = n+1, if d < collision tolerance, they collide.

2.4 Self-collision in flexible object (SCF)

For every soft body object, it is necessary to solve self-collisions for internal constraints. Self-collision in soft bodies can be achieved by using the contact surfaces. The method of self-collision avoidance is to create collision tolerance around the points of flexible object. This collision tolerance force acts like a shield which rejects the points passing through the objects. Thus our algorithm solves for both self-collision detection and normal collision detection.

To summarize the algorithm, three major steps are computed for collision detection in every frame. First, the bounding boxes are subdivided into n-level of subdivision (3 levels in this simulation). Then, we apply the box hash function to subdivided bounding boxes and create hash table. Finally, for each box hash table, we compute the contact surface to the vertices in the subdivided bounding box. As a property hash table we consider more than a single pair of collisions. Thus multiple soft bodies are detected for collision in a large simulation environment.

3. Analysis

First we analyze the method with regard to time complexity. Since the proposed method is using hash table, performance depends on subdivided bounding box distribution in the scene.

- Best Case: O(1). In this case every subdivided bounding box hashes to a different grid cell. No collision will be performed at all.
- Worst Case: $O(N^2)$. In this case, every subdivided bounding box in the scene is in the same grid cell. N is number of vertices.

The benefit of subdivided bounding box is to reduce cost of hash function. In stead of applying hash function to ***N vertices*** in the scene, the proposed method spend only $\boldsymbol{O \log_n N}$.

- Without subdivided bounding box: O(N). In this case every vertex has to be applied to hash function.
- With subdivided bounding box: $O(\log_n N)$. In this case every vertex doesn't have to apply to hash function. Only 8 points of subdivided bounding box have to be applied to hash function. N is the number of vertices and n is level of subdivision.

4. Experiments and Results

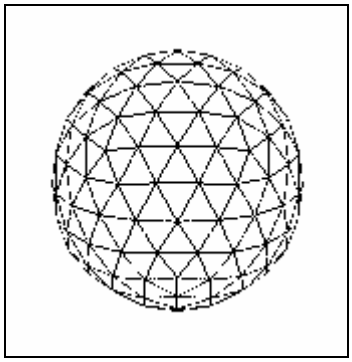


Figure 9: *structure of a soft body*

A simulation was developed using: Code (C++), Graphics (OpenGL), and Compiler (Microsoft Visual Studio). Figure 9 presents a soft body which is created from 162 vertices and 320 faces. We compare our proposed method to Grid Hash Function, and Bounding Box, Bounding Box and Grid Hash Function. Four different scenarios are created and called A, B, C, and D, in which all soft bodies are falling down and colliding with a sphere and ground surface shown in figures 10, 11, 12, and 13. There are 1k, 5k, 10k, and 15k faces in scenario A, B, C, and D respectively. For the hash table a bucket size of 101 is applied in scenario A and B, while a bucket size of 1001 is created in scenario C and D.

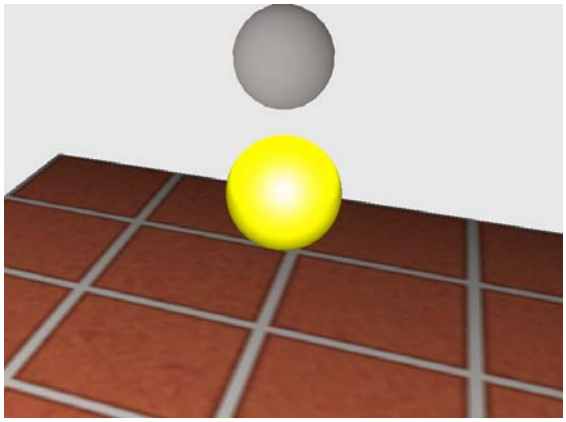


Figure 10: *Frame 100 showing a solid body sphere (below) and a falling soft body object (above)*

Initially in Figure 10 there are a solid sphere (below) and a soft body (above). Then the soft body collides with the sphere as shown in figure 11. Later on, there are more soft bodies falling down and colliding with other soft bodies presented in figure 12 and 13. They deform with vertices moving based on the laws of physics, according to detail in Maciej and Ollila, 2003. The simulation has been captured at frame 100, 400, 600, and 800.

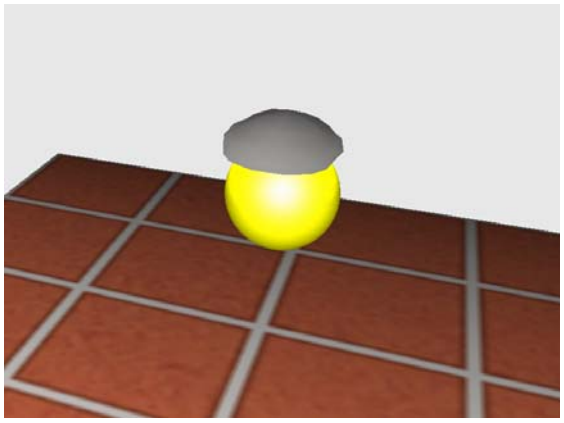


Figure 11 *At frame 400 a soft body object has collided with the solid body sphere*

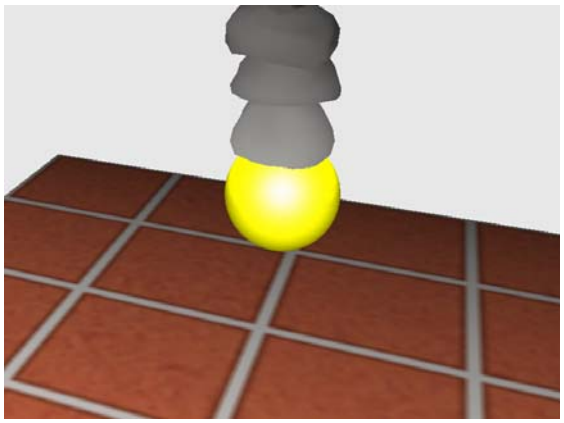


Figure 12: : *At frame 600 multiple soft body spheres have dropped upon each other, showing both soft-soft and soft-solid collision*

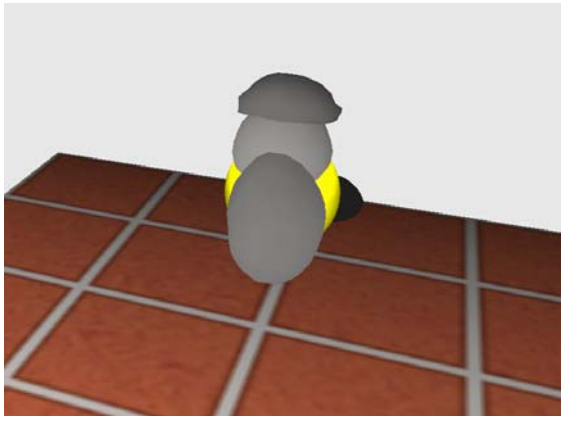


Figure 13: *At frame 800 gravity and friction eventually drop the soft body objects to the floor*

MODEL	Grid Hash Function (frame per second)	Bounding Box (frame per second)	Bounding Box and Grid Hash Function (frame per second)	Multi-level SB collide (frame per second)
A	20	22	40	70
B	18	16	35	57
C	14	13	31	49
D	13	14	25	41

Table 1: *The average running time (frame per second) in each model on a typical laptop: Pentium-4 3.2Ghz, 1Gb RAM, Nvidia GForce Go GPU.*

The experiment shows that using only a grid hash function yielded worst performance for simulations A, B, C, and D. The bounding box algorithm and combined bounding box with a grid hash function gave slightly better results. Multi-Level SB Collide gave the best performance, achieving significantly better frame rate in all four scenarios. The results of all experiments show that our algorithm can reduce time computation in comparison to the other methods.

5. Conclusions and Future Work

Multi-level SB collides presents the concept of tracking with Multi-level Subdivided Bounding box (Multi-level SB), box hash function (BHF), Self-collision and collision in contact surface (SCF and CF). The main strength of the multi-level SB collide lies in the efficiency in collision detection in soft bodies, only requiring multi-level bounding boxes and applying into box hash function. Our method essentially extends the existing approaches, multi-level subdivided bounding box and box hash function, with the concept of contact surfaces. All soft bodies are surrounded by a bounding box with maximum point and maximum point. Then, box hash table is defined by subdivided bounding box and box hash function. The vertices in same box hash list are, finally, observed for surface contact for self-collision and collision. Another benefit of this algorithm is that we can consider more than one pair of objects so that multiple, simultaneous soft body collisions are detected in a large environment of simulation. The result of this work performed with 10-15k faces showed that this algorithm is efficient for detecting collision for soft bodies in real-time considering the time spent for animation.

As for future work, it is possible to simplify the operation process to optimize the algorithm that has been implemented in this paper. Also, a tree structure could possibly be implemented to determine the area of collision in the object body and perform the collision detection in the overlapped area. However, a tradeoff would be introduced of additional time required to modify the tree structure in each time step.

6. References

Bergen, G.V.D. 1997. “Efficient Collision Detection of Complex Deformable Models Using AABB Trees.” *Journal of Graphics Tools 1997*, vol. 2, Issue 4 (Apr.): 1-13.

Bouma, W. and Vanecek, G. Jr. 1991 “Collision Detection and Analysis in a Physical Based Simulation.” *Eurographics Workshop on Animation and Simulation 1991* (Vienna) 191-203.

Cohen, J.D., Lin, M.C., Manocha, D., and Ponamgi, M. 1995. “I-COLLIDE: An Interactive and Exact Collision Detection System for Large-Scale Environments.” *Proceedings of the 1995 symposium on Interactive 3D graphics 1995* (Monterey, CA, United States) 189-196.

Erickson, J., Guibas, L.J., Stolfi, J., and Zhang, L. 1999 “Separation-Sentitive Collision Detection for Convex Objects.” *Proceedings of the tenth annual ACM-SIAM symposium on Discrete algorithms 1999,* Baltimore, Maryland , 327 – 336.

Ganovelli, F., Dingliana, J., and O’Sullivan, C. 2000. “Buckettree: Improving Collision Detection between Deformable Objects.” *In Spring Conference in Computer Graphics SCCG 2000* , Bratislava, 156-163.

Govindaraju, N.K., Redon, S., Lin, M.C., and Manocha, D. 2003. “CULLIDE: Interactive Collision Detection Between Complex Models in Large Environments using Graphics Hardware.” *Siggraph Eurographics Graphics Hardware 2003* (San Diego, CA, Jul. 26-27).

Gottschalk, S., Lin, M.C., and Manocha, D. 1996. “OBB Tree: A Hierarchical Structure for Rapid Interference Detection.” *Preceeding of ACM Siggraph 1996* (New Orleans, Louisiana, Aug. 4-6), 171-180.

He, T. 1999. “Fast Collision Detection Using QuOSPO Trees.” *Proceedings of the 1999 symposium on Interactive 3D graphics*, (Atlanta, Georgia, Apr. 26-29) , ACM 1999 55-62.

Held, M., Klosowski, J.T., Mitchell, J.S.B. 1995. “Evaluation of Collision Detection Methods for Virtual Reality Fly-Throughs.” *Proceedings Seventh Canadian Conference on Computational Geometry 1995*, 205–210.

Hubbard, P.M. 1995. “Collision Detection for Interactive Graphics Applications.” *IEEE Transactions on Visualizationand Computer Graphics 1995*, **1**(3):218–230.

James, D.L. and Pai, D.K. 2004. “BD-tree: Output-Sensitive Collision Detection for Reduced Deformable Models.” in *Proceedings of ACM SIGGRAPH 2004*, (Los Angeles, CA ,Aug 8-12).

Mesit, J.; Guha, R.;Hastings, E., 2004 "*Optimized Collision Detection For Flexible Objects*". International Conference on Computer Games: Artificial Intelligence, Design, and Education CGAIDE2004.

Klosowski, J.T., Held, M., Mitchell, J., Sowizral, H., and Zikan, K. 1998. **"**Efficient Collision Detection Using Bounding Volume Hierarchies of k-DOPs." *IEEE Transactions on Visualization and Computer Graphics*, vol4 issue 1(Jan.) : 21-36.

Larsson, T. and Akenine-Möller, T. 2001. "Collision Detection for Continuously Deforming Bodies" *In Eerographics 2001*., Manchester, UK, 325-333.

Moore, M. and Wilhelms, J. 1988. "Collision Detection and Response for Computer Animation" *In proceedings Computer Graphics SIGGRAPH 1988* , 22(4):289-298.

Maciej, M. and Ollila, M., '*Pressure Model of Soft Body Simulation*', SIGRAD2003, November 20-21, 2003

Naylor, B., Amatodes, J.A., Thibault, W. 1990. "Merging BSP Trees Yields Polyhedral Set Operations." *In proceedings Computer Graphics SIGGRAPH 1990* ,24(4):115–124, 1990.

Otaduy, M.A. and Lin, M.C. 2003. "CLODs: Dual Hierarchies for Multiresolution Collision Detection." *Proceedings of the Eurographics/ACM SIGGRAPH symposium on Geometry processing 2003* (Aachen, Germany, Jul. 27-31), 94-101.

Teschner, M., Heidelberger, B., Müller, M., Pomeranets, D., and Gross, M. 2003. "Optimized Spatial Hashing for Collision Detection of Deformable Objects." *Vision, Modeling, and Visualization 2003*. (Munich, Germany, Nov. 19-21).

BDI for Intelligent Agents in Computer Games

N.P.Davies and Q.H.Mehdi
Computer Games Centre
School of Computing and Information Technology
University of Wolverhampton
Wolverhampton, UK
E-mail: N.P.Davies2@wlv.ac.uk

KEYWORDS

Deliberative Agents, Computer Games, Artificial Intelligence.

ABSTRACT

With the emergence of complex computer games and advanced gaming hardware, possibilities for overcoming some of the deficiencies in traditional game AI are becoming feasible. These deficiencies include repetitive, predictable, and inhuman behaviour are caused by the reliance on simple reactive AI techniques. By using more sophisticated AI and agent techniques, we intend to overcome some of these problem areas. The aim of our research is to create new forms of intelligent characters (agents) that will exhibit human-like intelligence and provide more challenging and entertaining virtual opponents and team mates for computer games. We present here our prototype application that implements a BDI agent system within the 3D computer game Unreal Tournament via GameBots and JavaBots technology.

INTRODUCTION

With this continued popularity of computer games, game players are expecting new challenges with more sophisticated games and game playing experiences. Increases in processing power are now giving game developers opportunities to develop novel techniques to incorporate into their games. With graphics capabilities now reaching the point where game environments are becoming almost photorealistic, some of this power must become available for AI systems. Currently developers are looking for new and inventive ways to keep the game players entertained. The challenge is to produce artificial intelligence for computer games characters that can utilise the increased power afforded by improvements in games hardware, and make AI agents appear as human-like as possible so as to improve the game playing experience.

To produce agents capable of this behaviour in complex computer game environments we are using the Belief-Desire-Intention (BDI) model of agency (Bratman, 1987). In this model, agents are constructed using humanistic concepts such as goals to achieve, beliefs about the environment, and plans to achieve goals. Using BDI, we can simulate the decision making processes performed by humans, using the same information available to humans, in order to make the agent act in a human like way. It is expected that this will make computer game characters appear more realistic, and therefore, improve the experience of playing against artificial game characters. The paper is constructed as follows. In next section we outline the design of our system for integrating a BDI reasoning engine into the computer game Unreal Tournament, and detail the three layers of the system including deliberation, communication, and virtual environment. We follow this with our experimental results, including the initial implementation where agents interact with the game environment, exploring, attacking enemies, producing paths through the environment, following path, and building health by locating health packs. We conclude with our future aims, including the addition of a multi-agent layer for common goals amongst agents.

SYSTEM DESIGN

Our framework, (Figure 1), is constructed using three layers that integrate several tools available as open source projects and commercial applications. The layers are:

Intelligent Agent	:	Jadex
Game Environment	:	Unreal Tournament
Communication Layer	:	JavaBots/GameBots

Each layer is described in more detail below. It should be noted that our implementation is client-server based, and as such, requires an extra layer for external communication with the game engine. This means the agent is a combination of two separate entities. The first entity is situated within the Unreal Tournament game, and can be considered an avatar for our intelligent agent. The intelligent agent guides the avatar by sending commands over the network, and builds up a view of the environment by receiving perception messages. There are several reasons the system is implemented in this way; not least is the desire to implement our system in a modern commercial computer game. It is not possible to incorporate our AI directly within the game due to limitations in access to the engines source code. However, this architecture has the benefit of forcing the

agent to play the game in the same way as human players; based on sensor information. In addition, this type of architecture allows experimentation with human / agent teams in an extensible framework, and allows many agents to connect to the game server simultaneously.

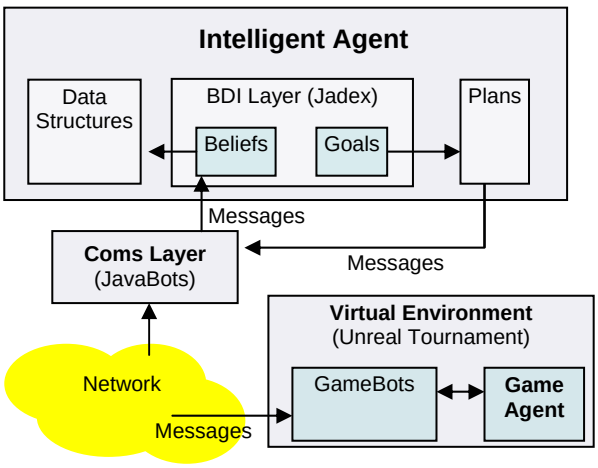


Figure 1: System Design

Intelligent Agent

The intelligent agent layer consists of a reasoning system, a knowledge base of plans, and data structures for storing environment information. The reasoning system is developed using Jadex (Braubach et al, 2004). This is an agent platform that allows the creation of BDI agents in the Java programming environment i.e. it allows the creation of agents that use the mental attitudes of belief, desire, and intention to model human like reasoning processes. Agents are created via the specification of beliefs, goals, plans, events, and capabilities. Goals relate to the state an agent would like to achieve and can be of several types; achieve, maintenance, and perform. Achieve goals are used to perform an action, such as move to a waypoint. This type of goal can either succeed or fail. Maintain goals are used to monitor the agent, e.g. make sure health stays above 50. If the agent's health drops below this level, then some action is triggered to rectify the situation. Perform goals are used to perform actions that are consistent with a state that don't have a target state to reach e.g. 'explore' where an agent will continue to search an environment until some other event causes it to change behaviour. To accomplish goals, relevant plans are used. Plans are created via extending an abstract Plan class that allows messages to be sent between plans and agents. Agents are defined in an XML based Agent Definition File (ADF) were beliefs, goals and plans are linked by specifying their applicability via trigger statements. There may be many plans applicable to specific goals, therefore the plans applicability can be further reduced with clarifiers such as context conditions, that state that plans can only be created if certain belief conditions are met. Beliefs specify agents' knowledge of the world, and are used to trigger goals, and plans success or failure. Beliefs can be any Java object. In our system, we have created belief objects that store the agent's status, enemy locations, navigation info etc. Using this system, we have developed AI agents that have the ability to respond to environmental events, identify appropriate plans to handle the events, and execute those plans in a timely manner. While executing plans, the agents monitor the environment, and internal belief structures, in order to ensure plans are still relevant, and identify new opportunities.

Virtual Environment

Our intelligent agent connects to a game environment, and interacts with it through perceptions and performing actions. We have chosen to develop our system using the game engine Unreal Tournament, a three-dimensional, networked FPS computer game. The game includes several game types. These include Death Match; a free-for-all match with the winner decided by the highest number of frags achieved over a certain time period, Domination; where players compete to capture and defend domination points for a specified amount of time, Capture the flag; where players have to retrieve the opposition's flag and bring it back to their base, while defending their own flag and Team Death Match; where teams work together to achieve the highest frag rate in a set time period. The game can be modified in several ways via an editor and a scripting language. New levels can be created in the graphical designer Unreal Edit. Game rules and physics can be modified via the scripting language Unreal Script. Lewis and Jacobson (2002) point out the benefits of this solution. The engine is inexpensive. The graphics rendering capability is superior to anything that can feasibly be created by small research groups. Also, the game logic is fully implemented i.e. the game comes complete with standard functions such as collision detection, physics systems, game maintenance etc. Other benefits include the large user base of players of the game. This gives access to domain experts for knowledge elicitation, and groups for evaluation of the completed system. The use of a game engine is not an ideal solution however. It would be better to develop a complete computer game where all functions are accessible at a source code level; a level that is unavailable through the use of game engines. Game engines impose limitations that are created by the game engine developers, which can cause problems, but is a neat solution to our requirements.

Communication Layer

To facilitate communication between the intelligent agent and the game world we have adopted the use of the dual middleware product of GameBots/JavaBots (Marshall et al, 2006). GameBots is an extension to Unreal Tournament that resides upon the game server. It is written in the scripting language provided by the Unreal developers; Unreal Script, and extends the basic AI and networking components shipped with the game.

GameBots allows external processes to access internal game AI functions through a network socket connection via the exchange of messages. For every game loop, agent perception data is sent as a synchronous message packet across the network to a JavaBots client. This information consists of currently visible navigation points, inventory items, and other visible agents. Status information is also sent including the agent's health, current location, current weapon etc. Event messages, such collision information, damage reports etc, are sent whenever they occur in the game via asynchronous messages. The communication is a two-way process, and GameBots also receives messages from the JavaBots client that consist of actions for the agent to perform. Actions include rotate, walk, run, shoot etc, and also more complex operations such as find path, which queries the navigation system of Unreal Tournament, and sends a list of navigation nodes back in the form of an asynchronous message. The client portion used by the intelligent agent is called JavaBots; a Java based system developed to connect to GameBots. It is an extensible API that contains example bots, visualisation applications, and a Bot Runner application that allows agents to be connected and visualised via a GUI interface. We have taken the original JavaBots project and removed the extra functionality to produce a very simple API that simply listens for messages, and sends instructions back to Unreal Tournament. We have removed the sample bots and Bot Runner classes, and instead use functions within Jadex to maintain the connection to the game.

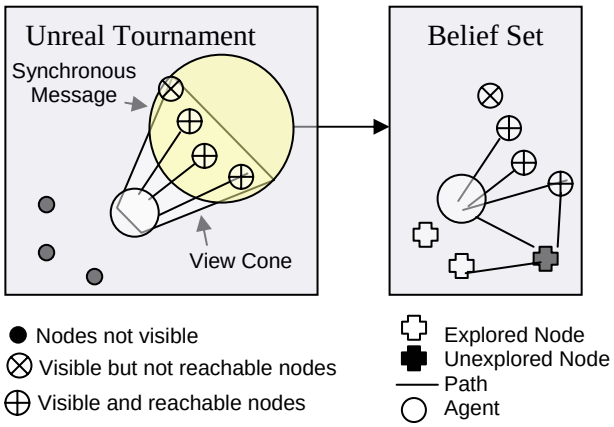


Figure 2: Navigation messaging system

Figure 2 shows an example of the messaging system in which GameBots sends a message block containing navigation point information to the intelligent agent, where it is recorded in a belief structure. In Unreal Tournament, the game agent has a set view cone of around 45 degrees. At any point in the game, the game agent is capable of observing a discrete portion of the game environment contained within this view cone, which is not occluded by walls. Figure 3 shows a typical Unreal Tournament view, the game agent can see inventory items (Heath Packs), and waypoint nodes. The waypoint nodes are either reachable, or unreachable. In the illustration, there is a gap between the game agent, and a platform containing the health packs; it can therefore see them, but cannot reach them directly. The navigation node to the right of the illustration is visible and reachable directly. Therefore, the agent can run directly to it. This information is grouped into a single message block, and sent to the intelligent BDI agent. The intelligent agent receives this message, parses it, and populates its belief sets. New nodes (nodes an agent has not seen before) are added to the list of known nodes. Nodes at the position of the agent are marked as 'visited' to indicate the agent has explored the position. Two other data sets are populated; visible nodes and reachable nodes. At each frame, these sets are cleared, and populated with the new data contained in the message i.e. only nodes that the agent can currently see are stored.

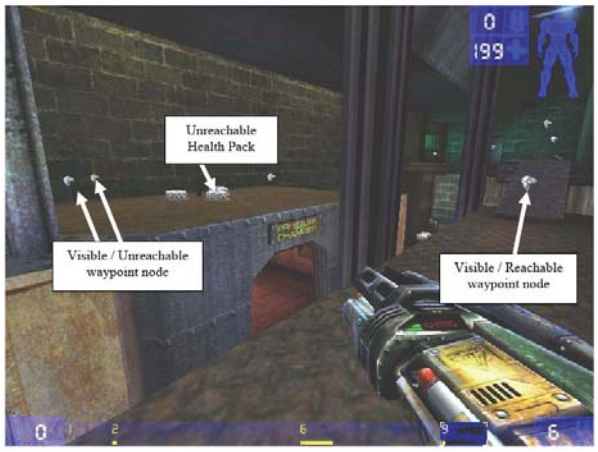


Figure 3: Navigation messaging system

EXPERIMENTAL RESULTS

A prototype application has been developed that shows the potential of the architecture. The intelligent agents are able to connect to Unreal Tournament, build a 3D view of the environment, and navigate the world. The agents are also capable of some basic behaviour in the game. This includes exploring, navigating, hunting and escaping. The overriding goal of the agent is to maintain health above 50. Once health drops below this level, the agent attempts to disengage from a combat situation, and find health until its health has reached 90 or above. This behaviour is achieved via a maintenance goal that inhibits the 'explore' and 'attack' goals. When the agent is attempting to build its health, it will observe the environment to see if any health packs are currently visible and reachable. If it can see a health pack, it will move towards it and pick it up. If there are no visible health packs, it checks its memory to recall the location of the nearest health pack to its current location. At this point, it queries Unreal Tournament to find a path to the identified health pack from it current location, and then follow this path. If, on route, it spots another health pack, it will temporarily drop the goal of following the path in favour of collecting the new health pack. It will then

resume the goal of following the path (assuming it still requires health). If the agent encounters an enemy agent while following the path, it will retreat, and choose a path to an alternate health pack. Following paths and collecting health behaviour is created using achieving goals. At each stage of the goal, the agent can either succeed or fail. If a section of the plan fails, the agent is capable of retrying the goal, or dropping the goal and starting again at any stage. If the agent health is adequate, the agent will revert to the default behaviour of exploring the environment. It will query its belief set to find a list of reachable nodes, and check if any of them have not been explored before. If it finds an unexplored node, it will move towards it. If it cannot see an unexplored node, it will run to a random reachable node. This behaviour is created using a perform goal, and the agent will continue with this behaviour until some event causes it to drop the behaviour. An example of a condition where the agent will drop the goal is if the agent spots an enemy. When an enemy is spotted, the agent will engage in an attacking behaviour that includes firing its weapon, running towards the enemy, and jumping left and right until it has either killed the enemy, its health drops to a point where the maintenance condition forces it to try to escape, or it is killed. The behaviour of the agent is presently for illustrative purposes, and is not intended as a sophisticated behaviour system. However, it does prove that developing a more complex agent is possible with a combination of goals and plans within the BDI framework.

CONCLUSIONS AND FUTURE WORK

It has been proposed that the goal based, deliberative architecture, BDI, has the potential to produce more human like behaviour in computer game characters that will exhibit more realistic behaviour than can be achieved with simple reactive AI techniques alone. We have identified a set of tools, and implemented a prototype application that incorporates the commercial computer game Unreal Tournament, the reasoning layer BDI through Jadex, and linked the systems using the communication system GameBots/JavaBots. We have created a set of agents that perceive their environment via sensory information gathered from the game, and use this information to build up a beliefs base. Using these beliefs, and a set of desired conditions, the agent uses its plan base to bring about beneficial states of affairs. The implementation currently allows the agent to explore the environment, attack enemies, find health packs, and navigate the game map by following a path list. In the next stage of development, we will expand upon this basic bot, and add more sophisticated behaviour and tactics. We will also implement team based behaviours, which will require agent negotiation techniques and social abilities. In addition, we will expand upon the BDI framework, and incorporate a layered architecture where fast, reactive behaviour can be accomplished in lower levels, and high level cooperative goals can be accomplished at higher levels.

REFERENCES

Agent Oriented Software Pty. Ltd. JACK Intelligent Agents. (2006) http://www.agent-software.com

Bratman, M. (1987) *"Intention, Plans, and Practical Reason"* Harvard University Press: Cambridge, USA

Braubach, L., Pokahr, A., Lamersdorf, W., Krempels, K., Woelk, P. (2004) *"A Generic Simulation Service for Distributed Multi-Agent Systems"*, in: From Agent Theory to Agent Implementation (AT2A1-4) 2004

Epic Games Inc. Unreal Tournament. http://www.unreal.com

Lewis, M., Jacobson, J., (2002), *'Game Engines in Scientific Research'*, Communications of the ACM, Volume 45, Issue 1. New York, USA

Marshall, A. N., Gamard, S., Kaminka, G. J., Manojlovich, Tejada S., Gamebots, viewed 10th Mar 2006, http://planetunreal.com/gamebots/

Norling, E. (2004) *'Folk psychology for human modelling: extending the BDI paradigm'*. In : Int. Conf. on Autonomous Agents and Multi Agent Systems (AAMAS), New York, 2004.

Rao, A., Georgeff, M., (1995) *'BDI agents: from theory to practice'* Tech. Rep. 56, Australian Artificial Intelligence Institute, Melbourne, Australia

Russell, S., Norvig, P., Artificial Intelligence: A Modern Approach. Prentice-Hall, Englewood Cliffs, New Jersey, 1995.

Sellars, W. (1956). ''Empiricism and the Philosophy of mind,'' in H. Feigl & M. Scriven, eds., *Minnesota Studies in the Philosophy of Science*, 1, Minneapolis: University of Minnesota Press.

Stich, S. & Ravenscroft, I. (1994): "What is Folk Psychology?". *Cognition* 50: 447-68

Valve Software (2004) Half Life 2, http://www.half-life.com/

Valdes, R.: In the Mind of the Enemy: The Artificial Intelligence of Halo 2 (2004): http://stuffo.howstuffworks.com/halo2-ai.htm

Welsh, T., (2005) *'A Typology of Givaways: An Evaluation of FPS Bots'* in Proceedings of CGAIMS'2005 6th International Conference on Computer Games: Artificial Intelligence and Mobile Systems 27-30 July 2005

AUTHOR BIOGRAPHIES

NICHOLAS P. DAVIES was born in Wolverhampton, UK, where he studied Computer Science at the University of Wolverhampton, and obtained a First Class Degree in 2003. He is currently researching AI and Computer Games, and has completed the first two years of his PhD.

List of Authors — Page No.