12TH INTERNATIONAL CONFERENCE

ON

INTELLIGENT GAMES AND SIMULATION

GAME-ON® 2011

EDITED BY

Colm O'Riordan

Sam Redfern

Valerie Butler

Fergal Costello

AUGUST 22-24, 2011

NUI

Galway

IRELAND

A Publication of EUROSIS-ETI

12^{TH} International Conference
on
Intelligent Games and Simulation

GALWAY, IRELAND

AUGUST 22 - 24, 2011

Organised by

ETI

Sponsored by

EUROSIS

Co-Sponsored by

Binary Illusions

BITE

Delft University of Technology

Ghent University

Higher Technological Institute

National University of Ireland

Larian Studios

University of Skövde

Hosted by

National University of Ireland

Galway, Ireland

INTERNATIONAL PROGRAMME COMMITTEE

Learning & Adaptation
Christian Bauckage, University of Bonn, Sankt Augustin, Germany
Christos Bouras, University of Patras, Patras, Greece
Adriano Joaquim de Oliveira Cruz, Univ. Federal de Rio de Janeiro, Rio de Janeiro, Brazil
Chris Darken, The MOVES Institute, Naval Postgraduate School, Monterey, USA
Andrzej Dzielinski, Warsaw University of Technology, Warsaw, Poland
Maja Pivec, FH JOANNEUM, University of Applied Sciences, Graz, Austria
Tina Wilson, The Open University, Milton Keynes, United Kingdom

Intelligent/Knowledgeable Agents
Nick Hawes, University of Birmingham, United Kingdom
Wenji Mao, Chinese Academy of Sciences, Beijing, China P.R.
Marco Remondino, University of Turin, Turin, Italy

Collaboration & Multi-agent Systems
Victor Bassilious, University of Abertay, Dundee, United Kingdom
Sophie Chabridon, Groupe des Ecoles de Telecommunications, Paris, France

Opponent Modelling
Pieter Spronck, University of Maastricht, Maastricht, The Netherlands
Ingo Steinhauser, Binary Illusions, Braunschweig, Germany
Andrew Ware, University of Glamorgan, Pontypridd, United Kingdom

Peripheral

Psychology, Affective Computing and Emotional Gaming

Myriam Abramson, US Naval Research Laboratory, USA
Gianna Cassidy, eMotion-Lab, Glasgow, United Kingdom
David Farrell, eMotion-Lab, Glasgow, United Kingdom
Eva Hudlicka, Psychometrix Associates, Blacksburg, USA
Romana Khan, eMotion-Lab, Glasgow, United Kingdom
Brian McDonald, eMotion-Lab, Glasgow, United Kingdom
David Moffat, eMotion-Lab, Glasgow Caledonian University, United Kingdom
Jon Sykes, eMotion-Lab, Glasgow Caledonian University, United Kingdom
Thomas Welsh, eMotion-Lab, Glasgow, United Kingdom

Artistic input to game and character design

Anton Eliens, Vrije Universiteit Amsterdam, Amsterdam, The Netherlands
Olli Leino, IT-University of Copenhagen, Copenhagen, Denmark
Sean Pickersgill, University of South Australia, Adelaide, Australia
Richard Wages, Nomads Lab, Koln, Germany

Storytelling and Natural Language Processing

Jenny Brusk, Gotland University College, Gotland, Sweden
Ruck Thawonmas, Ritsumeikan University, Kusatsu, Shiga, Japan
Clark Verbrugge, McGill University, Montreal, Canada

Online Gaming and Security Issues in Online Gaming
Marco Furini, University of Modena and Reggio Emiliano, Modena, Italy
Pal Halvorsen, University of Oslo, Oslo, Norway
Fredrick Japhet Mtenzi, School of Computing, Dublin, Ireland
Andreas Petlund, University of Oslo, Oslo, Norway
Jouni Smed, University of Turku, Turku, Finland
Knut-Helge Vik, University of Oslo, Oslo, Norway

INTERNATIONAL PROGRAMME COMMITTEE

MMOG's

Michael J. Katchabaw, The University of Western Ontario, London, Canada
Jens Mueller-Iden, University of Munster, Munster, Germany
Alice Leung, BBN Technologies, Cambridge, USA
Mike Zyda, USC Viterbi School of Engineering, Marina del Rey, USA

Serious Gaming

Wargaming Aerospace Simulations, Board Games etc....
Roberto Beauclair, Institute for Pure and Applied Maths., Rio de Janeiro, Brazil
Tony Manninen, Ludocraft Ltd., Oulu, Finland
Jaap van den Herik, Tilburg University, Tilburg, The Netherlands

Games for training
Michael J. Katchabaw, The University of Western Ontario, London, Canada
Jens Mueller-Iden, Universitaet Muenster, Muenster, Germany
Roger Smith, US Army, Orlando, USA

Games Applications in Education, Government, Health, Corporate, First Responders and Science
Russell Shilling, Office of Naval Research, Arlington VA, USA

Games Interfaces - Playing outside the Box

Games Console Design
Chris Joslin, Carleton University, Ottawa, Canada

GAME-ON®
2011

All articles published in these Proceedings have been peer reviewed

EUROSIS-ETI Publications are ISI-Thomson and INSPEC referenced

A CIP Catalogue record for this book is available from the Royal Library of Belgium under nr.12620

For permission to publish a complete paper write EUROSIS, c/o Philippe Geril, ETI Executive Director, Greenbridge NV, Wetenschapspark 1, Plassendale 1, B-8400 Ostend, Belgium.

EUROSIS is a Division of ETI Bvba, The European Technology Institute, Torhoutsesteenweg 162, Box 4, B-8400 Ostend, Belgium

Printed in Belgium by Reproduct NV, Ghent, Belgium
Cover Design by Grafisch Bedrijf Lammaing, Ostend, Belgium
Cover Pictures by Larian Studios, Ghent, Belgium

GAMEON® is a registered trademark of the **European Technology Institute** under nr: 1061384-761314

EUROSIS-ETI Publication

ISBN: 978-9077381-64-9
EAN: 978-9077381-64-9

PREFACE

Dear participants,

On behalf of all the people and institutions that made this conference possible, I wish to welcome you all here to NUI, Galway, Ireland for the 12th edition of the Annual Conference on Simulation and AI in Games. (GAMEON®'2011)

In addition to the interesting and varied submitted papers, we are very grateful to our two invited keynote speakers - Andre Gagalowicz and Nathan Griffiths for their talks on 3D face tracking and trust In multi-agent systems.

I wish to thank everyone who has contributed his or her time and effort in organising this conference. This includes all the authors who prepared and submitted papers and the international Programme Committee members who were involved in the review process.

I wish to also acknowledge the huge effort and contributions of Philippe Geril who is responsible for organising this conference this year and over the past couple of years.

Furthermore, I would like to express my gratitude to our industry visitors for dedicating their time and for sharing their expertise.

I wish to also acknowledge our sponsors for supporting the conference. I would like to thank the Discipline of Information Technology and the University Millennium Fund for financial support.

Finally, I hope you enjoy your stay here in Galway.

Colm O'Riordan

General Conference Chair GAMEON'2011
National University of Ireland
Galway, Ireland

CONTENTS

CONTENTS

SCIENTIFIC PROGRAMME

GAME DESIGN

PRESENCE IN COMPUTER GAMES: DESIGN REQUIREMENTS

Barbaros Bostan
Yeditepe University
Kayisdagi Cad. 26 Agustos Campus

Atasehir, Istanbul / Turkey, 34755
E-mail: bbostan@yeditepe.edu.tr

Sertac Ogut
Marmara University
Faculty of Communication, Nisantasi Campus
Sisli, Istanbul / Turkey, 34365
E-mail: sertacogut@marun.edu.tr

INTRODUCTION

Presence or the sense of 'being there' is an important and critical concept of computer gaming which relies on several factors. When we look at presence from a bottom-up approach to define its components, it is important to identify the design requirements for virtual environments and their effects on presence (Bostan, 2009). It is also imperative to define how much each requirement correlates with presence and how designers should address these. This study aims to solve these issues by using a presence questionnaire based on virtual environment design requirements defined by Stuart (2001).

Data collected from a survey study on a computer role playing game (RPG), which is available for 3663 participants, is used to identify the relationship between these design requirements and presence. Among the selected requirements; sociability, veridicality, autonomy and physics of the virtual world have the highest correlations with presence respectively; and interactivity is a separately analyzed requirement that has special focus on it. Results of the study indicated that, storyline, NPC (non-player character) characteristics and communication with these virtual characters are the most important factors that influence interactivity.

PRESENCE AND DESIGN REQUIREMENTS

Presence, which is defined as the subjective experience of being there, is a construct with various dimensions. In order to define the determinants of presence, Insko (2003) defined three categories of methods commonly used for measuring presence: Subjective measures, behavioral measures and physiological measures. Subjective measures rely on user responses obtained from questionnaires. Various conceptual studies on the nature of presence have been conducted by researchers (Sheridan, 1992; Held & Durlach, 1992) and several of them have used questionnaires to shed light on the dimensions of presence. Lombard & Ditton (1997) defined six interrelated but distinct dimensions of presence: Social richness, realism, transportation, immersion, social actor within medium and medium as social actor. Witmer & Singer (1998) presented 4 factors that contributed to a sense of presence: Control factors, sensory factors, distraction factors and realism factors. Regenbrecht (1998) defined a 3-factor solution for presence: Spatial presence, involvement and realness. Lessiter (2001) extracted 4 factors that influence presence: Physical presence, engagement, naturalness and negative effects.

These studies focus on the hypothesized factors that influence presence in virtual environments. Virtual environment design, however, relies on several requirements. When we look at presence from a requirements perspective, it is important to identify the design requirements for virtual environments and their effect on presence. This study aims to solve these issues by using a presence questionnaire based on virtual environment design requirements defined by Stuart (2001). Participants of this study are computer game players. According to Steinkuehler (2006), computer games are a productive context for research about cognition and culture in a world that is increasingly globalized and networked. According to Ondrejka (2006), since computer game players spend significant portions of their lives immersed within virtual worlds, computer games provide virtually limitless opportunities for research and study. The computer game selected for this study is a role playing game (RPG).

A RPG is an interactive story where the game player controls an avatar called a player character (PC). The ancestors of RPGs are MUDs, which are text based fantasy worlds that were very popular in the past. Over time, MUDs evolved into standalone RPGs and MMORPGs. Since Towell & Towell (1997) stated that research on MUDs may be helpful in understanding the contribution to presence by social interaction in other virtual environments, studies on RPGs may also provide new insights on the concept of presence. Supported by the story, settings and combat systems of tabletop role-playing games, computer RPGs provide interesting interactivity and openness opportunities to players. Selected RPG, Elder Scrolls IV: Oblivion, is an outstanding game of the genre, combining real world physics with 1500 AI supported virtual characters that have a 24- hour schedule of their own.

Stuart (2001) defined 22 functional requirements for designing virtual environments. All these requirements are of varying degree of importance depending upon the application in question. Stuart's framework is applicable to virtual environments and computer games since it explains important parameters for virtual environment design, but it fails to cover some important characteristics of virtual environments which require special attention and further explanation.

The three I's of virtual reality defined by Heim (1998): Information Density, Interactivity and Immersion, are crucial parameters for virtual environment designers. Information intensity, similar to the requirement of resolution in Stuart's framework, represents the level of detail resulting from the continuous information transfer from the virtual world.

Interactivity is defined by Rafaeli (1988) as an underdefined concept that has little consensus on its meaning, but researchers stated that interaction and interactivity have an important role in creating a sense of presence (Zahorik & Jenison, 1998 ; Sheridan 1992). According to Laurel (1993), interactivity is characterized with three variables: frequency, range and significance of interactions. According to Friedl (2003), interactivity in computer games has three dimensions: player-to-player, player-to-computer and player-to-game interactions. Player-to-player interaction is unique for multiplayer games.

Immersion is defined by Slater & Wilbur (1997) as a VR system's ability to deliver a surrounding environment, capable of shutting down the sensations from the real world. According to Ermi & Mäyrä (2005), immersion in computer games has three dimensions. Sensory immersion is related to audiovisual properties of the virtual world, challenge-based immersion is related with mental skills such as strategic thinking or logical problem solving, and imaginative immersion is related with the storyline and virtual characters. Comments of a 48 year old gamer obtained by this study can better emphasize the degree of mental immersion experienced by an Oblivion fan: " *I suffer from an immune disease that causes a considerable amount of pain. In the evenings after work, I use Oblivion to detach my mind from the pain and in turn do not require medication while immersed in the game environment.*"

Since the requirements defined by Stuart do not need to be addressed in every application and system, they are classified into two groups: General requirements and special requirements. General requirements are mandatory for every virtual environment and special requirements are optional considerations for the designer. Requirements of Stuart's framework are given below. Only those with an asterisk are taken into consideration for this study. Discussing the specifics of each variable is beyond the scope of this article, but those interested in them can consult (Bostan, 2009) who discussed these requirements in relation to presence.

Table 1: Design Requirements of Stuart (2001)

General requirements	Special Requirements
Interactivity *	Responsiveness
Sociability *	Stability
Veridicality *	Robustness
Presence *	Registration
Resolution *	Calibration and Customization
Representation of User *	
Navigation Techniques *	Reconfigurability
Physics of the Virtual World *	Degree of Virtuality
Viewpoint	Connectivity *
Autonomy *	I/O Bandwidth
Locus of Control *	Safety and Hygiene
Multisensory Requirements	Choice of Representation

RESEARCH FINDINGS

The presence questionnaire developed for this study consists of 10 questions, one of which measures presence as the core concept of this research and the other 9 questions are used to measure the relevancy of selected application requirements with presence. According to Slater (1999), in order to study the factors that influence presence and their relationships with it, presence questionnaires shall include a direct question about presence. Otherwise, questions will give no information about the influence of variables on presence.

The questionnaire, which uses a 5-point likert scale, was posted on the Web. Messages about this study were posted on 13 forums and administrators of various Oblivion fan sites are contacted via e-mail. Several websites announced the questionnaire in their websites, requesting their visitors to participate in this study. The questionnaire was online for 23 days, after which it gets a total of 8065 views and 258 posts in 13 forums. At the end of 23 days, data are available for 3663 participants. Web server statistics show that 6256 people visited the web page of the questionnaire, indicating a 58% response rate.

Table 2: Forum Statistics

Forum Website	Views	Posts
Official Forums	4981	118
Cyrodiil Forums	337	19
Planet Elder Scrolls	962	43
Elder Scrolls (co.uk)	58	3
Dark Brotherhood	313	32
FileFront	221	4
Oblivion Files	72	3
Rough Guide to Cyrodiil	78	4
Elderscrolls Oblivion	242	9
Canadian Ice	153	9
Blood and Shadows	93	4
Gaming Source	305	4
RPGDot	250	6
TOTAL	8065	258

Demographic Variables

When we take a look at the demographic variables, we see that approximately 5% of the respondents is female, 95% male. The mean age was 24.2, ranging from 10 to 71 years. 19.5% of the respondents are married, 80.5% single. Education levels are, 32.2% high school or below, 31.8% some college or vocational school, 24.3% bachelors degree and 11.7% with a graduate degree.

One-way ANOVA test performed on demographic variables show that age and educational level differences between groups of participants are significant in terms of presence. Levene's test is used as a post-hoc test to validate the assumption of homogenity of variances between groups. This assumption is not violated since Levene's test is insignificant for age (p=.227) and education level (p=.544). ANOVA is significant for age (p<.001) and education levels (p<.001), showing that degree of presence decreases as age and education levels increase.

When we analyze correlations, significant Pearson correlation coefficient (r= -.064, p (two-tailed)<.001) shows that there is a significant negative relationship between age and presence. Nonparametric correlation tests, Spearman correlation (rs = -.104, p (two tailed) < .001) and Kendall's tau (τ = -.078, p (two tailed) <.001), are also significant. So, presence decreases with age. Chi-square test between presence and education level is also significant (p<.001) indicating that presence decreases with education level.

Correlation of Design Requirements

In order to define the relationships between the design requirements and presence, parametric and non-parametric correlations of these variables are calculated. Pearson correlation is a parametric statistic with an underlying assumption of normality. When linear correlations are not strong enough, non-parametric correlations give more meaningful but less powerful results. Given below is the correlation table showing that, 8 of the 9 requirements are significantly correlated with presence, confirming their relationships. Among these, sociability, autonomy, veridicality and physics of the virtual world, are the most influencing requirements respectively.

Table 3: Correlation of Design Requirements with Presence

Design Requirement	**Pearson (r)**	**Kendall's tau (τ)**	**Spearman's rho (r_s)**
Sociability	.466 **	.385 **	.457 **
Veridicality	.348 **	.289 **	.344 **
Autonomy	.294 **	.240 **	.286 **
Physics of the Virtual World	.226 **	.188 **	.222 **
Representation of the User	.140 **	.108 **	.125 **
Connectivity	.123 **	.095 **	.116 **
Resolution	.065 **	.066 **	.077 **
Navigation Techniques	.053 **	.044 **	.053 **
Locus of Control			

** p (two tailed) < .001

Determinants of Interactivity

In this study, interactivity is not a direct question and is assumed to be a complex combination of the design requirements defined by Stuart (2001). In order to define the determinants of interactivity, a comment box is included in the questionnaire, which is not compulsory for respondents. Users were requested to identify the factors that will make this virtual environment more interactive. 942 respondents leave their comments willingly. User defined factors are subject to a frequency analysis and the results are given below.

Table 4: User Defined Determinants of Interactivity

19,8%	NPC Communication
19,3%	NPC Characteristics
19,1%	Storyline
13%	Artificial Intelligence
12,6%	Physics of the VE
9,6%	Object Design
9,1%	Small Scale Connectivity
7,7%	Nature Design
7%	World Design
6,4%	Guilds & Factions
5,7%	No Level Scaling
4,6%	Autonomy
4,3%	No Large Scale Connectivity
4,3%	Destructible Environments
4%	User Interface & Navigation Design
3,8%	3D Animations
3,7%	Large Scale Connectivity
3,2%	Companionship
3,2%	Combat Design
2,3%	Responsiveness
2,3%	Reputation
2%	World Economy

According to users, three most important factors for increasing interactivity are: NPC communication, NPC characteristics and storyline. NPC communication consists of facial expressions, non-verbal and verbal communication with the NPCs. These three user defined elements of communication are also components of the 'Rich Interaction Model' for virtual environments defined by Manninen (2003). According to user comments, designers should also implement more voice actors for the NPCs and avoid repetitive dialogue options. Given below is a gamer comment on voice acting, showing us the complexity of NPC communication.

"*One of the most immersion-breaking parts of Oblivion was the terrible voice acting. Unlike Morrowind, many voice actors were re-used across races (Orcs and Nords, for instance), and voice actors that I came to strongly associate with certain races were reused improperly or not at all--for instance, the old Orc voice entirely disappeared, as far as I can hear, replaced very obviously with the Nord voice actor. Also, the performance given by the new voice actors was, I felt, not up to the level of that of the performances they gave for Morrowind. The delta between the old and new voices, both in usage and in quality of acting, was so extremely jarring--especially with the massively increased amount of voice in the game--that nearly every NPC to whom I talk breaks my immersion to some degree.*"

NPC characteristics primarily consists of depth and personality in non-player characters (NPCs). Synthetic characters must be responsive to their physical interactions with the environment, their aims, their knowledge of the virtual world, their personality and their interactions with human players (Magerko et al., 2004). According to player comments, designers should create NPCs from all ages and goal-driven non-player characters. Given below is one of the many gamer criticisms obtained by this study on character personality and autonomy in Oblivion:

"*The key failing: Context sensitivity. NPCs, superficially, act in lifelike manners. However, they methods of reacting to the environments are limited to direct interaction with objects they are programmed to interact with; they sit on furniture, sleep in beds, eat food, and talk with other NPCs or the player. This seems realistic, until more exotic situations are presented; these behaviors are not changed whatsoever by numerous factors that would impact the behavior of real people, such as weather, crime, etc. As an example, when nearby an open Oblivion gate, one would reasonably expect nearby people to be responding to it, possibly with emotions such as panic, fear, or perhaps even curiosity. The game's NPCs have no different reactions. Likewise, NPCs are not reactive to the events that take place around them; major game-related events, such as the completion of a quest, may alter what dialogue they have available, but their behavior is unaffected. As an example, if the player enters a busy city street, and one NPC suddenly attacks the player, the NPCs may respond to the attack by aiding the player, but after the battle has concluded, the NPC's behavior, and even dialogue, is not affected. This would appear to be a keystone in the elements of Artificial Intelligence that is lacking in the game of Oblivion, that would've added the realism level sufficient to truly make the NPCs seem as lifelike characters, rather than as flat 'simulation bots.'*"

User defined components of storyline are meaningful play and user freedom of choice. Storyline is closely related to the description of plot given by Slater and Wilbur (1997) and freedom in the users' actions within a virtual environment is also highlighted by Mantovani & Riva (1999). According to Salen & Zimmerman (2004), user freedom of choice is an important requirement of game design and meaningful play is the relationship between player's actions and system outcome. Users also commented that they would like to see non-linear quests, consequences for their actions and moral choices in the gaming environment. Comments of gamers obtained by this study shed light on different dimensions of freedom. Given below is an example comment:

"*I feel that in Oblivion's current form it offers much freedom of action (e.g. you may approach a problem as you wish, by using stealth magic or diplomacy) but not much freedom of morality and few consequences for moral choices. I would like oblivion and games in general to make me care and understand how I alter the world and why I should be careful about my morality.*"

CONCLUSION

The extent of this paper is to indicate a number of variables that need to be considered in order to maximize presence in virtual environments, especially in computer games. This study does not claim to have identified all of the design requirements that affect presence but it addresses how certain requirements shall be addressed by designers. Research findings show that sociability has the highest correlation with presence. Computer games, regardless of their multiplayer capabilities, should be capable of creating social virtual environments. Since Heeter (1992) defined sociability as one of the three dimensions of presence, game designers should support social virtual characters that build communities and groups. Oblivion is a single player computer game and sociability primarily consists of PC-NPC interactions. Players commented that sociable non-player characters shall have entertaining and non-repetitive dialogue options, believable behaviors, and near-realistic 3D appearance and animations. Players also indicated that guilds, NPC companions and character reputation are important social characteristics.

The virtual environment, with its physical appearance and object behaviors, should accurately represent the real world we are living in. Thus, veridicality is the consistency of information with the objective world and is one of the hypothesized realism factors that contribute to a sense of presence (Witmer & Singer, 1998). In this sense, players indicated that object design, world design, nature design and destructible environments are important characteristics of veridicality. According to Sutcliffe (2003), user interaction with the virtual world objects should conform to the laws of physics that constrain real-world interaction, yet players commented that real-world physics enhances the sense of presence experienced but without touch and force feedback, too much realism in world physics breaks the sense of presence.

Autonomy is the third best requirement correlating with presence. To sustain the feeling of presence in a virtual world, virtual agents should be able to make autonomous decisions independent of other entities in the environment and behave like real persons (Aylett & Luck, 2000). Designers should implement autonomy with caution, bearing in mind that autonomous characters are more life-like if their behaviors are consistent and sociable. Players commented that, NPCs of Oblivion who have a 24 hour schedule of their own, are less life-like than the NPCs who stand around in Morrowind: the predecessor of Oblivion.

Connectivity is the opportunity to share the virtual world together by connecting multiple computers via a network, usually either a LAN or the Internet. User defined interactivity requirements indicate an important difference in terms of connectivity. According to user comments, local area network (LAN) multiplayer capabilities and massive multiplayer capabilities are two different predictors of presence. Frequency analysis shows that, 4.3% does not want massive multiplayer capabilities, 3.7% wants massive multiplayer capabilities and 9.1% wants LAN multiplayer capabilities. This study used large scale connectivity as an independent variable but user comments indicated that large scale connectivity is not a good predictor of presence.

REFERENCES

Aylett, R., & Luck, M. 2000. "Applying artificial intelligence to virtual reality: Intelligent virtual environments". *Applied Artificial Intelligence*, 14 (1), 3-32.

Bostan, B. 2009. "Requirements Analysis of Presence: Insights from a RPG Game." *ACM Computers in Entertainment* 7, No.1 (March).

Ermi, L., & Mäyrä, F. 2005. "Fundamental Components of the Gameplay Experience: Analysing Immersion". Paper presented at *DIGRA 2005: Changing Views: Worlds in Play*, Vancouver, Canada.

Friedl, M. 2003. *Online game interactivity theory*. Hingham, MA : Charles River Media.

Heeter, C. 1992. "Being there: The subjective experience of presence". *Presence: Teleoperators and Virtual Environments*, 1(2), 262-271.

Heim, M. 1998. *Virtual Realism*. New York: Oxford.

Held, R. & Durlach, N. 1992. "Telepresence". *Presence: Teleoperators and Virtual Environments*, 1(1), 109–112.

Insko, B. E. 2003. "Measuring presence: Subjective, behavioral and physiological methods". In *Being There: Concepts, effects and measurement of user presence in synthetic environments,* Riva, G., Davide, F., & IJsselsteijn, W. A. (Eds.), Amsterdam: Ios Press.

Laurel, B. 1993. *Computers as theatre*. Addison-Wesley Publishing Company, Reading, MA.

Lessiter, J., Freeman, J., Keogh, E., & Davidoff, J. 2001. "A cross-media presence questionnaire: The ITC-Sense of presence inventory". *Presence: Teleoperators and Virtual Environments*, 10 (3), 282-297

Lombard, M., & Ditton, T. 1997. "At the heart of it all: The concept of presence". *Journal of Computer Mediated Communication*, 3 (2).

Magerko, B., Laird, J.E., Assanie, M., Kerfoot, A., & Stokes, D. 2004. "AI Characters and Directors for Interactive Computer Games". In Proceedings of the *2004 Innovative Applications of Artificial Intelligence Conference*, San Jose, CA. AAAI Press.

Manninen, T. 2003. "Interaction manifestations in multi-player games". In *Being There: Concepts, effects and measurement of user presence in synthetic environments*, Riva, G., Davide, F., & IJsselsteijn, W. A. (Eds.),. Amsterdam: Ios Press.

Mantovani, G., & Riva, G. 1999. "Real presence: How different ontologies generate different criteria for presence, telepresence, and virtual presence". *Presence: Teleoperators & Virtual Environments*, 8 (5), 540- 550.

Ondrejka, C. 2006. "Finding common ground in new worlds". *Games and Culture*, 1(1), 111-115.

Rafaeli, S. 1988. "Interactivity: From new media to communication". In *Sage Annual Review of Communication Research: Advancing Communication Science*, R. P. Hawkins, J. M. Wiemann, & S. Pingree (Eds.), 16, 110-134. Beverly Hills: Sage.

Regenbrecht, H., Schubert, T., & Friedmann, F. 1998. "Measuring the sense of presence and its relations to fear of heights in virtual environments". *International Journal of Human-Computer Interactio*n, 10 (3), 233-249.

Salen, K. & Zimmerman, E. 2003. *Rules of Play : Game Design Fundamentals*. Cambridge, MA: The MIT Press.

Sheridan, T.B., 1992. "Musings on telepresence and virtual presence". *Presence: Teleoperators and Virtual Environments*, 1(1), 120-26.

Slater, M. 1999. "Measuring presence: A response to Witmer and Singer questionnaire". *Presence: Teleoperators and Virtual Environments*, 8 (5), 560-566.

Slater, M., & Wilbur, S. 1997. "A framework for immersive virtual environments (FIVE): Speculations on the role of presence in virtual environments". *Presence: Teleoperators and Virtual Environments*, 6 (6), 603-616.

Steinkuehler, C. A. 2006. "Why Game (Culture) Studies Now?". *Games and Culture*, 1(1), 97-102.

Stuart, R. 2001. *The design of virtual environments*. Ft. Lee, NJ: Barricade Boks.

Sutcliffe, A. 2003. *Multimedia and virtual reality: Designing multisensory user interfaces*. Mahwah, NJ.: Lawrence Erlbaum Associates.

Towell, J., & Towell, E. 1997. "Presence in text-based networked virtual environments or 'MUDS'". *Presence: Teleoperators and Virtual Environments*, 6 (5), 590-595.

Witmer, B. G., & Singer, M. J. 1998. "Measuring presence in virtual environments: A presence questionnaire". *Presence: Teleoperators and Virtual Environments*, 7 (3), 225-240.

Zahorik, P., & Jenison, R.L. 1998. "Presence as being-in-the-world". *Presence: Teleoperators and Virtual Environments*, 7 (1), 78-89.

AUTHOR BIOGRAPHIES

Barbaros BOSTAN is an Assistant Professor at Yeditepe University, Information Systems and Technologies Department. Bostan earned a BS at Electronics and Communication Engineering at Istanbul Technical University, an MBA from Yeditepe University, a Ph.D. degree at Informatics Department of Marmara University. Bostan has teaching experience in the areas of computer networks, virtual reality systems and interactive web technologies. His research areas include interactivity, presence, computer games, RPGs, virtual environments, multiplayer virtual worlds and interactive storytelling.

Sertac OGUT is an instructor and a designer. He teaches Visual Communication and Interaction Design courses at the Communications Faculty of Marmara University Istanbul/TURKEY. He earned his BA on Communicative Informatics at the Marmara University. Ogut completed his MA study at the Yeditepe University in Visual Communication Design. He had his PhD degree at the Informatics Department of Marmara University. Ogut focuses on Interaction Design, New Media Studies and 3D Animation. Besides his academic carrier, he is working on several web-based game projects as an consultant.

A CONCURRENCY MODEL FOR GAME SCRIPTING

Joseph Kehoe
Institute of Technology, Carlow
email: joseph.kehoe@ITCarlow.ie

Joseph Morris
Dublin City University
email: joseph.morris@computing.dcu.ie

KEYWORDS
Concurrency, Games, Scripting

ABSTRACT

In this paper we outline a new model of concurrency that is specifically designed for the specialized domain of game scripting. Scripting is used extensively in game development both for the implementation of AI based behaviors and for providing game players with the ability to customize commercial games. Scripting languages have not, as yet, benefited from the move to multicore architectures. We discuss the properties unique to game scripting that any proposed model must satisfy. Then we propose a model of concurrency particular to games that addresses these issues while allowing game scripting languages to fully utilize multicore processors. The next steps in developing this model further are discussed.

INTRODUCTION

Game scripting is an integral part of computer game development. Scripting is essential for two reasons, the nature of the game development process and the type of game developers who write game behaviors.

The game development process is an iterative one. This is particularly true for entity behavior design and implementation. Entities in games are any objects that can interact with their surroundings and the player of the game. They range from pretty trivial items such as doors to non player characters that can form their own plans. Game entity behaviors can really only be properly assessed by actually playing the game with the entity behaviors *in situ*. This leads to the employment of rapid *design-implement-playtest* iterations. For this to be a viable process each iteration needs to be as short as possible. It can take many iterations before a behavior becomes acceptable within a particular game. These rapid iterations presume the use of a scripting language. Scripting languages are high level languages that allow for rapid implementation (or prototyping) of behaviors. As these languages are almost always interpreted rather than compiled they can be rapidly deployed, usually without requiring a recompilation of the surrounding game framework.

The second aspect of the development process that is applicable here is the programming ability of game designers. Behavior design is the domain of game designers in that they know best what behaviors are suitable for each game scenario. Game designers, however, are not game programmers. They do not have the full range of programming skills that are available to professional programmers.

Forcing game designers to pass their behavior designs onto professional programmers for implementation is not a viable option. It would tie up an expensive professional programmer who could be employed on other parts of the game while also slowing down the *design-implement-playtest* cycle by an unacceptable amount of time. Ideally, the game designer herself, as the person who fully understands the required behavior, should be able to implement the behaviors directly. To make this possible a simple high level scripting language is required.

Processor power (and speed) has been increasing at an almost constant rate, following Moore's law, since the introduction of the integrated circuit. Future increases in speed will be through the use of multicore processors, (Blake et al. (2009)). More speed will mean more cores on each processor. Processors which used to contain a single complex core will now be replaced by processors containing many cores (J. Held and Koehl (2006), Borkar et al. (2006)).

The only way for software to take advantage of these new architectures is by simultaneously using as many of the cores as possible. In other words software must switch from being sequential to concurrent. Unfortunately, writing concurrent code is more difficult than writing sequential code, particularly when using programming languages designed mainly for sequential programming. New techniques for designing, writing and testing concurrent software are needed and new programming languages may also be required.

Our model of concurrency is specific to scripting in the game development process. Scripting in games poses some unique challenges. We have two competing issues. Firstly, games are real time systems with hard time constraints. This implies the use of a low level language where developers have complete control over hardware resources. Secondly, and in opposition to this, it involves the use of a high level scripting language that will hide the details of the hardware from the programmer. Real time programming is a specialist skill that is acquired by highly competent programmers only after

many years experience while scripting in games must be open to non professional programmers.
Game developers are willing to accept the usage of scripting languages in the development of games for the reasons given in the previous subsections. Using concurrency to help improve the efficiency of scripting languages is one way of tackling this issue.

Overview of paper

In the next section we give an overview of the model we propose. We look at key features of this model of concurrency and show how it fulfills the expectations set out in the previous section. We follow by reviewing related work in concurrency and game scripting. Finally we finish with our conclusions and list further work that needs to be completed.

PROPOSED CONCURRENCY MODEL

A game is a simulation that consists of a set of entities interacting with each other in some world. Each entity has its own state and a set of behaviors that determines how it responds to various stimuli. The game world that is being simulated has rules (gravity for example) that determine how certain types of interaction between entities take place. These world rules can be encoded in the entities themselves. The global state of the game is given by the sum of the states of all the entities that it contains.
Games proceed in a stepwise manner as a sequence of discrete moments in time. Each step represents a tick of the clock, or one particular moment in time, and a sequence of steps represents the passing of some duration of time. The step frequency can vary, with the frequency representing the granularity of the representation of time in the game. Step frequency is ultimately determined by hardware factors such as processor speed. At each step the entities update their state based on the events or stimuli that were generated during the previous step. Overall entity behavior through the lifetime of a game comprises the sequence of step behaviors made during that game. A play-through of a game is a finite sequence of these steps.
In the next sections we will look at the basic structural components of the model and how they fit together. These components are: entities, messages, steps and constraints.

Entities

The entity is the basic building block of game simulations. Games consist of many different entities that interact with each other under some game defined rules. These rules may include, for example, physics based interaction such as gravity and elasticity. Entities range from the simple, like a bullet or item of furniture, to the more complex such as a non player character (NPC) that has its own beliefs, desires, intentions and plans of action.
In general, although entity types are varied we can assume that they have some common attributes such as geographical position in the game world, an associated model (a visual 3-D representation of the item) and boundary dimensions (used for collision detection). More complex games will also model mass, elasticity and internal structure (a skeleton) for each entity as well. Everything can be defined in terms of individual entities and their rules of interaction. In our model only entities exist in games.
Game design consists of (among other things not relevant here such as ensuring game playability) identifying all the entities that make up a game, deciding which of their properties are relevant to the game and how they are allowed to interact.
Every entity runs in its own thread. Concurrency is limited only by the number of entities in the game. Games are simulations designed to be fun to play. Given the nature of games it is unlikely that there will be only a few entities in a game. Since the identification of entities within a game is already part of game design this approach does not add any extra overhead to game design. It is a natural way to identify concurrency and brings identification of concurrency easily within the remit of the game designer at no extra cost.
Every entity is composed of four components: state, interface, constraints and message queue.
State is a non-empty set of named attributes. Each attribute will have an associated value. A non-empty subset of these attributes will be immutable. Immutable attributes are given values when the entity is created and these values remain unchanged until the entity is destroyed. Each attribute is also labelled as either visible or hidden. The full set of named attributes with associated labels belonging to an entity is called the entity state. The particular value of a state is determined by the values assigned to each attribute in the state.
Each entity contains at least one attribute as part of its state. This is the ID attribute which uniquely identifies the entity. This is both immutable and visible. The value of the ID attribute is generated automatically on entity creation and guaranteed to be unique for each entity.
Entities know a non empty subset of the state belonging to every other entity. This subset will contain all the attributes that are labelled visible. If an entity knows the attribute of another entity then it is allowed to read the value contained by that attribute but it cannot write to a attribute belonging to the state of any other entity. Changes to an entity attribute value can only occur through the entity interface.
Every entity has a defined set of message signatures. Each message in this set represents the response of the entity to a specific type of stimulus. Each stimulus event

triggers a specific message for each entity affected by that stimulus. An entity can only respond to a stimulus if that stimulus generates a message matching the signature of a message defined in its interface. The signature of a message is determined by a message name and a sequence of attributes.

Each signature must be unique within an interface. The sequence of attributes is non empty and will include at least the sender attribute. For any message instance the value of this attribute will match the value of the ID attribute of the entity that generated the message. The set of message signatures is known as the interface. Entities know the full interface of every other entity. An entity is allowed send a message to any entity if it knows the value of that entities ID attribute.

The set of local constraints determines the set of acceptable combinations of values that the attributes belonging to the entity state can hold.

The constraint set may also determine allowable combinations of values that different entities can simultaneously hold. That is, the allowable values of an entity state can be determined by the values of other entity states. For example, it may be the case that two entities cannot occupy the same position in space simultaneously.

The message queue contains the full set of outstanding messages that the entity has to respond to during the current step. This queue will contain all messages generated for the entity during the previous step. Messages are processed by each entity in the order in which they appear in the message queue. Each entity message queue can only hold messages that match message signatures defined in that entities interface. When an entity is destroyed its associated message queue and any remaining messages in that queue are also destroyed.

Messages

A message consists of a name, a receiver ID and a collection of attributes and the values of those attributes. A message generated (or sent) during one step will always be received during the next step. This guarantee that all messages are processed during the succeeding step means that, as a consequence, we cannot fix an upper limit on step duration in advance without limiting the number of messages allowed to be generated in each step. As the number of messages increases step duration will also increase. A fixed step duration would be an advantage for any real time system but it has a number of associated costs.

Most importantly by fixing step duration we would lose computational determinism. Different processors are able to accomplish different amounts of work in the same duration. The overall result of any step would then become dependent on the processor speed. We maintain that determinism is more important than fixed step duration.

Determinism gives us independence from the underlying processor executing the scripts. This greatly simplifies the testing and debugging of scripts. If testing and debugging depended on the processor used it would become beyond the capabilities of many scripters and reduce the possibility of using the model in the high level prototyping and rapid iterative development cycles used in game development. Since we can decouple the step rate from the frame rate any increase in step duration can be handled in a graceful manner by the game engine.

The value of the Receiver ID is used to determine who the receiver of the message should be. Each entity has an associated message queue and messages are put in the queue belonging to the entity whose ID attribute value matches that of the receiver ID in the message. Messages are processed in the order that they appear in the message queue.

A message is acceptable only if it fulfills two conditions Firstly, the receiver ID value matches the ID attribute value of an existing entity and secondly that the message name and attribute collection matches a message signature defined in the interface of the entity whose ID matches that of the receiver ID.

All unacceptable messages are discarded. Only acceptable messages appear in message queues. In response to a message an entity can do any or all of the following:

1. Send messages to other entities if it knows the values of those entities ID attribute;
2. Create one or more new entities;
3. Update any of its own mutable attributes provided that these updates do not violate any of the constraints in its constraint set;
4. Destroy itself.

Steps

A complete computation consists of a sequence of two or more steps where the first step is the initialization step, the last step the shutdown step and all other steps are intermediate steps. During the initialization step two processes occur: entities are created and initial Messages are generated. The final step consists of two parts: the remaining messages are discarded and finally all entities are destroyed.

For every intermediate step all acceptable messages that were generated during the previous step are processed. All acceptable messages generated during the previous step will be present in the appropriate message queues at the start of the current step.

Entity state is updated instantaneously and simultaneously at the end of each step. State update is defined as the sequential composition of the messages contained in the message queue, in order, modified by a conflict resolution algorithm.

Any messages generated during this process are delivered instantaneously at step end. Delivered messages are put in the message queue belonging to the entity whose ID value matches that of the receiver ID value in the message.

The order that messages are placed in the message queue is defined by the message sorting algorithm. The message sorting algorithm can be any algorithm that guarantees:

1. Messages generated by a single entity for the same receiver are placed in the message queue in the same order that they were generated in;
2. The final order of the message queue is deterministic. That is, for any given set of messages their ordering is unique and will always be the same regardless of how many times the ordering algorithm is applied.

The default sorting algorithm orders message queues by using the message Sender ID attribute as the primary key and message generation order as the secondary key. This ensures that both conditions hold. Any other algorithm that fulfills our two guarantees is acceptable. The message sorting has an associated cost. Between steps sorting will have to be carried out. This overhead is justifiable because the algorithm ensures determinism in script execution.

Determinism is important because it isolates the script and the scripter from the underlying processor architecture. As we have already stated this makes testing and debugging feasible in the rapid development cycles used in game development and also in the prototyping environment of casual and hobbyist game development.

Conflict Resolution

That conflicts can arise between different entities is a recognized problem in games. Entities exist in a common world. In this world rules will exist that govern how these entities can interact with each other. Attempts to perform certain actions will bring entities into conflict with these rules. There are three possible approaches to conflict resolution:

1. Put onus on scripters;
2. Handle constraints using other parts of the game engine;
3. Let scripting system handle constraints automatically.

The first option is, in many ways, the simplest. The scripter should ensure that any code they write does the proper error checking. This has the advantage that it can be the most efficient technique. Scripters will know when run time checks are required and when they are not. Although this is a common approach it has the disadvantage of putting the burden on the individual scripter. Under our model there is the extra complication that entities can only see the state of other entities as they were at the start of the step making it difficult to check for conflicts with other entities during a step.

The second option is suitable when the constraints logically fall within the remit of some other specialized subsystem. This is the case when the constraints between entities are real world constraints. In this case the game physics engine can handle the constraints very efficiently.

The final option covers cases where the first two options are judged unsuitable. We use a predefined conflict resolution algorithm to determine how conflicts are dealt with. The conflict resolution algorithm is any well defined algorithm that ensures that state update does not violate any constraints defined in the entities constraint set. Constraints can be divided into two different types: *internal* and *external* constraints. Internal constraints are constraints that exist only within a particular entity. External constraints are constraints that hold between two or more different entities.

Internal, or local, constraints are the easier to deal with than external constraints. They are, by definition, internal stand-alone constraints and so each entity can deal with them independently of any other entity. Internal constraints are state invariants that must hold throughout the entity lifecycle. These constraints are defined at entity creation and can be checked after each message is processed. If a conflict is detected the algorithm can take the appropriate corrective action.

The default algorithm simply discards any messages that cause conflicts during message composition. Other algorithms may be employed that take different error correction measures.

External conflicts can only be detected once the step is completed. This is because we cannot tell the final state of each entity until the end of the step. Once all entities have completed their state update the value of each entity state needs to be checked for conflict with every other entity. Once a conflict is found between two or more entities a state rollback, of some predetermined kind, of one or more of these entities will be required. After rollback of an entity state we may have to recheck the new value of the state against the global constraints. This has the potential to be more costly than internal constraint checking as one state rollback can raise more new conflicts.

We do not feel that external constraints will be common or form an essential part of any game script. Firstly, most external constraints will be handled independently by the physics engine. Secondly, remaining external constraints can be encoded as one or more equivalent internal constraints. For these reasons we do not have a separate default algorithm to handle external constraints.

RELATED WORK

BSP - Bulk Synchronous Processing

BSP has been proposed as a bridging model for general purpose parallel computation by Valiant (1990). A BSP computation consists of a sequence of super-steps. In a super-step each component (a processor or core) is allocated a task consisting of a combination of local computation and, message transmission and reception from other components. After L time units have passed a check is made to see if the super-step has completed. If it has, then the next super-step is started. Otherwise the next L units are allocated to completing the current super-step.
In simple terms, at each step a set of local computations is undertaken concurrently. According to Skillicorn and Talia (1998), the aims of BSP are to make it simple to write concurrent code, be independent of target architectures and make performance of a program on a given architecture predictable. BSP allows you to put an upper time bound on a computation for a particular architecture. This makes the performance more predictable. In addition, deadlock using BSP is impossible. BSP is also easier to debug in that computations can be rearranged inside a superstep without affecting the outcome. BSP has been successfully integrated in scripting languages in the past by, for example, Hinsen (2007).

COOP - Concurrent Object Oriented Processing

Three types of concurrent object model have been identified: Orthogonal, Homogeneous and Heterogeneous (Papathomas (1995)).
The Orthogonal model views the object model and the concurrency model as two separate independent systems. In this case, locks are used to resolve any issues raised by concurrency. The orthogonal approach does not gain us any ground as it still retains explicit locking and all the problems that this implies (Sutter and Larus (2005)).
In the Homogeneous approach all objects are *active objects*. An active object is an object that runs inside its own thread. It represents a merging of process and object (Briot et al. (1998), Hernandez et al. (1994)). The internal state of an active object is private to that object. Any interaction that must take place between objects must take place via message passing. Generally, messages are asynchronous but there is variation between explicit or implicit acceptance of messages by objects.
The heterogeneous model contains both the active objects of the homogeneous approach and the passive objects of the orthogonal model. The most popular form of concurrent object oriented programming model is based on active objects.
Actor models of concurrency are closely based on active objects. An Actor is an active object that can send finite set of messages to other actors, create a finite set of new actors and define how it will behave in relation to the next incoming message. - Corrêa (2009)

Network Scripting Language

The Network Scripting Language described in Russell et al. (2008) (NSL) is designed for distributed games development. It runs across remote processors rather than multiple cores and gives some assurances of determinism and consistency maintenance between the various processors during game execution.
NSL uses active objects and a frame based approach similar to the approach advocated here. Because this language is designed for multiple distributed processors each processor will have its own copy of the state of the objects in the other processors. If there are n processors then there will be n copies of the overall state.
This approach, out of the three mentioned, is the most similar to our approach but the programming language is more complex. It is tightly coupled to the frame rate with a step being run exactly once for every frame but gives no guarantees as to when messages will be delivered.

CONCLUSION

We have outlined a model of concurrency developed specifically for games development, specifically game scripting. Game scripting is undertaken by game designers who are not professional programmers and therefore do not have an in-depth understanding of concurrency. As the game entity behaviors they develop have to be play tested to ensure they are appropriate they must use many rapid *design-implement-playtest* iterations during development. To enable them to make use of concurrency we developed a model that is easy to use, removes as much of the burden from the designer as possible and can be implemented in any standard game scripting language.

Further Work

Some work remains to be done on developing algorithms that can be used by the conflict resolver. Although a simple conflict resolution algorithm has been proposed it may be the case that different games will need to use different or more sophisticated conflict resolving algorithms.
We intend to produce a working implementation of our model. This implementation will demonstrate the viability of this approach. It will be incorporated into an existing game scripting language to show how it fits into already existing development tools and practices in the games industry. This will also serve to show how transparent this model is to game scripters in practice. As

well as demonstrating how simple the model is to use it will also show how easy it is to incorporate into existing game scripting languages.

Biography

Joseph Kehoe is a lecturer in Computing in the Institute of Technology Carlow. He has previously been director of the BSc in Games Development and is currently Director of the BSc in Software Development.

REFERENCES

G. Blake, R. Dreslinski, and T. Mudge. A survey of multicore processors. *Signal Processing Magazine, IEEE*, 26(6):26 – 37, november 2009. ISSN 1053-5888. doi: 10.1109/MSP.2009.934110.

S. Borkar, H. Mulder, P. Dubey, S. Pawlowski, K. Kahn, J. Rattner, and D. Kuck. Platform 2015: Intel processor and platform evolution for the next decade. 2006.

J.-P. Briot, R. Guerraoui, and K.-P. Lohr. Concurrency and distribution in object-oriented programming. *ACM Comput. Surv.*, 30(3):291–329, 1998. ISSN 0360-0300. doi: http://doi.acm.org.remote.library.dcu.ie/10.1145/292469.292470.

F. Corrêa. Actors in a new "highly parallel" world. In *WUP '09: Proceedings of the Warm Up Workshop for ACM/IEEE ICSE 2010*, pages 21–24, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-565-9. doi: http://doi.acm.org.remote.library.dcu.ie/10.1145/1527033.1527041.

J. Hernandez, P. de Miguel, M. Barrena, J. Martinez, A. Polo, and M. Nieto. Parallel and distributed programming with an actor-based language. In *Parallel and Distributed Processing, 1994. Proceedings. Second Euromicro Workshop on*, pages 420 –427, 26-28 1994.

K. Hinsen. Parallel scripting with python. *Computing in Science and Engineering*, 9(6):82–89, 2007.

J. B. J. Held and S. Koehl. From a few cores to many: A tera scale computing research review. Intel White Paper, 2006.

M. Papathomas. Concurrency in object-oriented programming languages. In O. Nierstrasz and D. Tsichritzis, editors, *Object-Oriented Software Composition*, pages 31–68. Prentice Hall, 1995.

G. Russell, A. F. Donaldson, and P. Sheppard. Tackling online game development problems with a novel network scripting language. In *NetGames '08: Proceedings of the 7th ACM SIGCOMM Workshop on Network and System Support for Games*, pages 85–90, New York, NY, USA, 2008. ACM. ISBN 978-1-60558-132-3. doi: http://doi.acm.org.remote.library.dcu.ie/10.1145/1517494.1517512.

D. B. Skillicorn and D. Talia. Models and languages for parallel computation. *ACM Comput. Surv.*, 30(2):123–169, 1998. ISSN 0360-0300. doi: http://doi.acm.org/10.1145/280277.280278.

H. Sutter and J. Larus. Software and the concurrency revolution. *Queue*, 3(7):54–62, 2005. ISSN 1542-7730. doi: http://doi.acm.org.remote.library.dcu.ie/10.1145/1095408.1095421.

L. G. Valiant. A bridging model for parallel computation. *Commun. ACM*, 33(8):103–111, 1990. ISSN 0001-0782. doi: http://doi.acm.org.remote.library.dcu.ie/10.1145/79173.79181.

Incorporating Reinforcement Learning into the Creation of Human-Like Autonomous Agents in First Person Shooter Games

Frank G. Glavin and Michael G. Madden

College of Engineering and Informatics
National University of Ireland Galway

frank.glavin@nuigalway.ie, michael.madden@nuigalway.ie

KEYWORDS
Reinforcement Learning, First Person Shooter Games, Sarsa, Human-Like, Unreal Tournament 2004, Pogamut

ABSTRACT

As graphics in modern computer games move closer to photorealism, the emphasis for game developers is switching towards improving the in-game Artificial Intelligence (AI). Traditional scripting and rule-based systems are being replaced by more intelligent and immersive approaches. The goal of AI in computer games is to create intelligent autonomous agents that mimic human behaviour as closely as possible, in order to create a challenging yet enjoyable experience for human players. This paper describes the application of Reinforcement Learning (RL), an approach inspired by how humans learn, to the creation of intelligent "bots" in a First Person Shooter (FPS) game.

INTRODUCTION

Artificial Intelligence in Computer Games

The task of designing and implementing an agent in a game that appears to be both intelligent and make human-like decisions is certainly a difficult one. A variety of techniques have been proposed in order to emulate human intelligence in computer games. Some of these techniques, as detailed in Westra (2007), will now be briefly described.

Hard coding is the most basic way of implementing AI. A simple example would be to have a list of conditional checks that have corresponding behaviours associated with them. If an agent in a FPS game, for instance, has very little health left, it should concentrate on finding "power ups"[1] as opposed to engaging in combat with other players. Hard coding gives the programmer full low level control of the agent. *Scripting* involves a further abstraction of hard-coded behaviours which are grouped into specific tasks. Scripted actions can often become predictable and human players can exploit this weakness.

A *Finite State Machine* (FSM) is usually combined with the aforementioned techniques to create representations of different scenarios. The other techniques can then be used to make scenario specific choices. A finite series of states exist and transitions between these states are predefined. Some states cannot transition to others and the transitions are initiated by either the internal state of the agent or by a trigger from the environment.

Fuzzy set theory involves the use of fuzzy sets whose elements have degrees of membership as opposed to being assessed in binary terms. *Fuzzy logic* involves the use of logical expressions for describing the membership in fuzzy sets (Russell and Norvig 2010). Fuzzy logic is used when we would like to know the degree of membership of an element as opposed to whether its membership is true or false. For this reason, truth values of between 0 and 1 are calculated. It can be very complicated to manually produce fuzzy logic for the complex interactions of all the values that make up a computer game agent.

First Person Shooter (FPS) Games

FPS games take place in a fast-paced, three dimensional environment in which the world is seen from the first person perspective of the player. The most basic game types are Death Match or Team Death Match in which the objective is to kill the opposing players using either the weapons that the player spawns with, or ones that are picked up from the environment. As the names suggest, these game types involve either working alone against all other players or forming part of a team and fighting against another team. A large variety of objective-based games also exist such as Domination[2] and Capture the Flag[3]. The team players and opposition players can be made up of human players over a network, programmed bot players, or a mixture of both.

[1] In-game items that increase player health.

[2] Players gain control of Domination locations.

[3] Players capture flags and return them to their base.

RELATED RESEARCH

McPartland and Gallagher (2011) applied the tabular Sarsa(λ) (Sutton and Barto 1998) reinforcement learning algorithm to a purpose-built first person shooter game. The algorithm was used to learn the controllers of navigation, item collection and combat individually. The experimentation involved three different setups of the RL algorithm, namely, HierarchicalRL, RuleBasedRL and RL. The results showed that reinforcement learning could be successfully applied to a simplified purpose-built FPS game. While the results are promising, they do not address the challenges evident in 3D commercial games with complex environments and, unlike our work, do not involve playing against human opposition to train the bot.

Smith et al. (2007) developed an algorithm called RETALIATE (REinforced TActic Learning In Agent-Team Environments) for Unreal Tournament[4]. The authors used an RL algorithm called Q-learning (Watkins and Dayan 1992) for learning winning policies in the Domination game type. That work was concerned with co-ordinating the team behaviour, as opposed to learning behaviours of the individual players, which is the focus of our work in this paper. They carried out experiments against three different teams with varying strategies. The results showed that the algorithm adapted well to the changing environments. Auslander et al. (2008) developed an agent called *CBRetaliate* in Unreal Tournament. This work aimed to enhance the use of the RETALIATE algorithm by introducing the use of Case-Based Reasoning (CBR). The results showed that the use of CBR could speed up the adaption process of the RL algorithm.

Di Wang et al. (2009) proposed the use of FALCON (Tan 2004) for developing a computer-controlled agent in Unreal Tournament 2004. The authors built two FALCON networks, one for weapon selection and one for behaviour selection. The bot learned by using cognitive nodes which could be translated into rules by associating a state and a particular action with an estimated reward. The bots created these rules in real time. The bot was tested by entering it into a 2K Bot Prize competition in which bots compete in order to convince human judges that they are human by engaging in human-like behaviour. While the proposed bot did not win the competition, it did receive the highest game score and managed to fool some of the human judges.

[4]`http://www.unrealtournament2003.com/ut2004/`

GAME AND DEVELOPMENT SOFTWARE

Unreal Tournament 2004

Unreal Tournament 2004 is a commercial first person shooter game that was developed by Epic Games and Digital Extremes. This multiplayer game allows players to compete with other human players and/or computer controlled bots. The game is built upon the Unreal Engine which has an open scripting language called UnrealScript. The availability of UnrealScript, for simple high-level programming of the game, has lead to a dedicated following of software modifiers (modders) and new content developers. Almost everything, excluding the graphical and physics part of the engine, can be modified by the user.

Pogamut 3

Pogamut 3[5] is an open-source platform toolkit for creating virtual agents in the 3D environment of Unreal Tournament 2004. It makes use of UnrealScript for developing external control mechanisms for the game. The main objective of Pogamut 3 is to simplify the coding of actions taken in the environment, such as path finding, by providing a modular development platform. The toolkit integrates the Unreal Tournament 2004 game, GameBots2004 (Kaminka et al. 2002), the GaviaLib library (Pogamut Project Core), the Pogamut agent and the NetBeans IDE[6]. A detailed technical description can be found in Gemrot et al. (2009).

REINFORCEMENT LEARNING

Reinforcement learning (Sutton and Barto 1998) is a branch of Artificial Intelligence in which a learner, often called an agent, interacts with an environment in order to achieve an explicit goal. The agent receives feedback for its actions in the form of numerical rewards. The agent learns from its interactions with the environment and aims to maximize the reward values that it receives over time. The agent must make a trade-off between *exploring* novel actions and *exploiting* the knowledge from earlier exploration.

The Suitability of Reinforcement Learning

For a human, learning to effectively play a first person shooter game is a difficult task that takes time and patience. In order to develop useful strategies and tactics, the player must observe his or her success by monitoring the outcomes of their individual actions. Over time, they build up a knowledgebase of information which helps guide them towards winning behaviours.

[5]`http://diana.ms.mff.cuni.cz/main/tiki-index.php`
[6]`http://www.netbeans.org`

In order to create human-like artificial opponents, we believe that bots should learn in a manner that closely mirrors how humans do. They should receive positive feedback for actions that increase their chances of winning the game and negative feedback for actions that contribute to losing the game. In this way, the bot receives valuable information from its experiences playing against human players. This information can then be used to improve its decision making in the future. We hypothesize that this process will lead to the creation of challenging, realistic opponents as opposed to ones that are predictable and either too strong or too weak.

Sarsa Algorithm

The Sarsa algorithm (Rummery and Niranjan 1994) is an on-policy *temporal-difference* (TD) control algorithm. TD methods learn directly from raw experience without any model of the environment's dynamics. Being an on-policy method, it continually estimates state-action values (Q-values) for a specific behaviour policy π while, at the same time, changing π toward greediness with respect to the Q-values. The algorithm, as described in Sutton and Barto (1998), is shown below.

Algorithm 1 Sarsa, an on-policy TD control algorithm.

```
Initialise Q(s, a) arbitrarily
repeat
  Initialise s
  Choose a from s using policy derived from Q
  repeat
    Take action a, observe r, s'
    Choose a' and s' using policy derived from Q
    Q(s, a) ⇐
    Q(s, a) + α [r + γQ(s', a') − Q(s, a)]
    s ⇐ s'; a ⇐ a';
  until (steps of single episode have finished)
until (all episodes have finished)
```

IMPLEMENTATION

In order to embed the Sarsa algorithm into the logic of the bot, we had to design appropriate states, actions and rewards. Our initial implementation places an emphasis on simplicity. The state, action, and reward representations that we use are described in the subsections that follow.

For the experiments reported in this paper, we use Sarsa algorithm settings of $\alpha = 0.2$ and $\gamma = 0.8$. α is the *step-size* parameter, which influences the rate of learning. γ is the *discount rate*, which determines the present value of future rewards. In our implementation, an episode consists of one single lifetime of the bot. The episode ends when the bot is killed. These episodes consist of several steps in which an action is taken and a reward and new state are observed. The Q-values are updated during each step.

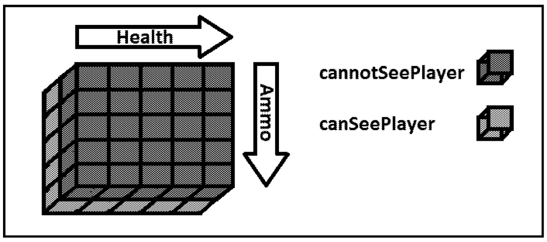

Figure 1: A graphical representation of the state space

States

There are 50 states in which the bot can be in at any given time. These states depend on whether or not the bot can see an opposing player, the bot's current ammunition level and the bot's current health. The levels of health and ammunition have both been discretized to include 5 levels for each. The values are calculated by checking what range they currently lie in. For example, if the bot has health greater than or equal to 100 it will return 0, if it has health in the range of 80 to 99 it will return 1, and so on. A 3D representation of the state space is shown in Figure 1.

Actions

There are eight actions available to the bot. These are listed and described in Table 1 below. The bot's logic method is called four times a second. The Sarsa algorithm is implemented in this logic method; therefore, actions are selected and evaluated in real-time, with Q-values being updated simultaneously. Actions such as *jump* and *dodge* must be allowed to complete once they are started, as a result of the Pogamut logic syncing mechanism. This prevents the bot, for example, from starting a jump and then cutting it short and immediately performing the next selected action. When the bot is looking for a player or a pick up item, it moves to the nearest available NavPoint[7] until an opposing player or an item is in view. The bot is also designed to only shoot its weapon when it can see an opponent, to eliminate pointless random shooting. Every gun in the game has two different shooting modes; primary and secondary.

[7] Each map is made up of a series of NavPoints in which the bot can use for movement.

Action	The bot will:
lookForPlayer	move through the map stopping when it sees a player.
lookForPickup	search for and move to the nearest visible item.
shootPrimary	shoot at any visible player in primary mode.
shootSecondary	shoot at any visible player in secondary mode.
dodge	perform a dodging maneuver in a random direction.
jump	perform a jump to a random height.
changeWeapon	change its weapon to another from the inventory.
goToLastSeenPlayer	go to the location of the last visible player.

Table 1: Actions that are available to the bot.

Status	Reward:
seeOpposingPlayer	+10
hasJustKilledOpposingPlayer	+10000
isCausingDamage	+1000
hasDamageBonus	+100
isHealthy	+10
hasCollectedPickUp	+50
isNotHealthy	-10
isColliding	-10
hasJustDied	-1000
isBeingDamaged	-200

Table 2: Rewards received depending on current status.

Rewards

In Reinforcement Learning in general, the design of the rewards is an important aspect which contributes to the success of the learner. In our current implementation, the reward signal is calculated by carrying out a series of checks to see what the current status of the bot is. For instance, the bot receives positive reward points for seeing the opposing player, causing damage, killing, and so on, whereas is receives negative reward points for being damaged, having low health, dying, and so on. The overall reward signal is a summation of the positive and negative reward points at any given time. A summary of the reward point system is shown in Table 2. The rewards are designed to reinforce the use of actions that result in the opposing player being damaged or killed.

EARLY EXPERIMENTATION

The following section describes some of the experiments that we have carried out in the current early stages of this research. We also go on to discuss some of the challenges and issues which have been raised during this time.

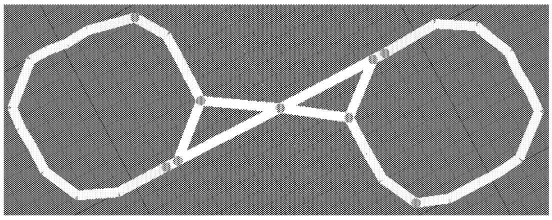

Figure 2: The Training Day map.

Experiment Details

All of the experiments consisted of 1 vs 1 death match games. The sole objective of this game type is to kill the opposing player using a variety of guns, some of which can be picked up from the environment. These games were played on the smallest map in the game, called Training Day, which is designed for 2-3 players. We believe that this is a good map to use as it removes any large search times between players and encourages almost constant combat. The layout of this map is shown in Figure 2.

The first stage of the experimentation involved a human playing against the reinforcement learning bot, which will we call Sarsa-bot. Every time the Sarsa-bot died (one episode) the current state of the Q-table was stored, so that we could keep track of the learning that was occuring as the Sarsa-bot was gaining experience. Sampling from these human vs bot experiments, we took the Q-tables from 0 to 140 episodes. The first of these Q-tables corresponds to when the bot has no experience at all whereas the last one corresponds to the bot's Q-table having played and died 140 times against a human player. These Q-tables summarise all of the learning of the Sarsa-bot, and any one of them can be loaded at the beginning of games in order for the Sarsa-bot to start the game with some experience. In order to identify if, in fact, any learning was occurring, we play the Sarsa-bot at different levels of experience (which we will call XP[8]) against a fixed-strategy bot from the game. We ran 5 games of 20 episodes with different levels of XP. It is important to note that once we loaded the XP Q-tables, we froze learning and did not allow the Sarsa-bot to update the table during the game against the fixed bot. The results are shown in the following section.

Results

The first measurement that we took from the games was the total number of actions that the Sarsa-bot managed

[8] 1-XP corresponds to the experience gained after dying once

to take during the game. This was an accumulation of all the steps taken for each episode of the game. These results are shown in Figure 3. While we can notice an increase in the total amount of steps during the games which included XP, the results are not definitive and could be caused by random elements of the game, the small sized map and the possibility that the bots could avoid each other by chance.

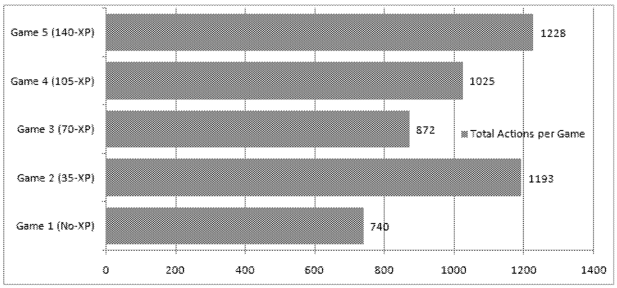

Figure 3: Total actions taken during each game.

We also recorded the maximum number of actions that the Sarsa-bot was able to take during one life in the game. This essentially corresponds to the maximum amount of time that the bot was able to remain alive. These results are shown in Figure 4. These results show a pattern in which the maximum time alive is related to the amount of XP that the bot has. This provides good evidence that the bot is improving over time. Detailed examination of performance traces shows that it acquires the ability to look for a pick up (escaping the situation) when injured.

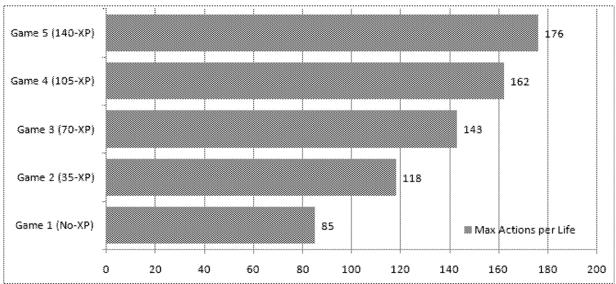

Figure 4: Maximum actions taken for one life.

We recorded the total reward received by the Sarsa-bot during each game. This value is the total positive rewards received minus the total negative rewards received. These results are shown in Figure 5. The total reward can be seen to be extremely negative for the bot without any experience, as expected. The bots that had XP all had positive total rewards for the game, albeit in substantially varying degrees. This varies from game to game based on how many times the Sarsa-bot actually managed to kill the opponent bot, since it receives a very large reward for killing. As noted earlier, the Q-tables in these experiments are frozen, so the Sarsa-bot does not have the ability to alter its behaviour in these games. The fact that all of the Sarsa-bots with some experience end up with positive total awards demonstrates that learning is occurring. There are, however, some issues of learning while playing against a human player as we will discuss later.

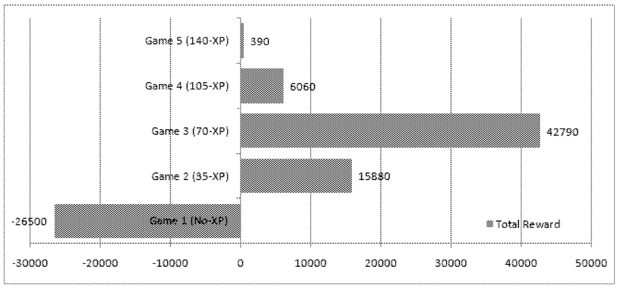

Figure 5: Total reward accumulated for each game.

The median reward per game is shown in Figure 6. The values, which are all negative, are significantly reduced when the Sarsa-bot has past experience. Once again, this shows that learning is occurring. Given the element of randomness associated with FPS games, we would not expect these values to uniformly decrease as experience in increased; in future work, we plan to have a much larger number of rounds in each game, which we expect will yield more uniformly progressing results.

Discussion

One challenging aspect of this work is in deciding how best to evaluate the performance of the proposed bot. Some issues arise when we begin to train the bot by playing many games against a human player. The human player for instance, can quickly became familiar with the game controls and learn how to play well. This can lead to instances in which the human player "goes easy" on the bot in order to give it more of a chance when, in fact, the bot is making poor decisions and inadvertently be-

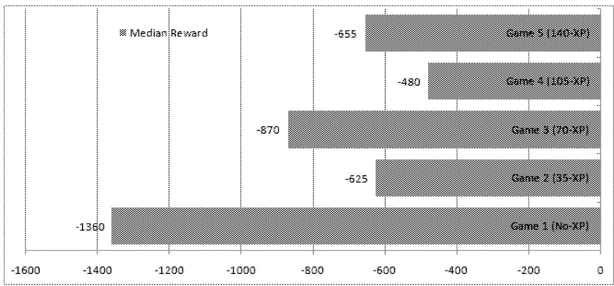

Figure 6: The median reward per life during each game.

ing rewarded for them. Also, if the human player takes a ruthless approach then they can kill the bot several consecutive times without giving it a chance to learn, for example, that shooting a player on sight is a good strategy. From our experiences we now believe that experiments involving human opposition would need to be both extensive and varied. Connecting the bot, as if it was human, to an online network could be a useful approach in which we plan to investigate. However, we were successful in using a fixed-strategy bot as a benchmark against which to assess the learning progress of the Sarsa-bot.

CONCLUSIONS

The results presented here are from initial experiments that took place in order to validate the use of a reinforcement learning algorithm into the logic of an agent in a FPS game. These results have given rise to very interesting questions about meaningful evaluation procedures and complex implementation issues.

In conventional RL settings, the agent can choose a single action in each state, which leads to a new state and a possible reward. However, in the setting we are considering, the bot must make decisions in real time (at a rate of 4 timesteps per second) and when it decides to perform an action, it may take several timesteps to complete, such as performing a dodge or a jump. While one action is underway, the bot can decide to take another action, so that two actions may continue at once. Therefore, when a reward is received, it may not relate to the most recently taken action. For example, if the bot accidently jumps off of a ledge, then fires its gun in the next timestep, and subsequently dies from the jump, it will incorrectly associate the negative reward from dying with the action of firing the gun.

While we have been able to demonstrate some success in designing a bot that can learn from experience as it plays against a human, we believe that it will be necessary to extend the standard reinforcement learning framework to deal with the real-time, multiple-action, complex-reward setting that is required for these games. To the best of our knowledge, this has not already been done. Such developments would also be of value to other real-time scenarios in which reinforcement learning can be applied.

FUTURE WORK

As well as extending the standard reinforcement learning framework to suit our needs, we also aim to define improved methods for evaluating the behaviour of the bot, and in particular for evaluating user enjoyment of games played against the bot. One possibility that we will consider is to carry out blind tests using a large variety of human players with differing skill levels, using Amazon's Mechanical Turk[9]. Games would be set up in which the human players would play against regular bots, the Sarsa-bot and other human players without knowing the true nature of their opponent. They could then rate the experience of playing each of the games and the results could be interpreted accordingly. Future work will involve designing such experiments.

REFERENCES

Auslander B.; Lee-Urban S.; Hogg C.; and Munoz-Avila H., 2008. *Recognizing the enemy: Combining reinforcement learning with strategy selection using case-based reasoning.* In *Advances in CBR.* Springer.

Di Wang B.S.; Tan A.H.; and Ng G.W., 2009. *Creating human-like autonomous players in real-time first person shooter computer games.* In *IAAI 09.* 173–178.

Gemrot J.; Kadlec R.; Bida M.; Burkert O.; Pibil R.; Havlicek J.; Zemcak L.; Simlovic J.; Vansa R.; Stolba M.; Plch T.; and C. B., 2009. *Pogamut 3 Can Assist Developers in Building AI (Not Only) for Their Videogame Agents.* In *Agents for Games and Simulations, LNCS.* Springer, 1–15.

Kaminka G.A.; Veloso M.M.; Schaffer S.; Sollitto C.; Adobbati R.; Marshall A.N.; Scholer A.; and Tejada S., 2002. *GameBots: a flexible test bed for multiagent team research.* In *Communications of the ACM.* vol. 45, 43–45.

McPartland M. and Gallagher M., 2011. *Reinforcement Learning in First Person Shooter Games.* In *IEEE Tr. on CIG.* vol. 3, 43–56.

Rummery G. and Niranjan M., 1994. *On-line Q-learning using connectionist systems.* Technical Report, Cambridge University.

Russell S. and Norvig P., 2010. *Artificial Intelligence: A Modern Approach (Third Ed.).* Prentice-Hall, Upper Saddle River, NJ.

Smith M.; Lee-Urban S.; and Munoz-Avila H., 2007. *RETALIATE: Learning winning policies in first-person shooter games.* In *AAAI 07.* vol. 22.

Sutton R.S. and Barto A.G., 1998. *Reinforcement Learning: An Introduction.* MIT Press.

Tan A.h., 2004. *FALCON: a fusion architecture for learning, cognition, and navigation.* In *IEEE IJCNN.* 3297–3302.

Watkins C.J.C.H. and Dayan P., 1992. *Q-learning.* In *Machine Learning.* Springer, vol. 8, 279–292.

Westra J., 2007. *Evolutionary Neural Networks Applied in First Person Shooters.* Master's Thesis, University Utrecht.

[9] https://www.mturk.com/mturk/welcome

PLAYER-TRACED EMPIRICAL COST-SURFACES FOR A* PATHFINDING

Sam Redfern
National University of Ireland, Galway
Galway
Ireland
E-mail: sam.redfern@nuigalway.ie

KEYWORDS

A* Pathfinding, Player Modelling, Player Tracing

ABSTRACT

This paper discusses the use of empirical cost-surfaces derived from substantial amounts of player-traced movements in an online vehicular combat game, for the purposes of improving A* pathfinding by AI vehicles. The fundamental concept is that we derive navigational meshes from human-player movements, with each node weighted by frequency of use. Our goals include the improvement of path travel times, aesthetic improvements, and the reduction of damage sustained while travelling across the map.

The results presented include quantifiable timings and observational characteristics. Quantifiable improvements include both algorithmic efficiency and travel time efficiency, while observations include the improved ability to avoid risky terrain features as well as other subtle human-like behaviours.

A best-performing non-linear cost function for the A* algorithm, based on player data, is suggested. Continued and future work on the AI in the game is discussed.

INTRODUCTION

This paper discusses the development of empirically-derived (player-mimicking) cost surfaces for AI pathfinding in the online vehicular combat game "Darkwind: War on Wheels". This game has been developed by the author since 2005 and, since it provides a substantial player-base and thousands of live games per week, is an ideal test-bed for AI research (Redfern 2007, 2010).

Although pathfinding in general, and the A* ("A Star") algorithm in particular, are well established techniques in computer games, improvements continue to be proposed in terms of aesthetics (Coleman 2009) – producing believable 'human-like' routes – and in experimental refinements for complex environments (Hale et al. 2010). The pathfinding requirements of typical open-terrain First Person Shooter (FPS) and Real Time Strategy (RTS) games, are fundamentally simpler than those of a vehicular combat game with realistic physics, tyre and chassis degradation, and collision damage models. In Darkwind, a car's momentum is critically important to its tactical movement and performance during combat; cars receive damage from poor driving and poor surfaces; various surface characteristics exist (e.g. sand, dirt, tarmac); and, effective routes across the terrain require cover from enemy fire. Furthermore, it is often appropriate to maintain a safe distance around dangerous obstacles such as cliff edges rather than choose an absolute shortest route.

Our core hypotheses are that (i) there are a number of subtle factors related to both effectiveness and aesthetic value, which define optimal routes around the terrains, and that (ii) it may not be feasible to deal with these factors algorithmically. We aim to achieve efficient, believable ('human-like') routes which navigate terrain features and surface types sensibly, are safe from collision damage and, where possible, enemy fire. Since Darkwind is a well established online multiplayer game, it provides substantial amounts of empirical evidence about player-chosen routes. This paper describes the use of this evidence to improve AI pathfinding.

A* PATHFINDING

The A* algorithm was first proposed in 1968 (Hart et al. 1968) and has been the most widely used pathfinding technique by games programmers, due to its effectiveness and efficiency. Numerous introductory explanations are available in the literature; a particularly good online description, for example is (Lester 2005).

The fundamental operation of A* is to traverse a map by exploring promising positions (nodes) beginning at a starting location, with the goal of finding the best route to a target location. Each node has four attributes other than its position on the map:

- *g* is the cost of getting from the starting node to this node
- *h* is the estimated (heuristic) cost of getting from this node to the target node. It is a best guess, since the algorithm doesn't (yet) know the actual cost
- *f* is the sum of *g* and *h*, and is the algorithm's best current estimate as to the total cost of travelling from the starting location to the target location via this node
- *parent* is the identity of the node which connected to this node along a potential solution path

The algorithm maintains two lists of nodes, the *open* list and the *closed* list. The former consists of nodes to which the algorithm has already found a route (i.e, one of its connected neighbours has been evaluated or *expanded*) but which have not themselves, yet, been expanded. The latter (closed) list consists of nodes that have been expanded and which therefore should not be revisited.

Progress is made by identifying the most promising node in the open list (i.e., the one with the lowest *f* value) and expanding it by adding each of its connected neighbours to the open list, unless they are already closed. As nodes are expanded, they are moved to the closed list. As nodes are added to the open list, their *f, g, h* and *parent* values are recorded. The *g* value of a node is, of course, equal to the *g* value of its parent plus the cost of moving from the parent to the node itself. If a node is already on the open list when it is evaluated, its *f, g, h* and *parent* values are only updated if the new *f* value is lower than the previously recorded one – this means a better path to the node has been found than the previous one.

The algorithm concludes when the target node is found, or when the open list is empty – the latter case means that a path does not exist from source to target, which is possible when you consider that some positions on the map may be non-traversable (e.g., mountains, lakes, walls, buildings).

There are various ways of calculating the cost of moving from a node to a connected node – the simplest and most common is to use Euclidean distance. It is also very common to take into account factors such as terrain cover or elevation changes. By applying a higher cost to difficult or steep terrain, the algorithm will be encouraged to find cheaper routes around these features rather than simply finding the shortest path. The current paper is primarily concerned with the identification of an appropriate mechanism for calculating costs, based on recorded player behaviour.

The choice of heuristic function *h(n),* which estimates the *h* value for a node (the cost of getting from the node to the target location), has a strong influence on the optimality and accuracy of the identified solution. If *h(n)* is always lower than (or equal to) the cost of moving from a node to the target, then A* is guaranteed to find a shortest path. The lower *h(n)* is, the more nodes A* expands, making it slower. If *h(n)* is sometimes greater than the cost of moving from *n* to the goal, then A* is not guaranteed to find a shortest path, but it can evaluate faster (Patel, 2011). In practice, therefore, it may be possible to dynamically modify the heuristic function in order to trade-off speed and accuracy as required during a game, if this is appropriate to the game.

PREVIOUS WORK

In recent years, the research literature has increasingly stressed the fact that game AI is not simply about winning the game or discovering the most optimal solution, but more critically is about making the game fun for the human player. From a pathfinding perspective, this means avoiding mechanical-looking routes in favour of believable, human-like ones – straight lines look better and more plausible, for example, than routes which zigzag and track around obstacles (Coleman 2009). Rabin (2000) uses splines and hierarchical approaches to introduce aesthetics into routes, while Higgins (2002a) and others use a second pass through a route in order to apply "aesthetic corrections". Coleman (2007) proposes a metric based on second-order derivatives and obstacle tracking in order to quantify the "beauty intensity" of paths, and later refines this approach to include fractal dimensions and rescaled range analysis (Coleman 2009).

John et al. (2008) propose a novel approach based on probabilistic pathfinding to produce varied high-quality routes and thereby improve game replayability – their examples presented provide a convincing argument for this approach in a team-based AI combat in a maze-like environment.

Few previous papers have discussed the use of recorded player behaviours in order to train AI systems – one exception is a case-based reasoning system developed to learn high-level strategies by mining recordings of expert human players playing a real time strategy game (Ji-Lung and Chuen-Tsai 2008). No previous work that we are aware of has taken this approach for navigation purposes. However, the increasing industry emphasis on logging player interactions and movements for other purposes, such as player category modelling and game personalization (Thawonmas et al 2009; Oda et al. 2009) is expected to be reflected in an increased research interest in this area. Online games are particularly suited to this approach, since the server can easily record data centrally, and since regular updates to the game are a normal part of the lifecycle after initial release: we can gather data over a period of time and use this to incrementally improve the AI in the live game.

One technique of interest is the 'heatmap' which can be used to visualise regular patterns of player behaviour over a spatial domain (Youngblood et al. 2011). Related work also includes the use of graph-based discovery algorithms to perform supervised learning (Cook et al., 2007), and the dynamic modification of navigation meshes based on the experiences of AI agents in complex game worlds (Hale et al., 2010).

EMPIRICAL COST SURFACES FROM PLAYER TRACES

Since 2008 we have been recording player movements on the game maps of Darkwind, and constructing A* nodes from these. However, a voting system was not established until June 2010: prior to this our data simply recorded where player vehicles had ever safely travelled. We now have 12 months of voting data collected: each time a node is revisited safely, a vote is accumulated for it. There are an average of about 3000 combats played per week, and an average of about 4 player-controlled vehicles per combat, spread across about 40 game maps. Vehicles typically travel 1-2km during a combat.

In order to ensure that we record only suitable votes, a 5-second cache of recently visited nodes is stored for player vehicles; if any damage is received due to collisions with terrain or other static obstacles, the cache is emptied without committing its data.

Figure 1 provides a visualisation of the A* vote nodes stored in the region of a desert mountain in the game. Each blue square represents a node, with both size and brightness proportional to the relative number of votes accumulated at that node. In this case, the cost of travelling to a node which has accumulated x votes, from a previous node at distance d metres, is taken to be $d/\sqrt{x}$.

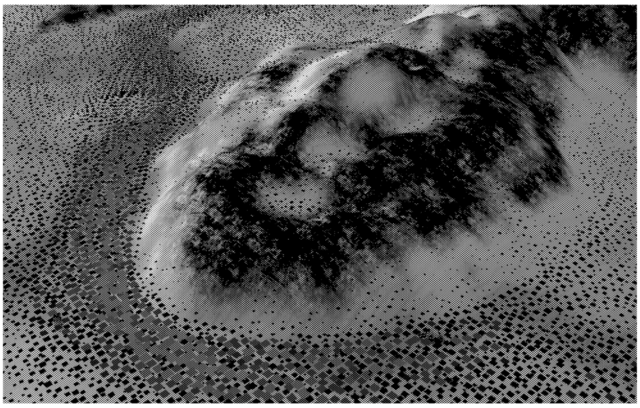

Figure 1: A Visualisation of the A* Player-Traced (Vote) Nodes Stored in the Region of a Desert Mountain in the Game

ADDITIONAL MODIFICATIONS TO THE A* ALGORITHM

Our implementation of the A* algorithm includes a number of modifications to improve efficiency and suitability for our requirements.

In order to provide rapid identification of the node closest to a world co-ordinate, nodes are pre-sorted into a world location-indexed hash table. This is implemented as a two-dimensional array of pointers to nodes, with one dimension indexed as a binned world x coordinate, and the other dimension indexed as a binned world y coordinate.

Long distance searches are calculated using a pessimistic (high) heuristic – speeding up the search substantially, while accepting sub-optimal routes. Since the AI drivers typically re-evaluate their paths every few seconds, a guaranteed shortest route is not needed on long routes.

We also maintain a sorted shortlist of 'promising' open nodes (i.e., those with the lowest f values), which allows rapid identification of the next node to expand without the need to maintain all open nodes in a sorted list. When the shortlist is emptied, the entire open list is searched in order to refill it, and if a newly opened node has a lower f value than the worst of the 'promising' nodes, it is added to the 'promising' list. This latter performance improvement is discussed in (Higgins 2002b).

We also treat separately by direction the edge (connection) between two nodes – since in rough terrain a path may be popular in one direction but unpopular (or impossible) in the other. In figure 1, for example, the nodes on the steep sides of the mountain are effectively only connected in the downwards direction.

EVALUATION OF PLAYER-TRACED VERSUS ELEVATION-BASED COSTING

The most obvious, and often well-performing function for algorithmically defining a cost-surface is the slope (local change of elevation) of the terrain. For purposes of comparison with our votes-based function, we therefore computed costs at each node based on the average absolute difference between that node's z position (i.e., on the world's 'up' axis), and the z position of its connected neighbours. Figure 2 provides a visualisation of this scheme: the size of the squares is inversely proportional to the node's cost.

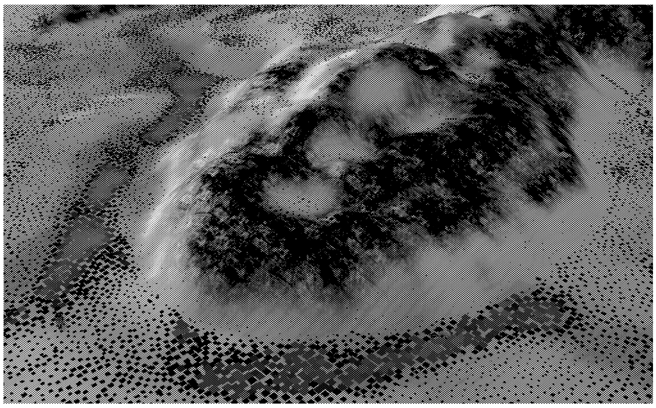

Figure 2: A Visualisation of the A* Elevation-Based Nodes Calculated Near the Same Desert Mountain

In many cases, the routes obtained when using elevation-based costing appeared very similar to those taken by the player-tracing approach. Although there were some exceptions, the general rule upon running time-trials was that the player-traced route was faster, on average by about 3%.

More importantly, however, the player-traced routes were frequently safer: elevation-based costing tended to produce routes closer to dangerous features such as cliff edges and trees. In figure 3, the route taken by the elevation-based approach was too close to the cliff, and the AI vehicle tumbled over the edge; in figure 4, the route taken through the garden caused a collision with both fencing and a tree, leading to a poor travel time. In figure 5, the route taken towards the town gates, while comparably fast, caused damage to the vehicle as it collided with the terrain while negotiating the small hills.

The game map illustrated in figure 4 consists of a ruined town with a good, wide road through its centre. The accumulation of votes indicates a very strong player preference for driving along the centre of the road. Players tend to drive fast on this road, and want to avoid collisions with fences or trees if their car spins or loses control due to weapons fire. This is a good example of subtle player behaviour that would be very hard to produce with algorithmic AI.

In terms of aesthetics, sometimes the elevation-based routes looked unnatural, especially on flat ground where features such as pits were 'edge-hugged' rather than driven around in a natural-looking way. It is probably also useful to note that, due to the underlying physical simulation, vehicles in the game are incapable of following zigzag paths due to their momentum – therefore the unsmoothed appearance of the routes illustrated in the images in this paper do not cause a problem aesthetically in the live game: we had no need to perform 'aesthetic improvement' calculations on them.

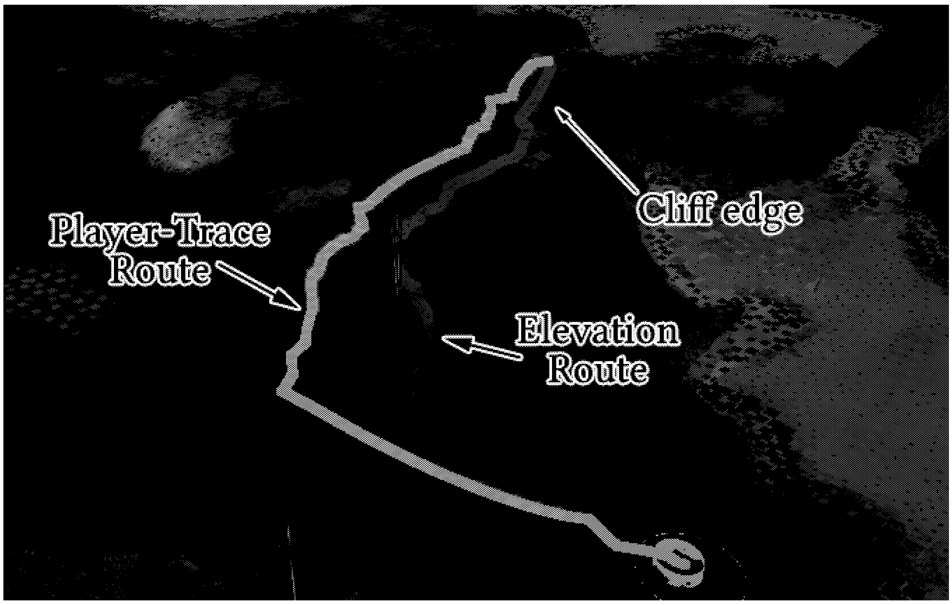

Figure 3: Elevation (Red) and Player-Traced (Green) Routes Near a Cliff

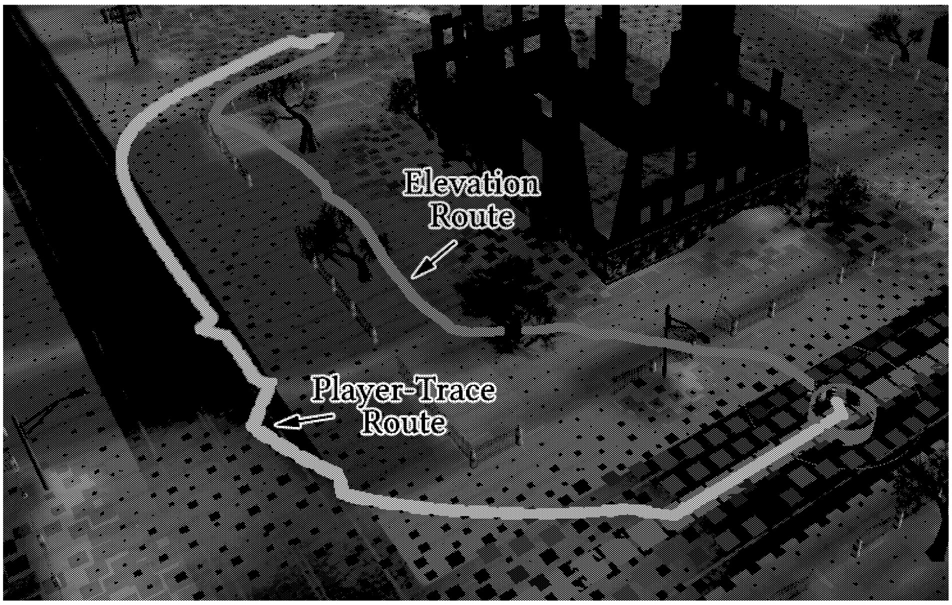

Figure 4: Elevation (Red) and Player-Traced (Green) Routes Around a Ruined House and Fenced Garden

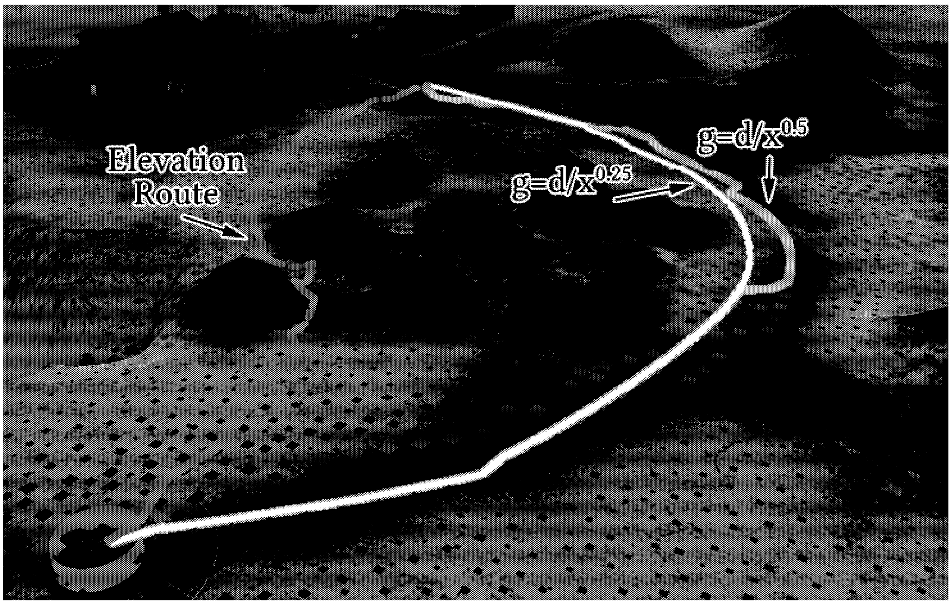

Figure 5: An Elevation (Red) Route Compared with Two Player-Traced (Green and White) Routes Which Use Different Non-Linear g (Cost) Functions

We frequently found the player-tracing costing approach to be more computationally efficient than the elevation approach, since it may direct the search far more tightly, expanding less nodes. This is clearly because the elevation approach often produces numerous almost-identically-scoring nodes close together. From a number of randomly-chosen tests across several maps, the performance benefit versus elevation-based costing ranged from zero (on hilly maps) to several hundred percent (on flat maps).

DEFINING THE A* PARAMETERS

We experimented with a variety of *g* cost functions, which is used to define the cost of travelling to a node based on the number of votes it has received. On safe, wide roads, it was found that a function which discriminated weakly between low amounts and high amounts of votes was more effective: for example in figure 5, the function $g=d/x^{0.25}$ produced a quicker (to travel) route than our generally best-performing function $g=d/x^{0.5}$ by about 4% - weak discrimination has a tendency to choose a shorter path towards the inside edge of corners. However, there is clearly a trade-off between speed and safety, and on routes such as the mountain in figure 1, the function $g=d/x^{0.25}$ performed poorly due to routing the car over rough terrain too close to the base of the mountain, and losing momentum: in this case, $g=d/x^{0.5}$ was quicker to travel by about 11%.

We found, again with some exceptions, that functions which discriminated very strongly between low amounts and high amounts of votes, such as $g=d/x$ or $g=d/x^2$, tended to produce erratic behaviours as the AI focused too strongly on finding 'popular' nodes, to the detriment of the overall route. This is clearly a complex situation, where factors such as the absolute number of votes cast on the map and the frequency with which games have been played on the current section of the map will have an effect. The general rule over 50 randomly-chosen test routes on various maps was that $g=d/x^{0.5}$ performed the best, on average, in terms of travel time, safety, and aesthetic value.

In order ensure an optimal path, we generally use a highly optimistic heuristic: we calculate $h = d * b$, where d is the Euclidean distance from the node to the target location and b is the cost attributed to the best-scoring node on the map. As mentioned previously, we do however vary this dynamically: a more pessimistic heuristic is used for long paths, in order to speed up the search process by expanding fewer nodes far away from the target location. To achieve this, we simply raise d to the power of 1.5 if it is larger than 50m.

CONCLUSIONS AND FUTURE WORK

We have described a novel use of player-traced navigation information as part of a voting system to inform cost-surfaces in AI A*-based navigation in a vehicular combat game with accurate physics. Experimental tests have validated the superiority of this approach over a cost-surface implementation based on local elevation changes.

We have also observed subtle behaviours in the player-traced approach, for example the avoidance of cliff edges and the preference for wide, flat routes rather than narrow gaps between terrain features. While we acknowledge that algorithmic terrain analysis could (with effort) provide *some* of these behaviours, our contention is that *every* subtlety of effective terrain navigation in this specific game has already been taken into account implicitly in the player traces.

This paper describes what is essentially a work in progress; although results are very promising and indeed Darkwind players report that they have witnessed a substantial improvement in AI navigational behaviour since the new cost function was implemented, we still have more work to do.

We intend to work on algorithmic terrain analysis, in order to produce comparable behaviours to those witnessed by the player-tracing AI. This will allow for more challenging (and therefore meaningful) comparisons between player-traced navigation behaviour and purely algorithmic AI. It will also, we hope, provide some useful algorithms of interest to AI navigation systems which cannot benefit from the wealth of player data that Darkwind has available – for example, offline single-player games and games with player-produced maps.

This paper has focused purely on the low-level navigational pathfinding task of the game AI – little has been said about the higher-level decision making which decides what the actual target locations for travel should be. The current situation in Darkwind is that a mixture of algorithmic AI techniques are used by a finite-state machine to, ultimately, produce these target locations. These techniques include simple terrain analysis (e.g. looking for 'sniper' points), group behaviours (such as re-grouping when separated, or scattering when receiving heavy ballistic damage), and outflanking behaviours which identify and respond to a 'gun line' by approaching enemies from the side. The latter tactic is in direct response to a favourite player technique in Darkwind by which a number of heavy vehicles form a static line and ambush the AI vehicles at choke points in the terrain. These techniques work reasonably in many cases, but there is generally a lack of high-level strategy or group co-ordination. The ability to navigate well across the terrain is not much use if you don't know where you want to go in the first place!

Generally, higher-level decision making needs to be improved in Darkwind, with influence maps (Tozour 2001) a likely candidate to supplement or replace the current rules-based AI. We have recently started gathering data for 'danger' influence maps, by logging the source and target positions of all successful gunfire attempts, along with the type of weapon and gunnery skill of the game character firing the weapon. Not only will this provide the data needed for 'danger' influence maps, but will also allow us to investigate our belief that gunfire avoidance is one of the subtle behaviours embedded in the player-traced data described in this paper. Additionally, we intend to experiment with an interesting approach to combining line-of-sight 'threats' into influence maps and thereby directly informing the cost function in A* path finding, as described in (van der Sterren 2002).

REFERENCES

Coleman, R. 2007. "Operationally Aesthetic Pathfinding". In *Proceedings of the International Conference on Artificial Intelligence*, 159-163.

Coleman, R. 2009. "Long Memory of Pathfinding Aesthetics". *International Journal of Computer Games Technology 2009,* 9 pages.

Cook, D.J; L.B. Holder, and G.M. Youngblood, 2007. "Graph-Based Analysis of Human Transfer Learning Using a Game Testbed". *IEEE Transactions on Knowledge and Data Engineering*, 19(11), 1465-1478.

Hale, D.H; G.M. Youngblood and N.S. Ketkar, 2010. "Using Intelligent Agents to Build Navigation Meshes". In *Proceedings FLAIRS Conference 2010*.

Hart, P.E.; N.J. Nilsson and B Raphael. 1968. "A Formal Basis for the Heuristic Determination of Minimum Cost Paths". IEEE Transactions on Systems Science and Cybernetics 4(2), 100–107.

Higgins, D. 2002a. "Pathfinding Design Architecture". In *AI Game Programming Wisdom*, S. Rabin (ed.), 133-145. Charles River Media.

Higgins, D. 2002b. "How to Achieve Lightning-Fast A*". In *AI Game Programming Wisdom*, S. Rabin (ed.), 114-121. Charles River Media.

Ji-Lung, H. and S. Chuen-Tsai. 2008. "Building a player strategy model by analyzing replays of real-time strategy games". *IEEE World Congress on Computational Intelligence*, 3106-3111.

John, T.C.H; E.C. Prakash and N.S. Chaudhari, 2008. "Strategic Team AI Path Plans: Probabilistic Pathfinding". *International Journal of Computer Games Technology 2008,* 6 pages.

Lester, P. 2005. "A* Pathfinding for Beginners". Retrieved 17th June 2011, http://www.policyalmanac.org/games/aStarTutorial.htm

Oda, J; R. Thawonmas and Chan, K-T. 2009. "Comparison of User Trajectories Based on Coordinate Data and State Transitions". *Proc. Fifth Int. Conf. On Intelligent Information Hiding and Multimedia Signal Processing*, 1134-1137.

Patel, A. 2011. "Heuristics". Retrieved 13th June 2011, http://theory.stanford.edu/~amitp/GameProgramming/Heuristics.html

Rabin, S. 2000. "Aesthetic Optimizations". In *AI Game Programming Gems*, M. DeLoura (ed.), 264-271. Charles River Media.

Redfern, S. 2007. "Psychic Software and Darkwind: War on Wheels." *New Age Gamer Magazine* 10(6), 32-35.

Redfern, S. 2010. "Evolving Racetrack Knowledge in a Racing Game". In *Proceedings AICS 2010*, 220-229.

Thawonmas, R.; J. Oda and K-T. Chen. 2009. "Analysis of User Trajectories Based on Data Distribution and State Transition: a Case Study with a Massively Multiplayer Online Game Angel Love Online". *Proc. GAME-ON2009*, 56-60.

Tozour, P. 2001. "Influence Mapping" In *Game Programming Gems 2*, M. DeLoura (ed.), 287-297. Charles River Media.

van der Sterren, W. 2002. "Tactical Path-Finding with A*". In *Game Programming Gems 3*, D. Treglia (ed.), 294-306. Charles River Media.

Youngblood, G.M.; F.W.P. Heckel; D.H. Hale, and P.N. Dixit, 2011. "Embedding Information into Game Worlds to Improve Interactive Intelligence". In *Artificial Intelligence for Computer Games*, P.A. González-Calero and M.A. Gómez-Martín (eds.), 31-53. Springer, New York.

BIOGRAPHY

SAM REDFERN attended the National University of Ireland, Galway, where he studied for a B.A. in English and Archaeology (1992), followed by an M.Sc. (1994) and Ph.D. (1998) in Information Technology. He has worked as a lecturer in Galway since 1996, and has published in the areas of digital image processing, various types of A.I., graphics, collaborative virtual environments and serious games. He has been an independent game developer in his spare time since 1984, with games published on the BBC Micro, Amiga, PC, Mac, iPhone and Android.

RACING GAME ARTIFICIAL INTELLIGENCE USING EVOLUTIONARY ARTIFICIAL NEURAL NETWORKS

C Süheyl Özveren
University of Abertay, Dundee, UK
c.s.ozveren@abertay.ac.uk

Victor Bassilious
University of Abertay, Dundee, UK
v.bassilious@abertay.ac.uk

Hamid Homatash
Outplay Entertainment, Dundee, UK
hamidhomatash@gmail.com

KEYWORDS
ANN, EANN, NEAT, rtNEAT, HyperNEAT, AI Agent, ML, Racing Game Simulator, Neuro-Evolution

ABSTRACT
This paper investigates the viability of using an Evolutionary Artificial Neural Network (EANN) approach as an alternative to standard Artificial Intelligence techniques used in a racing game. Use of Neuro-Evolution of Augmenting Topologies (NEAT) algorithms is compared to a standard AI technique which employs steering behaviours and a finite state machine to navigate an AI-driver agent around a circuit. We present a comparison between the NEAT algorithm and the standard AI technique described. Our initial literature review of the different available EANN approaches and the reasons for the choice of the NEAT algorithmic approach for our investigations is followed by the description of the implementation of our modified NEAT algorithm based EANN AI-driver agent and the racing simulation used for testing. Finally comparison of the results achieved with the implemented NEAT algorithm and the standard AI technique is followed by our conclusion on the comparative effectiveness of the NEAT and standard AI-driver agents and how our reported results can be further improved by future studies

INTRODUCTION
This paper focuses on developing a racing game Artificial Intelligence (AI) that uses an Evolutionary Artificial Neural Network (EANN) to train a racing game driver AI. This is then compared to a standard technique used in racing games to see how efficient, competent and viable EANN technique is from a game developer's perspective.
Throughout the history of video games Artificial Intelligence (AI) has played an important part in their success as evidenced from early games such as Space Invaders [Taito, 1978] and Pac-Man [Namco, 1980]. Although the AI was simplistic still the games were compelling, addictive and a huge success earning large amounts of revenue [Kendall, 2008; Goldberg].
Presently increasing emphasis is given to the creation of better (more advanced) AI. Games developers have been looking for new techniques to implement from academic AI research to improve the AI of their game agents [Rabin 2006], partly in response to the game playing public wanting more sophistication within the games they play [Baekkelund 2006, p. 78]. Focusing on learning algorithms Machine Learning (ML) is seen as one of the ways forward. There is currently much research going into ML algorithms to see how they can be implemented to produce AI agents that can adapt to their environments, while remaining challenging opponents to the player [Rabin, 2004, p. xii] yet producing challenging, varied and fun game play.
One genre of commercial video games that has successfully used ML techniques is the racing genre. A good example of this is Colin McRae Rally 2.0, which successfully used Artificial Neural Networks (ANN) to help AI agents learn to drive around a circuit in an offline supervised process [Hannan, 2001]. A criticism often cited against current racing games is that their difficulty level is often too high or low for the player of the game and has sometimes caused games to be patched after it has been released to fix this problem [Erickson, 2008]. Instead of fixing this problem after a game's release a ML technique could be used to dynamically adjust the difficulty level of the AI agent to meet the player's demands while allowing the outcome of races to be closer between the player and computer controlled opponents creating more tension and hence more fun.
Even though the racing game genre has quite successfully used ML techniques in the past, not many different self learning techniques have been explored [Woodcock, 2007]. On top of this there does not seem to be a great deal of research into racing game AI that learns. This is evident when compared to how much research and literature is focussed on ML methods in other game genres such as the real-time strategy genre [Lucas, 2003]. There is however a lot of research from academia in computational intelligence techniques on ML about racing robots and vehicle control. A lot of this research is dedicated to improving vehicle control using Evolutionary Artificial Neural Networks (EANN) [Togelius, 2006].
An EANN is an evolutionary algorithm that operates on its contained ANNs to produce a solution to a modelled problem [Dewri, 2003]. The algorithm uses some mechanisms inspired by biological evolution such as reproduction (crossover), mutation and natural selection [Buckland, 2002, p. 96-116]. In the context of ANNs an evolutionary algorithm can be used to mate different ANNs that contain different connection weights and or different topologies to create new and different ANNs [Dewri, 2003]. By combining both ANNs and evolutionary algorithms together, an EANN is produced.

EANNs are an unsupervised ML technique that are good in learning to control vehicles even when the exact physics model used by the vehicles is not known or very complex, such as in simulating radio controlled cars [Togelius, 2006] and racing cars [Loiacono, 2008]. They can also avoid some of the problems that supervised ML techniques have. For example, in Colin McRae Rally 2.0 an ANN was used to make the AI learn the optimal racing line for the physics model used. The problem with using a standard ANN is that as it is a supervised learning technique an external teacher must train the ANN to actually drive a racing car. This is usually achieved through the play testers of the game by driving around every racing circuit many times. The data collected by this process is then supplied to the ANN to make it learn; this is a very time consuming process [Schwab 2000, p. 460]. To avoid this problem an unsupervised learning technique such as an EANN can be used which means that the AI will "learn" how to race around a circuit automatically without any external input outside of the simulation which also have been shown to be capable of producing complex AI behaviours by utilising this self learning approach [Parker, 2007]. An advantage of using such an EANN-AI is that a racing driver could dynamically modify its driving ability according to how the human opponent is doing, or to any other external race conditions. However it is not worth developing an EANN if it cannot at least compete with the traditional racing game AI techniques within its ability to race. This is because an EANN could then not compete against a good human opponent, which would render the application of this technique useless. In our investigations we have tried to answer the following questions:

1. How computationally efficient is the EANN AI compared to standard racing game AI techniques?
2. How effective is the EANN AI at racing around a circuit compared to standard racing game AI techniques?
3. How effective will the EANN AI be able to race across multiple circuits? Therefore how good is its ability to generalise?

We believe that the answers to the above questions will allow games developers to make an informed choice about whether or not to try and implement an EANN in a racing game AI; for the desired benefits an EANN can provide over conventional racing game AI.

EVOLUTIONARY ARTIFICIAL NEURAL NETWORK ALGORITHMS

A number of EANN algorithms were considered when deciding what particular algorithm was best suited to the task of evolving an AI controller to race around a circuit. The key qualities to be considered that the algorithm must meet were:

1. How easy does the algorithm generalise?
2. How long does the algorithm take to evolve a competent solution?
3. How computationally efficient is it?
4. How difficult or easy is it to implement?

Although there are numerous EANN algorithms, also referred to as Neuro-Evolution (NE) algorithms there are only a few main NE algorithms that would be suitable for the task of learning to drive. The NE algorithms we considered in some detail were:

1. Conventional Neuro-Evolution (CNE) [Floreano, 2008]
2. Cooperative Synapse Neuro-Evolution (CoSyNE) [Gomez, 2006]
3. Neuro-Evolution of Augmenting Topologies (NEAT)

The main benefit of NEAT over other NE algorithms is that NEAT evolves the topology of the ANNs to yield an optimal topology [Gomez, 2008; Stanley, 2005]. This is highly beneficial as investigating through trial and error to find the best topology takes a considerable amount of time and might not be even possible depending on how complex the problem is [Dewri, 2003]. NEAT should also produce the minimal topology needed in order to solve the problem at hand, which means an evolved solution should be as computationally efficient as the problem allows.

Variants of NEAT

There are two other variations of the NEAT algorithm called Hypercube NEAT (HyperNEAT) and Real-time NEAT (rtNEAT). HyperNEAT allows efficient computation on thousands of neurons at once [Gauci, 2008]. This is not appropriate for this project as there are not a large amount of inputs into the ANNs. rtNEAT on the other hand would be directly applicable to this project since it is essentially the same NEAT algorithm as above, but it operates in real time. In other words, the evolution process continues as the actual game is played. However, there are two main issues with this. The first is that it requires the algorithm to evolve during actual races and not just at the end of these races. This means that the game design becomes a lot more complex in order to incorporate the algorithm, especially if the program has not been designed with this in mind [Vanik, 2006]. The second issue which is more serious is that rtNEAT has a patent pending. This means that in order to use the algorithm for commercial use a license to use it must be purchased, which may lead to undesirable financial pressures.

Standard Racing AI Techniques

Standard racing AI techniques usually comprise of two main components for racing around a circuit. These are a way for switching between driving states such as a finite state machine, and a steering behaviour for guiding the car around the circuit. A popular technique used to control a computer controlled racing car is to represent the circuit's best racing line as a curve and have a steering behaviour to follow the racing line [Charles, 2003]. This is usually combined with a finite state machine or rules based system in order to switch between different steering algorithms for different behaviours, such as aggressive, pacifistic and recovery driving [Azdima, 2001]. There are also variations of this technique that do not use an optimal racing line. These usually use the centre of the circuit and the curvature of the circuit at the current point to compute where the AI should drive to [Azdima, 2001]. This has the advantage that no racing line information needs to be specified, but it usually means that the optimal racing line is not represented correctly. This can lead to an AI that is not as proficient at driving as the technique that specifies a predefined racing line.

Summary of Techniques

We have concluded from our literature review that the EANN algorithm is most applicable to use in a racing game AI is NEAT since the NEAT algorithm evolves the topology of the contained ANNs and not just the connection weights. This means that a considerable amount of time will be saved from a trial and error approach to finding the best ANN topology. Although the NEAT algorithm is not the quickest at evolving, it should still be fast enough for this project. Therefore the chosen EANN algorithm to use will be NEAT.

As for the comparison technique a standard finite state machine combined with a set of steering behaviours that does not use a predefined racing line will be used for comparison. The reason why the technique will not use a predetermined racing line is based on the decision of the testing application that is used (see Testing Application). This testing application comes with a standard racing AI driver that does not use a predetermined racing line. Also this choice was made to ensure a fair comparison between the two algorithms since the NEAT algorithm will not have any information relating to a specific racing line either.

METHODOLOGY

In this section a detailed description of the particular choices for the game engine used are described, and why the project was developed in this way. After the taken approach has been discussed, an in depth look of how the NEAT algorithm was created and what parameters were used for evolution are described. Finally the choice of statistical data to collect is reasoned.

Testing Application

In our investigations we have used an open source racing game engine called The Open Source Racing Game Simulator (TORCS) [TORCS, 2001], to evaluate the AI techniques. This had numerous advantages, the main one being that the overall game itself is very close to being a commercial quality game. TORCS features multiple circuits and racing cars, along with a complicated physics model that could be found in most commercial racing games. Therefore this is a highly accurate way of comparing both techniques in commercial quality like game without actually testing it in one.

Another advantage that using TORCS had was that it shortened development time considerably, while still allowing full access to the source code for modification. The shortened development time was due to two main factors. The first factor was that the actual game had been built and therefore no development time was needed to further develop the internal game engine. The second factor was that the standard AI technique used is one of the standard AI drivers that come with the TORCS. Therefore the standard AI technique did not need to be created either, allowing more time to be focused on creation of the NEAT algorithm.

Quite a few development problems led to seeking a better implementation of TORCS. Fortunately a better implementation has been created by Daniele Loiacono of Politecnico di Milano, which was used in the WCCI 2008 competition [Loiacono, 2008]. This enhanced version features a client server model which abstracts the AI driver modules, detailed documentation, and also features a lot more car sensors than in the original version of TORCS. Although this version does not fix the majority of problems with TORCS the server client model implementation just avoids them. This decreased development time significantly and also provided a debugged version of the framework.

Implementation of NEAT

The created NEAT algorithm was based on Matt Buckland's implementation instead of the original implementation by Kenneth O. Stanley. This was because there is a detailed description of how to create NEAT in Mat Buckland's book "AI Techniques For Game Programming". Also the differences between the two implementations were negligible, therefore the implementation with the most documentation was chosen.

Evolution Parameters

A list of evolution parameters used throughout evolution of the NEAT algorithm is listed in Table 1. For the sake of brevity, a description of what each parameter actually does is also listed. The values chosen for each parameter were decided by a mixture of trial and error, and default values that are used in Mat Buckland's implementation of NEAT. Recurrent neurons were allowed as it has been shown that they can help to solve difficult problems more easily, hence speeding up evolution [Stanley, 2002].

Parameter	Value	Description
iPopSize	100	The size of the population.
iNumTicks	100	The time allowed in seconds for an AI driver to race on a circuit.
iNumAddLinkAttempts	5	Number of attempts to add a link. Sometimes it is difficult to find a link to members with different topologies.
dSurvivalRate	0.2	Controls the percentage of the best species members from which to spawn a new member.
iNumGensAllowedNoImprovement	20	Number of generations that are allowed for a species to exist when there is no improvement in the fitness score.
iMaxPermittedNeurons	1000	The maximum number of neurons that can exist in an ANN.
dChanceAddLink	0.07	Chance a link is added during mutation.
dChanceAddNode	0.04	Chance node is added during mutation.
dChanceAddRecurrentLink	0.07	The chance that a recurrent link is added during mutation.
dMutationRate	0.04	The chance that mutation can happen on a population member.
dMaxWeightPerturbation	0.5	The maximum amount a network weight can be perturbed.
dProbabilityWeightReplaced	0.1	The probability that a network weight can be completely replaced.

dActivationMutationRate	0.1	The probability that the sigmoid activation response will be mutated.
dMaxActivationPerturbation	0.1	The maximum amount that the activation response can be perturbed by.
dCompatibilityThreshold	0.26	The percentage of different genes species are allowed before being classified as a different species.
iOldAgeThreshold	50	Number of generations until a species is classified as old.
dOldAgePenalty	0.7	The old age penalty fitness score multiplier constant.
dYoungFitnessBonus	1.3	The young fitness bonus score multiplier constant.
iYoungBonusAgeThreshhold	10	Number of generations for the bonus fitness score to apply to a newly created species.
dCrossoverRate	0.7	How many of the population will crossover with each other.
iMaxNumberOfSpecies	0	The maximum number of species to exist at once. 0 signifies no limit.
iMaxGenerations	200	The number of generations to evolve for.

Table 1: Evolution Parameters

Inputs and Outputs of NEAT

The inputs into the ANNs of NEAT are listed in Table 2. All of the hidden and output neurons use the sigmoid function while the input neurons were linear. The sigmoid function was used instead of the hyperbolic tangent function. This is because the sigmoid function works more consistently for greater variety of applications, even though the hyperbolic tangent function may help to accelerate evolution [Bourg, 2004].

TORCS offers more possible inputs than the ones used but it has been shown that a good racing ability could be achieved with fewer inputs [Simmerson, 2008]. In fact fewer inputs are used than in Matt Simmerson's experiment in the WCCI 2008 competition. In his experiment he supplied 19 car sensors although only 5 were actually used. Therefore only 13 car sensors were supplied in this experiment as it was proven NEAT does not require them all, but the opportunity was still there if it required more than 5. Fewer inputs are also used than in Matt Simmerson's experiment because inputs for gear changing were omitted. This ensures that NEAT is only concerned with steering, acceleration and braking. The gear changing policy used is from the standard AI. This makes the comparison between the different techniques fairer as only their steering and acceleration behaviours are compared. In total there were 21 inputs and only 2 outputs, these are listed in Tables 2 and 3 respectively.

Parameter	Quantity	Description
Speed	1	The current speed of the car.
Lateral Speed	1	The current lateral speed of the car. Therefore this is a measure of how much the car is sliding.
Wheel Spin Sensors	4	How fast the wheels of the car are currently spinning. This is used to detect if the car may be starting to slide when used in conjunction with the lateral speed.
Circuit Sensors	13	Sensors that are used to detect the edges of the circuit within a 100m radius. The 13 sensors sample the space in front of the car every 15 degrees, spanning clockwise from 90 degrees to -90 degrees with respect to the car axis.

Table 2: Inputs into NEAT

All of the inputs were scaled into the range of 0 to 1. The inputs were scaled to prevent large input signals such as the speed, overwhelming the smaller input signals [Simmerson, 2008]. A similar concept was used on the two output parameters. Since the useful working range for the sigmoid function is 0.1 to 0.9 [Bourg 2004, p. 279], the outputs are scaled linearly from 0.1 to 0.9; to -1.0 to 1.0 to represent minimum and maximum output for steering and acceleration. If the outputs had not been scaled then it would have taken a longer time for the evolutionary process to approach the minimum and maximum values. Also since the sigmoid function asymptotes at 0 and 1, a minimum value of 0 and a maximum value of 1 can never be reached without scaling of the outputs [Bourg 2004, p. 279].

Parameter	Description
Acceleration	Controls the acceleration of the car. -1 denotes maximum braking while 1 denotes full throttle.
Steering	Controls the steering of the car. -1 determines full steering to the right while 1 is full steering to the left.

Table 3: Outputs from NEAT

Fitness Function

The fitness function (Figure 1) was designed to make small changes in fitness values corresponding to small changes in phenotype. It was designed in this way to avoid the fitness landscape being too noisy, which increases the chance of the NEAT algorithm getting stuck in local maxima [Thomas, 2006].

$$Fitness = Distance + Constant - (Damage * 5) - (TimeOffCircuit)^3$$

Figure 1: Fitness Function

The "Constant" value was set at 5000 making most fitness evaluations a positive value. The "Damage" was multiplied by 5 to enforce the notion that colliding with barriers during a race is undesirable. The same principle applies to the "TimeOffCircuit" value. This was cubed to not punish drivers that left the circuit for small periods of time (under a second for instance), but to severely punish drivers that left the circuit for larger periods of time as this is undesirable. There are no speed variables in this equation because to achieve an increase in distance the racing AI must have been travelling faster. Therefore there is no need to over complicate the fitness function with any speed variables. Evaluation of the current member is aborted if the damage was greater than 200 or the TimeOffCircuit was greater than 15 seconds. This stopped unnecessary evaluation of a poor performing member of the population resulting in less time evaluating a generation.

Additions to NEAT

The NEAT algorithm was extended to have a saving feature that allows a whole generation to be saved to disk and loaded back into the program. There are two reasons behind this. The first and the most obvious reason is that there needs to be some way to save the best member of the population, in order to actually use that member in a proper race. The easiest method to do that was to save that member to disk and then load that member for the appropriate race. The second reason is that evolving a generation can take a substantial amount of time and sometimes this evolution needs to be interrupted. For example, evolution needs to be interrupted to see how well visually the best member of a population is doing in the current generation. This was achieved by creating a system that automatically saves out every member of each generation that is evolved.

There are a number of advantages to this. It protects against losing all of the evolution data if something were to go wrong with the test application as the last saved generation state could be used to continue evolution. Another benefit is that any previous generation state can be loaded in and tested visually to actually see how a particular AI may have evolved. One of the most important benefits is that all of the statistical data of the generation is saved out along with the generation state. This means data such as fitness evaluation for a member of the population for any generation can be viewed to graph how effective the evolution process is going.

Statistical Collection of Data

The statistical data collected from the race to compare the algorithms are: computation time, lap times, total race time, damage received, maximum speed reached, average speed reached and the time off the circuit. Most of these statistics are obvious why they are required to compare the two techniques, but two are not. The reason why time off the circuit was needed was in order to see whether the drivers were cutting corners and if so, by how much. A good driving style does not require cutting corners therefore this is a measure of how the algorithm behaves. A similar principle applies to the damage received. The more damage the car has taken the worse the driving ability of the AI technique. For convenience TORCS was modified to output the extra comparison data in the statistical xml files it produces at the end of every race.

RESULTS

Test Conditions

The NEAT algorithm was evolved on two circuits. The first evolution was an initial test to see that it was operating correctly. This test was only allowed to evolve for 50 generations with a population size of 50 on a simple oval shaped circuit; A-Speedway. The simulation time allowed was equivalent to 60 seconds of real time; approximately two laps on A-Speedway. The second circuit that NEAT was evolved on was the circuit CG3; a complex circuit with many different types of corner. In this test, evolution occurred for 200 generations with a population size of 100. The simulation time allowed was equivalent to 100 seconds of real time, approximately just over one lap on CG3. The best member from this test was then tested on two other circuits. This was to see how well the NEAT algorithm would generalise and compare to the standard AI algorithm.

Results for Circuit A-Speedway

Results from the preliminary test show that the NEAT algorithm out performs the standard based AI, Table 4. It is more than 2 seconds quicker due to reaching a higher average and maximum speed, and it does not manage to take any damage. Also visually the NEAT algorithm takes a much improved racing line than the standard based AI using much more of the road surface to create a more smooth racing line. The only downside from these results is that the NEAT based algorithm is approximately 2.6 times slower to compute than the standard based AI.

	NEAT	Std. AI
Average Lap Time (Seconds)	26.19	28.69
Average Speed (km/h)	262.62	239.95
Maximum Speed (km/h)	272.32	264.6
Average Damage per Lap (%)	0	0.04
Time-off Circuit per Lap (Seconds)	0	0
Computation Time (Clock Ticks per Second)	140465	53406

Table 4: Table of Results for Circuit A-Speedway

	NEAT	Std. AI
Average Lap Time (Seconds)	77.40	64.52
Average Speed (km/h)	132.21	158.80
Maximum Speed (km/h)	169.31	228.9
Average Damage per Lap (%)	0.09	0.072
Time-off Circuit per Lap (Seconds)	2.44	0
Computation Time (Clock Ticks per Second)	110213	22786

Table 5: Table of Results for Circuit CG3

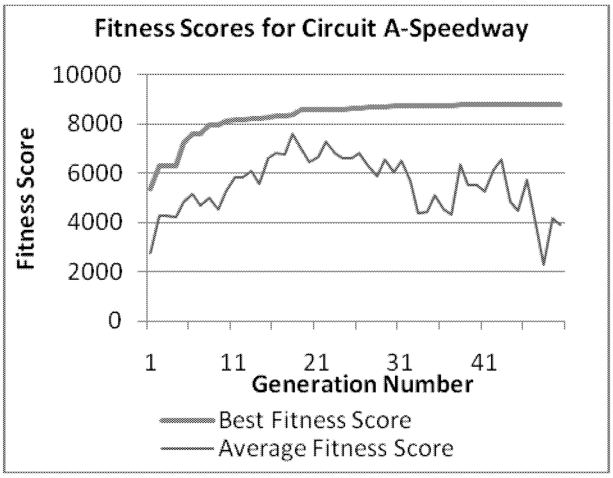

Figure 2: Graph of Evolved Fitness Scores for Circuit A-Speedway.

	NEAT	Std. AI
Average Lap Time (Seconds)	179.05	132.26
Average Speed (km/h)	129.35	173.47
Maximum Speed (km/h)	170.70	282.46
Average Damage per Lap (%)	6.03	0
Time-off Circuit per Lap (Seconds)	0	0
Computation Time (Clock Ticks per Second)	141104	28904

Table 6: Table of Results for Circuit Alpine1

In total the NEAT algorithm took approximately 3.7 hours to evolve in this test. Figure 2 shows the fitness scores produced from the NEAT algorithm per generation. From this graph it is evident that the NEAT algorithm quickly learns a solution to its problem. By the 10th generation it has very good driving ability as confirmed by the fitness score produced, which has been verified visually too. This shows that the NEAT algorithm produced an exceptional driver very quickly for this circuit. This is understandable because the circuit is a very simple search domain, which does not need any advanced behaviours. Another point is that the driver is over specialised to this circuit and therefore fails to complete a lap on any other circuit that is not a left-handed oval.

Results for Circuit CG3

The results from this test show that the NEAT algorithm does not compare favourably to the standard AI technique. It is on average about 13 seconds slower than the standard AI per lap. This is mainly due to the algorithm not evolving the behaviour to accelerate rapidly in fast sections of the circuit. For example, on the home straight it does not achieve a maximum speed of greater than 170km/h. The standard AI easily demonstrates that it is possible to reach speeds of 225km/h and above. The low maximum speed is probably due to the cautious behaviour that has been evolved where it uses the left side of the circuit as a guide. This is how the NEAT driver is able to follow the circuit by using the left edge of the circuit as a guide. This suggests that the NEAT algorithm is stuck in local maxima and cannot evolve an optimal solution to the problem. Therefore the fitness function may need to be adjusted to counteract this problem. Another explanation could be that given the inputs into the NEAT algorithm that this is close to an optimal solution. Therefore extra inputs may need to be provided to give the algorithm a more complete view of the racing environment to produce better results.

Another area where the NEAT AI does not fair favourably to the standard AI is in its computation time. It is over 4.8 times slower to compute than the standard AI, Table 5. This is due to the fact that the evolved ANN is more complex than in the previous test. This is understandable as the circuit CG3 is a more complex domain than the previous oval shaped circuit.

In total it took approximately 37 hours for the evolution of 200 generations with a population size of 100. This is a substantial amount of time and the graph in Figure 3 shows that it may have been unnecessary to evolve for 200 generations. This was because by generation 60 the NEAT algorithm had already evolved a competent driver that could complete a lap of the circuit within 81 seconds. This is only 4 seconds difference from the best population member, though its driving behaviour is not as refined.

Results for Circuits Alpine1 and Street1

As expected from the previous result the NEAT algorithm is a lot slower than the standard AI, on average approximately 47 seconds slower on the Alpine1 circuit, Table 6. It is expected that the NEAT algorithm would be slower as it had not been trained on this circuit, however the difference is significant. Another interesting race observation was that since there were no extremely long straights on the circuit CG3, it does not exhibit accelerating behaviour as fast as possible. This is evident because when approaching speeds of 170km/h (the maximum speed obtained on the circuit CG3), the NEAT based driver does not accelerate any further.

There are no results for the circuit Street1 for the NEAT algorithm. This is because there is an extremely tight corner in one part of the circuit which follows on from a high speed straight. Since the NEAT algorithm had never come across this type of situation before it does not react in time and veers off the circuit and into a wall. The impact leaves the car against the wall with the front of the car facing the wall and is unable to recover. Hence no results are produced.

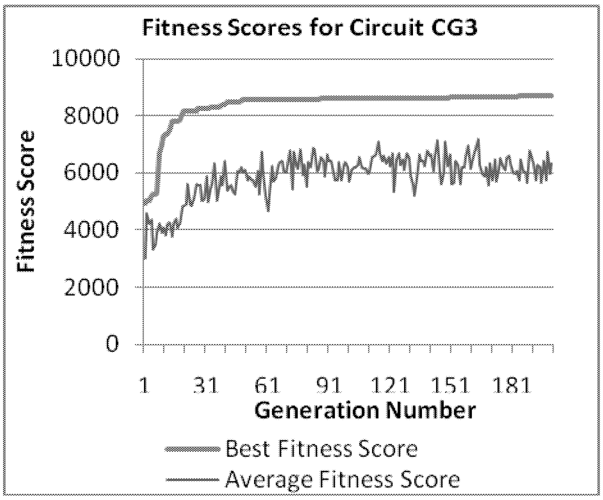

Figure 3: Graph of Evolved Fitness Scores for Circuit CG3.

Discussion of Results

Overall the results show that the NEAT based AI is not as competent a driver as the standard AI except in the first test. In the first test the NEAT algorithm excelled in such a simple search domain and produced a driver that was exceptional. The racing line that it followed was perfect for an oval shaped circuit and it achieved a higher average speed, maximum speed and quicker lap times than the standard AI driver. The only negative aspect was that the NEAT based AI was 2.6 times more computational expensive than the standard AI.

However, even though the NEAT driver had performed well in the first test, it only performed moderately well in the second. The reason for this was that the search domain that the NEAT algorithm had to find a solution was more complex. The circuit CG3 contained many different types of corners and as such the NEAT algorithm did not perform as well. The standard AI outperformed the NEAT based AI in every area. The most notable concern was that the NEAT algorithm was now more than 4.8 times more computationally expensive to evaluate than the standard AI.

This was due to the increased neurons that were added during "complexification" of the topology of the ANNs of the NEAT algorithm. The increase neurons mean more calculations are performed and as a result the AI driver became more computation expensive to evaluate.
The final test showed how well the NEAT algorithm could generalise. From being evolved on a completely different type of circuit the NEAT algorithm managed to generalise quite well. Its cautious driving style suited driving on circuits it had no knowledge of. However, it performed very badly compared to the standard AI technique, which is somewhat to be expected. The NEAT algorithm had to apply knowledge of driving from one domain into a different domain. Hence, why it performed poorly against the standard AI?
From the tests some interesting observations were made about the NEAT based AI driver. The first observation is that the NEAT AI driver has learnt to drive by following the left edge of the circuit. This is similar to path following steering behaviours that use the walls as guide in order to navigate [Reynolds, 1999]. This would seem to suggest that since there was no racing line to guide the NEAT driver, it used the only type of data that was similar to a racing line, and that was its position in relation to the centre of the circuit. Why it chose to follow the left edge of the circuit instead of the right edge is unknown. However, this is probably due to the first corner in the CG3 circuit being a right handed bend in which a racing driver would normally approach it from the left hand side. Therefore in early stages of evolution, this property may have been learnt and retained throughout successive generations.
Another interesting observation which is easily recognisable from the results obtained on the circuit Street1 (Table 7), is that the AI has no ability to recover itself from a crash. This is because the fitness function is geared towards avoiding a crash and therefore does not learn the behaviour needed to recover from a crash. This poses a problem which needs to be solved in order for an EANN AI driver to be a viable alternative to standard racing AI. There are two possible ways to solve this problem. One way is to use a finite state machine to check to see if the AI has crashed and use some sort recovery driving algorithm. Or the second method would be to create test cases in which the AI will crash and then let the NEAT algorithm learn to recover from such a situation. This would have to be done separately from the main evolution, as a different fitness function would have to be used in order to get correct behaviour.
One more interesting visual observation made about the NEAT based driver was that it was able to successfully slide around corners without losing control of the car. This is a significant observation as it is a complex behaviour. This is because it is producing enough steering force to maintain the slide, but not enough to lose control. It also means that the NEAT algorithm was able to learn how to use the lateral speed and wheel spin sensors inputs combined with the normal driving inputs to produce a complex behaviour.

CONCLUSION

The results point to two main conclusions. The first conclusion is that the NEAT algorithm can learn to navigate a circuit quite well and even race around a simple circuit such as A-Speedway better than the standard AI. The second conclusion is that on complex circuits it takes a lot longer to evolve a generic behaviour, and that this is not as good as the standard AI technique.
From the analysis of the results it is quite clear that an EANN is not more effective at racing a computer controlled car around a circuit. There are three main reasons for this. First and the most important reason is that the EANN AI produced worse behaviour than the standard AI controller. For example, the EANN AI was not as quick around the circuits as the standard AI, and it did not learn to take a racing line as good as the standard AI technique. There were also behaviours that the EANN failed to learn such as accelerating as quickly as possible in straight sections of the circuit, and being able to take very sharp corners effectively. These effects were probably due to the EANN being stuck in local maxima within the fitness landscape.
The second reason why the EANN is less effective than standard AI techniques is that it is more computationally expensive at runtime. This is exemplified when the EANN was evolved on a more complex circuit and had to learn more complex behaviour. It is nearly 5 times more expensive to compute than the standard AI technique. This computation cost could even go higher if the EANN were to learn more complex behaviour, as the number of neurons increase in the ANN, hence more calculation is required. This would mean that if a racing game had many computer controlled drivers, then an EANN is not an appropriate choice, as too much computation would be devoted to the AI drivers.
Finally, the EANN AI did not generalise very well. It was competent enough to drive around most circuits, but slowly, and in some circuits failed to complete a lap. This meant that compared to the standard AI, the EANN AI was not very effective.
Although this research shows EANN presently not as good as the standard AI techniques, however they do have many advantages. For example, they can learn complex behaviours with relatively few inputs as demonstrated by the sliding behaviour learnt by the NEAT algorithm. On top of this they can offer real time learning to make game play more varied and dynamic. This is their major advantage over traditional standard AI techniques. Finally, as demonstrated, an EANN can be more effective at racing around a circuit as demonstrated by the results for the circuit A-Speedway. Therefore they merit further investigative effort, which we will detail in the final section, in the search for a better AI technique than the standard AI techniques used in current racing games.

FUTURE WORK

To improve upon the results obtained from this project a number of possible future studies could be undertaken. The main problem that the NEAT algorithm had was that the produced driving style was cautious and slow. This was probably due NEAT fitness function stuck in local maxima and not finding the global maximum. Thus the effect of different fitness functions, evolution, mutation parameters and strategies could be looked at to see how they will affect the evolved driving behaviours of the NEAT algorithm.

Although this project used the NEAT algorithm as the EANN many others exist with their own characteristics. Different EANN algorithms may produce better results than what has been achieved by NEAT algorithm. Specifically the CoSyNE or rtNEAT algorithms could be investigated to see whether it can outperform the NEAT algorithm.

Another problem that the NEAT algorithm had was that it was general enough to race on most other circuits but was not able to race on them well. A possible way to improve this would be to train the NEAT algorithm on multiple circuits, with different types of straights and corners to ensure that the NEAT algorithm encounters all of the different possible types of circuits. This would be a lengthy evolution process and require radical restructuring of the TORCS game engine. However, this could be the only way to find out whether a fully trained EANN algorithm could be general enough to race on any circuit proficiently.

In this section we have detailed a few investigations, but there are many others that could be carried out. We believe that our results are promising enough for us and other researchers to continue further studies to find out if it is possible to create an EANN that can be more effective than the currently used standard racing game AI.

REFERENCES

Azdima, J. 2001. "AI Madness: Using AI to Bring Open-City Racing to Life." [Online] *Gamasutra*.

Baekkelund C. 2006. "Academic AI Research and Relations with the Games Industry." *AI Game Programming Wisdom* 3, Boston: Charles River Media. pp. 77-88.

Bourg, D.M. and Seemann, G. 2004. "AI for Game Developers." Sebastopol: *O'Reilly Media*.

Buckland, M. 2002. "AI Techniques for Game Programming." Ohio: *Premier Press*.

Charles, D. 2003. "Enhancing Gameplay: Challenges for Artificial Intelligence in Digital Games." *Proceedings of the 1st World Conference on Digital Games*.

Dewri, R. 2003. "Evolutionary Neural Networks: Design Methodologies." [Online] http://ai-depot.com/articles/evolutionary-neural-networks-design-methodologies/ [Accessed 12 December 2008].

Erickson, K. 2008. "Midnight Club LA Gets AI Patch." [Online] *PS3 Informer*, http://www.ps3informer.com/playstation-3/news/midnight-club-la-gets-ai-patch-009576.php [Accessed 23 April 2009]

Floreano, D. et al, 2008. "Neuroevolution: from architectures to learning" *Evolutionary Intelligence* vol 1, issue 1, pp. 47-62 [Online] http://dx.doi.org/10.1007/s12065-007-0002-4

Gauci, J and Stanley, K.O. 2008. "A Case Study on the Critical Role of Geometric Regularity in Machine Learning." *Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence (AAAI-2008).* California: AAAI Press.

Gomez, F. and Schmidhuber, J. and and Miikkulainen, R. 2006. "Efficient Non-Linear Control through Neuroevolution." *Proceedings of the European Conference on Machine Learning.*

Gomez, F. and Schmidhuber, J. and and Miikkulainen, R. 2008. "Accelerated Neural Evolution through Cooperatively Coevolved Synapses" *Journal of Machine Learning Research 9*

Goldberg, M. [No Date]. "Pacman: The Phenomenon - Part 1" [Online] http://classicgaming.gamespy.com/View.php?view=Articles.Detail&id=249 [Accessed 14 May 2009].

Hannan, Jeff, 2001. [Online] "generation5 interview" http://www.generation5.org/content/2001/hannan.asp [Accessed 23 October 2008].

Kendall, N. 2008. "Invade and conquer: Space Invaders thirty years on." [Online] *Times Online*. http://technology.timesonline.co.uk/tol/news/tech_and_web/gadgets_and_gaming/article4254644.ece [Accessed 14 May 2009].

Loiacono D et al. 2008. "The WCCI 2008 Simulated Car Racing Competition." *IEEE Congress on Evolutionary Computation, and Computational Intelligence and Games Symposium.*

Lucas, S.M. and Kendall, G. 2003. "Evolutionary Computation and Games." *IEEE Computational Intelligence Magazine.*

Namco. 1980. Pacman. [Arcade Machine]. Midway and Namco.

Parker, M. and Parker G.B. 2007. "The evolution of Multi-Layer Neural Networks for the Control of Xpilot Agents."

Rabin, S. 2006. "Preface. In: AI Game Programming Wisdom 3." *US: Charles River Media.* pp. ix-xii.

Reynolds, C.W. 1999. "Steering Behaviors For Autonomous Characters." *The proceedings of the Game Developers Conference 1999.* California. pp. 763-782.

Schwab, B. 2004. "AI Game Engine Programming." Boston: Charles River Media.

Simmerson, M. 2008. "Car Racing Competition at WCCI2008." [Online] http://www.scribd.com/doc/3450373/Car-Racing-Competition-at-WCCI2008-Matt-Simmerson#document_metadata [Accessed on 15 December 2008].

Stanley, K.O. and Miikkulainen, R 2002. "Evolving Neural Networks through Augmenting Topologies." *University of Texas at Austin* [Online] http://nn.cs.utexas.edu/downloads/papers/stanley.ec02.pdf [Accessed 1 November 2008].

Stanley K.O and Bryant, B.D and Miikkulainen R. 2005. "Real-Time Neuroevolution in the NERO Video Game." *University of Texas at Austin.*

Taito. 1978. "Space Invaders." [Arcade Machine]. Midway.

Thomas, D. 2006. "Encoding Schemes and Fitness Functions for Genetic Algorithms." In: *Rabin, S, ed. AI Game Programming Wisdom 3.* Boston: Charles River Media. pp. 677-686.

Togelius J. and Lucas S.M. 2006. "Evolving Robust and Specialized Car Racing Skills." *University of Essex.*

TORCS. 2001. "The Open Racing Car Simulator." [Online]. http://torcs.sourceforge.net/ [Accessed 10 January 2009].

Vanik, B. 2006. "Real-Time NeuroEvolution for Game Agent Control." [Online] *University of Central Florida*. http://www.hiranipra.com/files/vanik-neuroevolution-2006-04-12.pdf [Accessed 24 April 2009].

Woodcock, S.M. 2007. "Games Making Interesting Use of Artificial Intelligence Techniques." [Online] Available from: http://www.gameai.com/games.html [Accessed 5 December 2008].

STRATEGY GAMING

Genetic Programming and Common Pool Resource Problems with Uncertainty

Alan Cunningham and Colm O'Riordan
College of Engineering and Informatics, National University of Ireland, Galway
University Road,
Galway,
Ireland
email: a.cunningham6|colm.oriordan@nuigalway.ie

KEYWORDS
Artificial Intelligence, Genetic Programming, Common Pool Problems, Uncertainty

ABSTRACT

When applying an evolutionary computational technique to a problem, Game Theoretic predictions, like the Nash equilibrium, usually describe the problem's generated solutions. Human behaviour in the same scenarios has, however, been shown to not conform to these game theoretic predictions. In this paper, a Common Pool Resource (CPR) problem is used to compare the performances of human players and the generated solutions of a Genetic Programming (GP) algorithm. Previously, it has been shown that GP will converge as expected to the Nash Equilibrium predicted behaviour. However, under certain evolutionary scenarios, human-like play appears to emerge. In this paper, the effects of environmental pressures on the behaviours that Genetic Programming generates for a group based CPR dilemma. Similarities are drawn to human behaviours in similar games with a discussion of what influences the evolutionary process to generate human-like behaviours.

INTRODUCTION

In this paper, the ability of Genetic Programming to generate human-like behaviours for a group based Common Pool Resource game is explored. Firstly, a game from the literature is selected and the results of studies from previous work in varying fields are discussed. This research presents a study conducted with human subjects. These human trials are used to compare the behaviours generated in an evolutionary approach.

Secondly, previous work applying evolutionary computation to the CPR game is presented. Genetic Programming is chosen for a number of reasons: GP allows less constrained search than other methods, decision making is an inherent part of the GP algorithm which enables the possibility of human like reactive behaviours and it allows the understanding of the resulting behaviours by creating decision trees that are easily readable.

GP is applied to the same game that the human subjects played to establish a baseline for comparison with the human's behaviours and to investigate the claims of the game theoretic predictions for the players. Genetic Programming is shown to evolve solutions in line with expectations from game theoretic predictions which is shown to differ from the play by humans.

The main concern for this paper is the performance of GP in environments where there is uncertainty. In previous work, it has been shown that under certain circumstances GP will generates human like behaviours. Experiments are defined based on these findings, extending the environment to contain various types of environmental pressures. In each case, a comparison to human players's performance conducted.

Background

Common Pool Resource Problems

Common pool resources (CPR) are essentially a shared resource from which people or agents, who are sharing it, may extract some of the resource at a cost. The resource can typically be consumed by any agent, either freely or under regulations, and as such this affects all other agents wishing to share the resource. A characteristic of these resources is that they are not infinite and as such they may be damaged or destroyed by over extraction. An example of such a resource is a fishing grounds. The fishing grounds are shared among fishing vessels who agree to fishing quotas and to fish using certain nets. However, if a vessel was to violate these regulations in order to increase their catch it would have several effects. Firstly, there is an initial economic gain for that fishing vessel. Secondly, in the short term, there is a reduction in fish for others sharing the grounds to catch. Finally, there is the long term effect of potentially destroying future fish stocks as there are now less fish to reproduce. This is an example of how acting in one's own interest can provide a short term gain at the expense of others and at the expense of future possible gains. In this case, if everyone sticks to the quota the group as a whole does better and the fish stocks are preserved. CPR dilemmas are ones which try to model this interaction between short term selfish behaviour and long term group aspirations. In this case, we use the appropriation problem. The levels of return for a given level of input are known and the problem becomes one of excluding potential beneficiaries and allocating the returns from the pool.

Human Trials

Ostrom, Gardner, and Walker (1994) developed a series of laboratory experiments utilising human subjects in order to investigate the correlation between the behaviours of the human players and the behaviours predicted by non-cooperative game theory. The experiments designed all had multiple participants acting simultaneously in repeated rounds. In each round, the participants received an allotment of tokens and then decided separately where to invest the tokens. The tokens can be seen as a contribution of money or effort towards receiving a reward in return. In these games, there are two markets into which tokens may be invested. The first offers a fixed return based on each individuals investment in to it. The second is the common pool resource which offers a return based on the amount of total investment by the group in proportion to each individuals investment. The pay off function for the pool, in this experiment, is determined by a quadratic production function that is concave in form, that is, the amount the pool pays out increases with investment to a point, after which the return decreases.
The initial baseline experiment explored the performance of the humans in a game with minimal constraints and comprised the following parameters: eight human participants made finitely repeated investment decisions regarding an amount of tokens with which they were endowed at the beginning of each round. The tokens are then invested in either Market 1, offering a fixed return, or Market 2, the common pool offering a return based on the level of investment, or some combination of both. Participants know the number of other players, their own endowment, their own past actions, the aggregate past actions of others, the payoff per unit for output produced in both markets, the allocation rule for sharing Market 2 output, and the finite nature of the game's repetitions. Participants also know the mapping from investment decisions into net payoffs.
The main conclusion from this baseline experiment was that even as users reach the equilibrium point, net yield decays toward 0 and rebounds as subjects alter their investment strategies. In low endowment settings, aggregate behaviour results tend toward Nash equilibrium. In the high endowment setting, aggregate behaviour in early rounds is far from the Nash equilibrium however, it does approach it in later rounds. At the individual decision level, however, behaviour is inconsistent with the Nash prediction. In other words, the human subject sometimes appeared as if they collectively acted like the game theoretic predicitions said a rational subject would. Individually however, their strategies were not consistent with the predictions being a mixture of sub-optimal, reactionary and exploitative.

Genetic Programming Approach

In our previous work, a co-evolved genetic programming method for creating investment strategies in this baseline game is explored (Cunningham and O'Riordan, 2009). A comparison between the performance of the strategies generated and the group level behaviour of the agents with that of the human players performance is made. The findings showed the evolved GP behaviours converged to the Nash equilibrium point in the CPR dilemma as is expected by rational agents. The co-evolutionary process ensured that most of the population tended toward this point and as a result the agents do not get exploited. The evolutionary pressure, to create the simplest version of a given strategy with a penalisation on length of tree, results in fixed strategies devoid of decision making ability.
To explore the possibility of creating strategies that maintain their ability to be reactive, irrational play is introduced to evaluation process. In this case, the evaluation of each agent is against a selection of naive strategies instead of using a co-evolutionary process with all other parameters in the CPR game remaining the same. The result of this is the creation of human like play, that is, play by the agents that on a group level comes close to the group Nash equilibrium but that is varied at the individual level. The process of evolving against a set of fixed or irrational players has also maintained the GP created strategies ability to adapt in these environments.
An example of previous approaches to this problem using evolutionary approaches was the application of Swarm modelling to simulate the common pool resource dilemma (for an explanation of Swarm modelling see (Minar et al., 1996)). Deadman (1999) report almost identical performance from their agents, in terms of efficiency, when compared to the human players. In this model, 16 strategies are predefined some of which are derived from play in the human trails and others which attempt to maximise return round-by-round by increasing or decreasing investment. Agents are endowed with a subset or all of these strategies at the beginning of the game and the strategies remain fixed. A model of adaption is provided for the agents such that they may choose to use one of the strategies with which it has been endowed. These predefined strategies could be the reason that the performance was very similar to the human play. They also show that no strategy becomes dominant and even though the agents may have access to all of the strategies, their individual performances vary.
There are several other examples of CPR problems which have been explored such as the El Farol bar problem (Arthur, 1994) and the more widely studied Mi-

nority game (Challet and Zhang, 1998). Common pool problems have been discussed in the field of artificial life. Epstein and Axtell (1996) show how a common resource is consumed in an agent based modelling paradigm under various conditions. The workshop Bousquet et al. (2001), discusses the use of game theory and agent modelling as an approach for simulating resource management issues. The main difference in the game used in this paper to other examples is the partial membership to the pool that is provided through the investment of a token allotment.

There are not many applications of GP to this domain. There is an argument for applying GP in this way. Firstly, it provided a less constrained cognitive approach over GAs, as the tree structure allows for greater flexibility over a fixed gene length. Secondly, because the GP process creates decision tree structures, the reasoning used by the agent can more easily be understood.

Experiments

In this section, the effect of environmental pressures on group and individual behaviour is explored. The game rules will be altered to account for external forces on the agents and the results of their actions within the CPR dilemma. Firstly, an investigation into the effects of uncertainty or unpredictability of the actions of group members on the evolution of the behaviours is conducted. Randomness is introduced into the play of individuals in the group at different proportions and the change in behaviour of the evolved agents is studied. This will allow comparison with work from a similar field to see if these behaviours conform to their findings. Secondly, experiments are conducted to establish how introducing the notion of a finite resource, with probabilistic destruction of the pool resource, affected the behaviours of agents. Two experiments are carried out: one with a safe zone which permits some level of safe investment into the pool and the other without this safe zone. The probability of destruction of the pool is increase linearly by the amount of investment in the pool (outside the safe zone, where applicable). These experiments will allow the comparison between the behaviours of the evolved strategies and humans once more.

We use the GP process to create a tree for each agent representing its investment strategy for the CPR. As all tokens must be invested each round, the remainder are put into the fixed market. We use strongly typed GP to avoid any nonsensical tree creation and also to reduce the search space (Haynes et al., 1995).

The following evolutionary parameters were arrived at through experimentation and comparison with typical values used in similar research. In each evolutionary run, two hundred and fifty trees are evolved over one hundred generations. The fittest individual at the final generation is chosen as a representative of that run. A list of the GP parameters used in the experiments is shown in Table 1. Random trees are initialised according to the constraints in the nodesets and each tree must begin with any of the functions from the *Constants* set. These trees are then evaluated using the the cumulative profit for that generation. Once evaluated they undergo tournament selection in order to be chosen for the next generation. A tournament size of 5 is used in order to avoid a rapid convergence from selection pressure. The selected members of the population are then subjected to crossover and mutation with the probabilities 0.9 and 0.1 respectively.

Parameter	Value
Population Size	250
Number of generations	100
Games per Evaluation	At least 20 $f(2)$
Creation type	Ramped half and half
Creation Probability	0.02
Crossover Probability	0.9
Swap mutation Probability	0.1
Maximum depth for creation	6
Maximum depth for crossover	17
Tournament Size	5

Table 1: Parameters for GP

For each game, the CPR rules are set as follows: number of subjects is 8, production function (x_i is the investments by player i) is $23(\sum x_i) - .25(\sum x_i)^2$, market 2 return/unit of output \$.01, market 1 return/unit of output \$.05. These experiments are conducted with individual token endowments of 10 and 25 which result in the following payoffs: earnings/subject at group maximum are \$.91 and \$1.65 respectively, earnings/subject at the Nash equilibrium are \$.66 and \$1.40 respectively, earnings/subject at zero rent are \$.50 and \$1.25 respectively. In the finite resource experiments, the safe zones are 0 and 40.

The Effects of Randomness and Uncertainty

As is pointed out in (Jager et al., 2002) the introduction of uncertainty into a CPR dilemma leads to over harvesting which subsequently leads to under performance for the group or, where applicable, destruction of the resource. We conduct a series of evolutions which introduce an amount of randomness (in order to simulate uncertainty as to the behaviour of the group) to the group of individuals playing the game.

The individual trees are co-evolved, with each one representing the strategy for investment. The decision tree that is created dictates the amount of tokens to be invested into the pool with the remainder (from the allotment at the beginning of the round) being invested into the fixed return market. For evaluation, each member of the population plays against a random collection of other members 20 times and since each member can

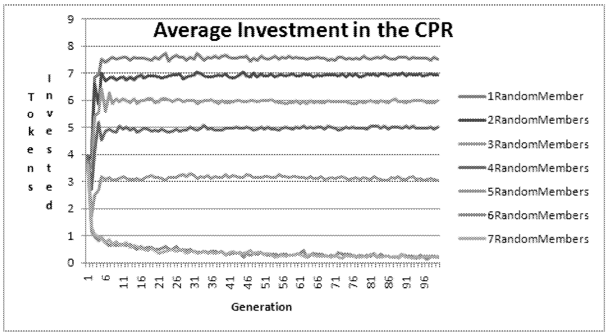

Figure 1: Average Investment in the CPR with varying degrees of randomness

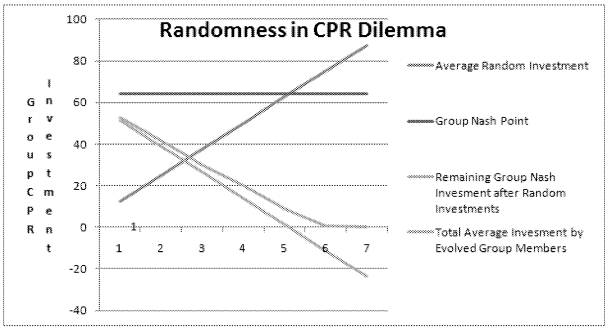

Figure 2: Average Investment by Evolved Members of the Group vs Random Members

be selected during other evaluations this number is substantially higher. The fitness of the individual for that generation is averaged over these games.

The results of this on the evolved behaviours are displayed in figure 1. Each line on the graph represents the average investment made in the CPR by the agents. Each point on the graph is the average of 8 runs for the same problem. For example, the $1RandomMember$ line represents the average investment at every generation for all the members of the population, averaged over 8 separate evolutionary runs, in an environment where one group member is playing randomly.

From the graph, it is evident that the populations are converging to specific investment points, which changes depending on the level of random investment by the members of the group. With only one random member, the group achieves close to the original Nash equilibrium investment point. When six or seven of the members of the group are playing randomly, the investment by evolved strategies drops to zero.

In figure 2, the predicted average investment by the random members of the group is plotted against the average investment by the evolved members and the group Nash investment point. In this scenario, the evolved players reduce their investment in such a way that each of the evolved strategies obtains an equal share of return on investment. As uncertainty increases, there is over exploitation, as the amount that the evolved strategies invest is, on average, above the remaining group Nash investment level.

Jager et al. (2002) use agents which are equipped with human-like cognitive processes in their simulations. These agents can use deliberation, social comparison, imitation and reputation of previous behaviour when making investment decisions. They show that increased uncertainty may stimulate an imitation effect that promotes over-harvesting. The increased uncertainty also leads to an increased optimism about future returns and a lessened ability of agents to adapt during resource depletion. In this experiment, the increase in uncertainty leads to a decrease in investment by the evolved agents in the CPR inline with the increase average random investment.

Probabilistic Destruction of the Commons

The notion of a limited resource is introduced into the CPR dilemma. This extension examines what happens when there is the potential to destroy the resource which is being shared. The effects of, not only the other agents, but also the environment in which the game is being played provides an influence on the behaviours of the agents.

Probabilistic destruction exists in two forms: the first with a safe-zone of investment and the second with the safe zone removed. In this case, destruction occurs when (outside the safe-zone) each token invested in the pool increases the chances that the game will terminate by 0.5%. The derivation of the figure comes from the total number of tokens resulting in a 100% probability of destruction of the pool. Once either twenty rounds have been played or the pool has been destroyed, the game ends. Eight separate evolutionary runs are completed and the evolutionary trajectories are plotted as averages of these. For comparison, the best agent at generation 100 of each run is chosen as a candidate solution.

The evolved results show the population converging to a point where each member invests exactly five tokens each into the pool, for a group total of 40, with the remainder being invested in the fixed market. The GP process converges to the rational strategy of maximising the return from the CPR whilst ensuring that the resource is not destroyed. An analysis of the candidates from each evolutionary run reveals that, at an individual level, this rational behaviour is indeed the case. This behaviour differs from that of the humans. Even though the same information was available to the humans playing the game, they were unable to avoid destroying the resource in this case.

In the second experiment, the safe-zone is removed leaving only an increasing probability of destruction of the resource. In this case, the evolved solutions are unable to avoid destroying the resource. An analysis of the candidate solutions from the eight evolutionary runs reveals that the agents preserve the resource for an average of 4.04 rounds (100 trails). One interesting feature of their behaviour, is the fact that the agents play close to the Nash predicted level of investment (average of 7 tokens for the 8 individuals). This indicates that, while the resource is being prematurely destroyed, whatever income is garnered from it, is distributed almost equally among the participating agents. In this experiment, the human trials also destroy the pool quickly although they do not display the same coherency of investment pattern.

In these experiments, the evolutionary processes produce rational strategies that are predicted by the Nash equilibrium when there is a safe-zone. This play is unlike the humans, who are unable to preserve the pool with the safe-zone in place. When the safe-zone is removed, the evolved strategies do not preserve the length of the game as they invest into the pool. This play is similar to the humans at a group level. At an individual level however, the evolved investment strategies still preserve some notion of equality, by converging close to the Nash predicted investment point.

Discussion and Future Work

In this paper, the solutions generated by GP for a CPR dilemma are compared with behaviours of humans. Particularly, the effects of environmental pressures are studied. Previously, the introduction to individual irrationality within the evolutionary process changed the behaviour of the agents such that they behaved similar to humans. Introducing randomness into the game environment sees the behaviours converge to fixed points, sharing equally the returns. The increase in uncertainty leads to a decrease in investment by the evolved agents in the CPR inline with the increase average random investment.

The introduction of the probability of destruction of the pool resource has two effects on the group, depending on whether or not there is a safe-zone of investment. When a safe-zone of investment is established, the agents evolve to play within bounds of the safe-zone, preserving the pool and maximising the return for the game which was unlike human play. With the removal of the safe-zone, the agents are no longer able to maintain the pool and games typically last a short amount of time. At a group level, this evolved behaviour is the same as the human behaviour however, at an individual level the population of agents still tends to converge to an investment point close to the Nash prediction.

The rationality of the evolved agents is displayed when agents are co-evolved in the baseline game. Similarity to human play can emerge from the evolutionary process in two different ways. Firstly, at an individual level, if irrational agents are introduced to the evolutionary process. Secondly, at a group level, when enough disruption to the environment occurs because of the behaviours of individuals.

For future work, an exploration of the injection of randomness, irrationality or other attributes to attempt to generate human-like behaviours automatically. Other evolutionary goals are added to achieve this, rather than constrain the evolutionary process, to avoid guiding the solutions. An extension of the game environment is planned in order to introduce extra complexity to the problem. Extra dimensions would represent coordination as well as cooperation seen in this game.

REFERENCES

W. Brian Arthur. Inductive reasoning and bounded rationality. *The American Economic Review*, 84(2):pp. 406–411, 1994. ISSN 00028282.

F. Bousquet, R. Lifran, M. Tidball, S. Thoyer, and M Antona. Agent-based modelling, game theory and natural resource management issues. *Journal of artificial societies and social simulation*, 2001.

Damien Challet and Yi-Cheng Zhang. On the minority game: Analytical and numerical studies. *Physica A: Statistical and Theoretical Physics*, 256(3-4):514 – 532, 1998. ISSN 0378-4371.

Alan Cunningham and Colm O'Riordan. A genetic programming approach to an appropriation common pool game. In *ECAL (2)*, pages 167–174, 2009.

P. J. Deadman. Modelling individual behaviour and group performance in an intelligent agent-based simulation of the tragedy of the commons. *Journal of Environmental Management*, 56(3):159 – 172, 1999. ISSN 0301-4797.

J. Epstein and R. Axtell. *Growing Artificial Societies, Social Science from the Bottom Up.* Brookins Institution Press/ The MIT Press, 1996.

Thomas Haynes, Roger Wainwright, Sandip Sen, Ip Sen, and Dale Schoenefeld. Strongly typed genetic programming in evolving cooperation strategies. In *Proceedings of the Sixth International Conference on Genetic Algorithms*, pages 271–278. Morgan Kaufmann Publishers, Inc, 1995.

Wander Jager, Marco A. Janssen, Charles, and A. J. Vlek. How uncertainty stimulates over-harvesting in a resource dilemma: Three process explanations. *Journal of Environmental Psychology*, 22:247–263, 2002.

Nelson Minar, Roger Burkhart, and Chris Langton. The Swarm Simulation System: A Toolkit for Building Multi-Agent Simulations, 1996.

Elinor Ostrom, Roy Gardner, and James Walker. *Rules, Games and Common-Pool Problems.* Michigan, 1994.

EVOLUTION AND ANALYSIS OF STRATEGIES FOR MANCALA GAMES

Damien Jordan and Colm O'Riordan
Information Technology
College of Engineering & Informatics
NUI Galway
Ireland
E-mail: {d.jordan1, colm.oriordan}@nuigalway.ie

KEYWORDS

Mancala Games, Game Artificial Intelligence, Genetic Algorithm, Simulation

ABSTRACT

Mancala games refer to a large family of strategy games. The research described in this paper investigates strategies for the game. We initially identify a useful set of simple heuristics for the game and then evolve a strategy that combines these heuristics in a useful way. Our overall goal in this research is to develop robust strategies that do not require a huge look-ahead for a range of mancala games. In this paper we describe our results obtained on one particular variant, namely, Bantumi. The results are analysed to assess the effect that changing the number of seeds per bowl has on the best evolved strategy. We investigate the weakness of the best evolved strategies and discuss ways in which they can be further improved.

INTRODUCTION

In this work we aim to evolve a robust strategy that can be used in bantumi games. Mancala is a family of two-player board games that are popular all over the world. There are over 300 variants of Mancala games, each with their own board configuration and set of rules. The object of mancala games is usually to capture more seeds than one's opponent.
The game begins with the players placing an equal number of seeds, as per the variation in use, in each of the bowls on the game board. A turn typically involves removing all seeds from a bowl, placing one seed in each of the following bowls in sequence, and capturing seeds based on the rules of the game. The exact rules for a capture vary considerably among the variants. Figure 1 shows a typical mancala board.

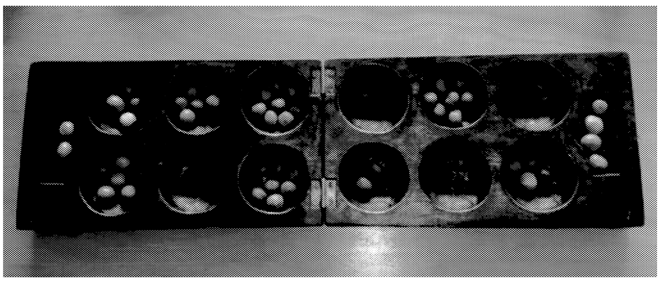

Figure 1: The layout of a typical mancala board

Mancala games are challenging strategy games but yet are governed by relatively simple rules. Some of the existing veriants have been studied in depth with some even having been solved with their complete search space having been examined. However, given the large number of variants, the similarities across these variants and the ease with which new variants can be developed via minor changes to the rule base, it may not be feasible to resort to exhaustive exploration of the rule base to solve these variants. Furthermore, exhaustively searching the space does not necessarily give insight into how humans should best play the game.
Heuristic based approaches with limited look-ahead capability represent a useful approach into gaining insights into more human like strategies for the game. Coupled with evolutionary computation, one can search for useful combinations of heuristics for differing variants of the game. These useful combinations of heuristics can allow us to play the game more effectively.
Many interesting research questions exist in the domain of mancala games. These include: are there winning strategies, that is, does a particular pattern of moves exist that can guarantee a win? For which variants do these strategies exist? Can the application of heuristics help us achieve these strategies? Are heuristics developed for one game transferrable to another? Which changes to the rules change the difficulty?
We hypothesise that a set of useful heuristics can be developed for the game, Bantumi, and that these can be empirically tested to measure their efficiency. We argue that a number of these heuristics can be combined to form a useful playing strategy. We hypothesise that evolutionary computation can be used to learn a robust strategy for Bantumi.
In this work, we develop a simulator for Bantumi. A set of suitable heuristics are then identified. These are then empirically tested to determine the relative strength of each heuristic. We then combine a number of heuristics together to form a playing strategy. Evolutionary computation is used to evolve a robust strategy. We analyse the best evolved strategy to measure the effect of making changes to the game such as, varying the number of seeds per bowl and varying the starting order of the game. We also explore some of the weaknesses of this strategy and discuss some possible solutions.
In the next section we discuss some of the background material that we researched as part of this study. We outline some of the existing approaches taken in the field of mancala games together with a description of evolutionary

computation. In section three, we describe the heuristics used and the model adopted.. We describe the experiments undertaken and discuss the results achieved. In the final section we present our conclusions.

RELATED RESEARCH

Mancala Games

There are over 300 documented variations of Mancala games. Laurence Russ, who has been studying Mancala games since the late 1960s, describes many of these variations (Russ, 1984).
Kalah (more commonly known as Bantumi) is popular in the western world. The board consists of twelve bowls (or pits) arranged in two rows of six, with a score bowl at each end, one for each player. The number of seeds per bowl typically varies from three to six. If the last seed sown in a turn lands in the players score bowl, the player gets another turn. If the last seed lands in an empty bowl, that seed is placed into the player's score bowl along with the entire contents of the adjacent opponent's bowl.
Oware (also known as Wari or Awari) is the most popular version of mancala in Africa. It originates in Ghana and has the same board configuration as *Kalah*. It uses four seeds per bowl and, like *Kalah*, the object of the game is to capture more seeds than ones opponent. However the capturing sequence is different. Seeds can be captured if the last seed sown falls in an opponent's bowl, and brings the total number of seeds in that bowl to two or three seeds. If the second-last seed sown also brought an opponent's bowl to two or three seeds, the seeds in this bowl are also captured, and so on.

Game Strategies

There are many ways to play a game of Mancala (Hanson, 2003). Eason et al. (2000) used a search tree to determine if there exists an optimal opening move that guarantees a win for the player. They identified an opening move for the player going first that had a much higher success rate than all other moves. As part of their findings a number of strategies were identified such as hoarding. This is a strategy where the player builds up as many seeds as possible in one bowl during the game. These are then added to the player's score when the game ends.
Research into the strategies of Awele (Retschitzki, 2000) involved conducting research into the expertise of Awele players in the Ivory Coast in 1984. He tried to identify the processes that differentiate expert from non-expert players focusing on the decision-making processes when choosing the best move in a given situation.
In the 1990's, a closely related variant of Awele called Awari was created by computer scientists. It was solved (Romein et al., 2002) and proven that the perfect game is a draw. Based on this work a public Java applet - the Awari Oracle, was created. The Awari Oracle comprises a database containing the eventual outcome of all the 889,063,398,406 positions that can occur during a game of Awari.
In 2002, the Catalan Oware expert Viktor Bautista i Roca demonstrated that more than 95% of all moves in a game between two master players, Trevor Simon and Ibrahim Abubakar, were perfect according to the Awari Oracle.

Evolutionary Computation

Evolutionary computation (EC) is the collective term used to describe a range of problem solving techniques based on principles of biological evolution such as, natural selection and survival of the fittest. EC operates on a population of potential solutions using the principle of survival of the fittest to extract better solutions. After each generation, a new population of solutions is created by selecting individuals based on how well they perform in the problem domain. These are then subjected to mutation and recombination. This procedure hopefully leads to a population of inviduals that are better suited to their environment than the individuals from which that they were created.
Three of the most commonly used EC approaches are genetic algorithms (GA), genetic programming (GP) and evolution strategy (ES). A genetic algorithm (Holland, 1975) is a stochastic optimisation technique that uses the genetic operators selection, mutation and crossover to evolve solutions to a problem. Genetic programming (Koza, 1992) uses evolutionary search techniques to search through tree structures which represent computer programs. ES (Rechenberg, 1973) uses deterministic selection and normally distributed mutation for each generation. These approaches have been shown in to be very useful in a wide range of domains.

MODEL

A simulator for the mancala game Bantumi was designed and implemented. Seven useful heuristics were identified for Bantumi through a combination of background research (Donkers, 2002), (Hanson, 2003), (Eason, 2000) and thorough game play. We monitored our own patterns while playing the game, and made notes based on our observations. All players, after playing numerous games, will repeat patterns that worked successfully for them in previous games. We therefore assumed that, if these patterns can be of benefit to human players, then they can also be of benefit to an AI player. The seven heuristics are:

- H1: Pick a bowl that allows the player to have another go.
- H2: Pick a bowl that allows the player to make a capture.
- H3: If the player's opponent has seeds in a bowl that will allow him another go, disrupt it.
- H4: If the player's opponent can capture some of the player's seeds on the next go, move the vulnerable seeds.
- H5: Always pick the bowl closest to the score bowl.
- H6: Avoid picking a bowl that, after sowing, results in giving the opponent another go.

- H7: Avoid picking a bowl that, after sowing, results in allowing the opponent to capture some of the player's seeds

Initially we planned to use just five heuristics (H1-H5). We decided to increase this number to seven to broaden the robustness and scope of the strategies. Other heuristics were also considered for inclusion. One such example was: 'how close am I to winning' (count the number of seeds in the score bowl after each possible move). This was discarded as it was felt that it was a more of a 'machine' way of thinking than a human. Other heuristics were discarded as they were considered too complex to implement.

EXPERIMENTS AND RESULTS

Testing Heuristics

Our first set of experiments involved a round-robin tournament to determine how strong each heuristic was in comparison to the others. For these experiments both players picked their bowls by using the 7 heuristics, one at a time. If the application of the heuristic being used was unable to choose a suitable bowl, the bowl would be chosen randomly.

The results of these experiments showed that H1 and H5 were the two strongest heuristics of the group, while H3, H6 and H7 were the weakest. It was decided as part of the round-robin testing that each heuristic would be tested directly against themselves (H1 vs H1, H2 vs H2.....). This provides an opportunity to observe the effect of having the first move in a game. The performance of H1 against all other heuristics is shown in Table 1. The top row shows the seed per bowl combination and whether the player using H1 had the first or second move of the game (3 – F: 3 = seeds per bowl, F = first move).

	3 - F	3 - S	4 - F	4 - S	5 - F	5 - S	6 - F	6 - S
H1	50	52	54	48	66	30	54	47
H2	69	63	74	70	76	77	65	62
H3	50	50	76	79	89	79	66	53
H4	61	60	73	72	83	79	85	73
H5	1	0	23	36	83	20	23	40
H6	62	61	69	64	75	66	86	71
H7	56	52	80	77	81	82	78	83
Rand	72	73	82	80	94	89	87	69

Table 1: Win % of H1 vs other heuristics

From Table 1 we can see that when the heuristic H1 was played against itself there was a noticeable advantage in having the first move of the game. The first move of the game was also a benefit to H1 when played against H2, H4 and against a random strategy. Against other heuristics the advantage was not always quite so obvious. However, in the majority of cases having the first move was an advantage.

Other heuristics in the group had a slight advantage when having the first move of the game. H2 on the other hand had more success when moving second than when moving first. An interesting observation was made when H5 (always pick the bowl closest to player's score bowl) was tested against itself. It won 50% of games when 3 and 4 seeds per bowl were used. Alternating the starting position had no effect on this score. When moving first, using 5 and 6 seeds per bowl, it lost all games. When moving second, it won all games.

Another observation that was made was the performance of heuristics in relation to the number of seeds per bowl. Heuristics that performed well when using 3 seeds per bowl generally also performed well when 6 seeds per bowl were used. Also, heuristics that performed well for 4 seeds per bowl also performed well for 5 seeds per bowl. This trend was particularly evident for the heuristics H2 and H3.

Testing a combination of heuristics

Following on from the testing of each heuristic in isolation, we proceeded to measure the effect of combining two or more heuristics together. For this purpose we decided to group together the heuristics H1, H2, H4 and H5 in a linear order to form a single strategy. The motivation for choosing these heuristics was based on the early results of the round-robin tests which showed that H1 and H5 were the strongest of the group, while H2 and H4 were also quite strong. The reasoning behind placing them in the chosen order above was based on observations of how we played the game.

This combination of heuristics was shown to win an average of 83% of games when played against all other heuristics. The results of this experiment are shown in Figure 2. The trend described earlier of heuristics performing well for 3 and 6 seeds per bowl or 4 and 5 seeds per bowl is extremely evident here. The combined heuristic performed much better for 4 and 5 seeds per bowl than for 3 and 6 seeds per bowl.

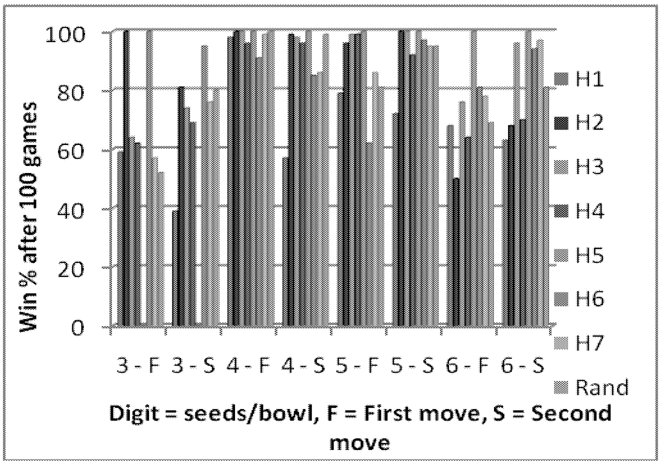

Figure 2: Performance of the heuristic combination H1, H2, H4 and H5 against individual heuristics.

Evolving Strategies

A genetic algorithm (GA) was designed and implemented in the simulator in order to evolve robust playing strategies. The population size was set to 50 chromosomes. The seven heuristics listed earlier were numbered from 1 to 7 and arranged in a random linear order to form a chromosome. A typical random chromosome from the population could look like 3415726. This particular pattern means that, during a game, the AI player will first use the heuristic H3 to pick a bowl for each move. If the application of this heuristic fails to choose a suitable bowl (the condition of the heuristic is not met) the AI player will then use H4. If H4 is unable to pick a bowl the AI player will continue to move onto the next heuristic in the chromosome until a bowl has been picked.

The fitness each individual was determined by calculating the average number of wins by that chromosome after playing a series of games against all other heuristics in isolation. In other words the pattern in which the heuristics are arranged for a particular chromosome is used as the strategy for player 1. The strategy used by player 2 is to use each heuristic, one at a time, for each game played against player 1, until all heuristics have been used.

The simulator was set to run 50 generations of the GA on each execution. After numerous generations, and millions of games played, a strategy has evolved when using 3 seeds per bowl that wins an average of 95.9% of games when played against all other heuristics. The graph in Figure 3 plots the number of games won by the fittest individual after each generation for a typical run of the simulator when using 3 seeds per bowl. The fittest individual after the first generation won just 86% of games. However after 50 generations the fittest individual won an average of 95.25% of games.

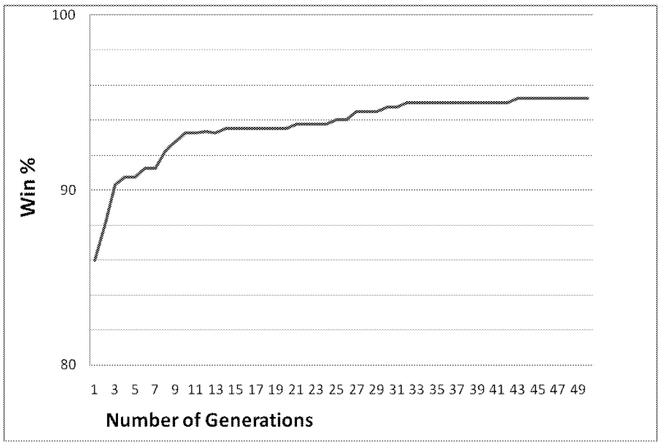

Figure 3: The fittest individual after each generation using 3 seeds per bowl.

When using 4, 5 and 6 seeds per bowl, strategies have been evolved that win an average of more than 99.1, 98.4 and 97.8 % of games respectively. The best evolved strategies for each seed per bowl combination, along with the percentage of games won and total number of seeds won are shown in Table 2.

	3 seeds	4 seeds	5 seeds	6 seeds
Best Strategy	3145672	1345672	1342576	1342567
Average wins	95.9	99.1	98.4	97.8
Seeds won	2048	2710	3440	4221

Table 2: The most robust strategies evolved against individual heuristics for each seed per bowl combination.

Each of the best evolved strategies shown in Table 2 were evolved by playing against all of the seven heuristics one at a time, including the random heuristic. In order to explore the landscape and to determine if there existed more effective strategies than those evolved, we evolved a new population of strategies by playing against the best evolved strategies in Table 2. Our fitness function was changed to be the number of wins obtained against the best evolved strategies from the previous evolutionary experiments. After 50 generations of the GA the strategies shown in Table 3 were evolved.

	3 seeds	4 seeds	5 seeds	6 seeds
Best Strategy	1543762	1342657	1325476	1342567
Average wins	99.4	98.2	95.6	93.4
Seeds won	2136	2803	3281	3816

Table 3: The most robust strategies evolved against the strategies from Table 2 for each seed per bowl combination.

Analysing Results

Having evolved strong strategies for each seed-per-bowl combination, we investigated ways to further understand the effectiveness of these strategies. For example, the best evolved strategy when using 4 seeds per bowl wins a total of 99.1% of games. This means that there are 0.9% of games that it loses. We decided to analyse these situations where the evolved strategy lost games to try to determine why those games were lost.

Our investigation began with the strategy evolved for 4 seeds per bowl. The bowls that were picked by the non-evolved player were noted for games where this player won against the evolved player. These games were then re-played, bowl by bowl, and studied in detail to determine how seeds were won. An observation quickly became apparent – a number of games that were won by the non-evolved player were achieved by using a tactic called hoarding (Eason, 2000). This is a tactic where seeds are built up in a player's bowl during a game and when the game ends these seeds are added to the contents of the player's score bowl. A series of tests were carried out to determine the percentage of games lost as a result of hoarding by the best evolved strategy for each seed per bowl combination. The results of these tests are shown in Table 3.

	3 seeds	4 seeds	5 seeds	6 seeds
Best Strategy	3145672	1345672	1342567	1342567
Hoarding %	73.0	83.5	69.3	59.0

Table 3: The number of games lost by each strategy due to the hoarding of seeds by the opponent.

We can see that hoarding by the non-evolved player has the biggest effect on the strategic player when using 4 seeds per bowl. However the strategy in this case is already extremely successful – it wins 99.1% of games. Therefore the high percentage of hoarding is not having a major impact on the overall performance of this strategy. The weakest strategy is the one evolved when using 3 seeds per bowl – it loses 4.1% of games to the non-evolved player. 73% of these 4.1% of games, or almost 3% of all games lost by this strategy are due to hoarding.

Further analysis was undertaken on the games that were won by the non-evolved player that did not involve hoarding. The results of this analysis showed up some weakness in the design of the heuristics H3 (If the player's opponent has seeds in a bowl that will allow him another go, disrupt it) and H4 (If the player's opponent can capture some of the player's seeds on the next go, move the vulnerable seeds). It was observed that the application of H3 could occasionally present the opponent with the opportunity to pick a bowl that will give him another go or to capture some of the player's seeds. To understand how this happens consider the following situation in the middle of a game between two players. The opponent has the correct number of seeds in a bowl to allow him another go. He also has a bowl that is one seed short of the requirement to have another go. The player applies the heuristic H3. While the opponent's bowl that initially had the correct number of seeds for another go is disturbed the bowl that was one seed short now has the correct number to give the opponent another go.

The weakness to H4 appears in a situation where a capture opportunity becomes available for the opponent after he has picked a bowl that gave him another go. The bowl that was picked to give the opponent another go becomes empty, meaning that the seeds in the player's adjacent bowl can be captured if the last seed sown by the opponent falls in this bowl.

The above weaknesses in our heuristics demonstrate the possibility of further tweaking that could be done to increase the overall effectiveness of the strategies. One such method would be to create some new heuristics. A combination of the heuristics H3 and H4 could be used to overcome the weakness shown in H4 above. The combination would work as follows:

- Check if the opponent has seeds in a bowl that will allow him another go
- If such a bowl exists, temporarily sow the seeds from the opponents bowl
- Now apply the heuristic H4 – check if the opponent can capture some of the player's seeds

The weakness to H3 could be overcome by using the following method:

- Check if the opponent has a bowl that is one seed short of the requirement to allow him another go
- If such a bowl exists, check if the application of the heuristic H3 will increase this bowl by one seed
- If it does, do not apply the heuristic H3

It should be considered that this solution to H3 might not improve the overall performance of the strategy as it reduces the number of situations where the heuristic can be applied.

These proposed modifications to the strategies require a greater complexity in terms of 'look-ahead'.

CONCLUSION

This paper explores the use of heuristics and a genetic algorithm to evolve playing strategies for mancala games. The inspiration for this study is to analyse techniques for playing strategy games that can be of value to the games industry. We start out by designing and developing a set of heuristics for Bantumi. These heuristics are then used to form a playing strategy for Bantumi. We then use a genetic algorithm to evolve a set of robust strategies for each combination of seeds per bowl for Bantumi.

The results have shown that our chosen heuristics provide an effecient mechanism in allowing the AI player to play the game more effectively. The stronger heuristics have been identified from the weaker ones. We have identified weaknesses in some of our heuristics and we have proposed methods on how these weaknesses can be overcome.

Our results have also shown that the strategies evolved for each seed per bowl combination differ. We can therefore verify that the optimal strategy required to play the game changes as the number of seeds per bowl varies. More testing is required to further evaluate the robustness of these strategies.

We checked whether or not an advantage exists for the player who has the first move in the game. Our results have shown that in the majority of cases an advantage does exist. There are however some strategies where it is an advantage to have the second move. This is due to the design of some of the heuristics used.

In the future we aim to introduce changes to the game rules and board layout and further analyse the robustness of our strategies and methods. These changes will in effect focus our studies on a new mancala variant. This will provide an opportunity for us to measure how quickly and effectively we can evolve dominant strategies for this new game.

We may also explore the potential of using co-evolution to evolve new more-robust strategies. In this situation we would use the previously evolved strategies as the opponent for the fitness function in our GA. The effect of this would mean that each newly evolved strategy would have to win games against increasingly more difficult opponents, which should in turn lead to the evolution of more robust strategies.

REFERENCES

J. Donkers, J. Uiterwijk, A. de Voogt. "Mancala Games – Topics in Artificial Intelligence and Mathematics". *Editions Universitaires, Fribourg, Switzerland,* 2002.

H. R. Eason & Adam Brandenburger. "How to Play Mancala". *Harvard Business School*, 2000.

S. F. Hanson & John S. Hanson. "All about Mancala: Its history and How to Play". *Happy Viking Crafts Mahomet*, Il. 61853, 2003.

J. H. Holland, "Outline for a logical theory of adaptive systems,". *J.Assoc. Comput. Mach.,* vol. 3, pp. 297–314, 1962.

J. H. Holland. "Adaptation in Natural and Artificial Systems". *University of Michigan Press, Ann Arbor*, 1975.

J. R. Koza. "Genetic Programming: On the Programming of Computers by Means of Natural Selection". *MIT Press,* 1992

I. Rechenberg. "Evolutionsstrategie: Optimierung technischer Systeme nach Prinzipien der biologischen Evolution". *Fromman Holzboog*, 1973.

J. Retschitzki. “Strategies of Expert Awele Players”. *Proceedings of the International Colloquium Board Games in Academia III. Florence, 84-94*, 2000.

J. Retschitzki, A. J. de Voogt & F. Gobet. “Moves in Mind: The Psychology of Board Games.” *Psychology Press Ltd, Hove UK*, 2004.

J. W. Romein & H. E. Bal. “Awari is solved”. *ICGA Journal; 25 (3),* 2002.

Laurence Russ. “Mancala Games”. *Reference Publications Inc*, 1984.

HISTORICAL ACCURACY IN GRAND STRATEGY GAMES: A CASE STUDY OF SUPREME RULER: COLD WAR

B. Srivastava and M. Katchabaw
Department of Computer Science
The University of Western Ontario
London, Ontario, Canada N6A 5B7
bsrivast@csd.uwo.ca, katchab@csd.uwo.ca

G. Geczy
BattleGoat Studios
Lynden, Ontario, Canada L0R 1T0
george@battlegoat.com

KEYWORDS

Historical accuracy, grand strategy games, video games

ABSTRACT

Historical accuracy is an often overlooked and understudied topic in the study of realism in video games. For some games, however, this topic is both an extremely interesting and important one, quite deserving of attention.

In this paper, we investigate many of the issues and challenges of historical realism in video games, with a focus on strategy games. In particular, we examine these issues and challenges with reference to Supreme Ruler: Cold War, developed by BattleGoat Studios, providing both researcher and developer perspectives.

INTRODUCTION

Realism in video games is always an interesting topic for discussion. Whether the topic is the visuals, audio, artificial intelligence, story, or gameplay, every designer has their own take on balancing fun, marketability, and accuracy (Moreno-Ger et al. 2008).

In this paper, we investigate the challenges of historical realism in strategy games, and we compare the pre- and post-launch developer perception of the balancing choices made[1]. In particular, we present a case study of the choices made to depict the world in Supreme Ruler: Cold War (SR:CW), developed by BattleGoat Studios, currently scheduled to be published by Paradox Interactive in Q3 2011 (BattleGoat Studios 2011). At the time of this writing there have been 22100 game start-ups, of which 10910 were of a pirated version (which may not fully work since it has a forced end date). Our observations were primarily taken from forum posts and e-mailed bug reports.

BattleGoat's two previous titles, Supreme Ruler 2010 (BattleGoat Studios 2005), and Supreme Ruler 2020 (BattleGoat Studios 2008) were set in the future relative to their release dates, but still required much historical accuracy. Units, terrain, political maps, even technologies had to tie into real world work that has been done, or was known to be worked on. Fortunately, in both of those games, from the moment the player starts, the world can diverge fairly rapidly from a historical path, but developers do not always have that luxury.

In this paper, we examine a variety of realism and accuracy choices made in historical strategy games, organizing them into several categories: cosmetic, strategically important, balance, and legal or social issues, and discuss the various trade-offs in each area. Throughout this discussion, we present a short case study of the perspectives of how the designers of SR:CW made their choices, and some of the problems encountered. Many of the cited examples of other similar grand strategy games tend to also be conveniently published by Paradox (Paradox Interactive 2007, 2009, 2010). The well-known Civilization series (most recently Firaxis 2010) falls in a similar style, but is less historical.

The remainder of this paper is organized as follows. We begin by discussing various categories of historical accuracy. Using this categorization, we examine the issues of realism and accuracy in grand strategy games, drawing upon SR:CW as an example. We then conclude by discussing these issues from the developer's pre-launch perspective as well as post-launch commentary once it is available.

CATEGORIES OF HISTORICAL ACCURACY

To study and assess historical accuracy in a game, we first need a simple way to define different types of adherence to accuracy based on how it impacts the game. We decided to view these design choices as being in one of four broad categories.

First are cosmetic changes. These are things that do not have a significant impact on gameplay; in

[1] BattleGoat and Paradox released Supreme Ruler: Cold War July 19th, but it was not fully in retail channels by time of final submission.

other words, if these things were changed, the game plays no differently for the player.

Second are strategically important issues. These can impact gameplay in a meaningful way, encompassing ideas that directly tie into the game's model of the world, and how that world is going to behave. Players should be presented with a believable historical world. However, real systems, such as diplomacy, involving written and unwritten agreements are very complex. Approximations are usually made for playability. A simple example can be found by examining Article 5 of the NATO treaty, which reads in part "an armed attack against one or more of them in Europe or North America shall be considered an attack against them all" (North Atlantic Treaty Organization 1949). An attack on French Guiana (the main European Space launch facility) is not apparently covered, but an attack on Saint Pierre and Miquelon (some small French fishing islands off the Canadian Coast) would be. Given the complexity of both understanding and modelling such treaties approximations are made. One option is a limited or colonial[2] war (Paradox interactive 2003, and 2009), or an all or nothing approach, where war is war, and fully activates alliances as is done in pretty much everything else, including previous Supreme Ruler titles. SR:CW attempts to create a more realistic situation, where countries may be funding an insurgency, or conducting spy mission incursions (naval or air) without a declaration of war. Countries being incurred on can fire on neutral incursions or not, which may or may not trigger a war; the player can choose the nature of their responses to this. The all or nothing approach of previous Supreme Ruler titles was also exploitable or could cause accidental wars when a player inadvertently moved a unit into the wrong place, which is also ahistorical (and especially troublesome when it happens with a friendly state).

Third is balance, or systems design. How does one decide on the statistics of a Panther tank compared to a T-34, or a Queen Elizabeth class battleship and a Queen Elizabeth class aircraft carrier? Here, regardless of developer intention or effort, it is likely not possible (in a grand strategy game) to be perfectly authentic to those units. A tank or a ship simulator, for example, might have a more direct, more authentic model, but the combat model in a strategy game necessarily requires abstraction. Otherwise, the game quickly becomes unwieldy, unplayable, and generally not enjoyable for the player. The Total War Series (e.g. Creative Assembly 2011) solves this problem by having both a real time strategy game, which is reasonably direct, and a grand strategy game together.

Fourth are legal or social choices. While these may also fall under other categories to some degree, they require special attention (Rosenthal 2009). After all, if a developer wants to market and sell its game to a profitable level, it needs to follow the rules and norms of society, even if those rules require compromises in historical accuracy, especially in certain locales or jurisdictions (China Daily 2004).

With this categorization in hand, we now delve into each area in more detail.

COSMETIC ISSUES

As discussed earlier, cosmetic areas of the game are things that have no real gameplay impact. People may feel strongly about them or not, but changing them one way or another will not impact gameplay.

A few examples come from something as seemingly benign as country flags. For instance, Switzerland uses a 1:1 aspect ratio flag, whereas nearly everyone else uses a 2:3[3]. This has implications on a game's user interface, requiring that interface layout needs to be setup to handle a small number of oddly shaped flags (Nepal uses two triangles), that artists need to pad the art around it, or that players will need to cope with some minor interface deformities. For SR:CW, the interface handles a 1:1 flag properly, though it is never explicitly made clear to the player why only one flag is like this, and in many cases people think is a bug.

Other flag issues depend on time. For instance, SR:CW starts in 1949, a country such as Canada, which chose its current flag in 1965 can confuse the player by using a historical flag, that looks nothing like the current one, but has no particular significance. The expectation, created to some degree by French and German flags, as they have changed over the years, is that different flags reflect vastly different governments and ideals. In the case of France the modern tricolour is a strongly republican symbol, and would not be appropriate for Bourbon France for example. East and West Germany pose a more interesting problem, having, from 1949 until 1959 the same flag, at which point East Germany added a hammer

[2] Limited and colonial war can be different as well. A limited war would be where allies are not explicitly called, and colonial would only allow fighting over colonial areas or the like.

[3] Federal Department of Home Affairs FDHA White on red: the Swiss cross http://www.bar.admin.ch/archivgut/00591/00601/00603/index.html?lang=en

and compass decal to theirs. Using the historical flag can cause some confusion, but changing it realistically later would be legitimate learning experience and appropriate flavour. After testing SR:CW, it was found preferable to start East Germany with the later (post 1959) flag, because it was too difficult to distinguish between units of each Germany. In a similar fashion, the USSR and Peoples Republic of China used very similar flags, a red background with a yellow decal in the top left corner. They use slightly different reds, and decals, but, especially as small individual unit flags, which are just shrunk down versions of the national flag, they are basically indistinguishable. This issue has not yet been resolved to anyone's satisfaction.

Another area in which developers face cosmetic choices is the selection of place names. For SR:CW the choice is to use conventional short forms, such as United Kingdom, rather than United Kingdom of Great Britain and Northern Ireland, and so on. Problems arise in contested areas, Jerusalem or Al Quds, Falkland islands or Islas Malvinas, and so on. Since our primary market is in English we use the conventional English form. Further problems are presented by certain places such as the Côte d'Ivoire, which lists its country name as such, even in English[4]. On the other hand, the Supreme Ruler series typically uses the English transliteration local form for city names, such as Beograd rather than Belgrade, Warszawa rather than Warsaw and so on.

As with city names, SR:CW encountered issues with the names of some historical persons. The text processing engine used in-game reads in first and last names. Faisal II, the last King of Iraq, posed a particular problem because he used only one name, and was the only leader listed during the time period for the game to do so. As it turned out the, text parser failed to properly parse a single name, and at release Iraq appeared to be leaderless. The easiest solution is to simply put his name in the database as first name Faisal, surname II.

Pictures of real people pose another problem. SR:CW used a combination of custom portraits for various leaders (most of the major national leaders), and generic leaders for other countries, as well as all ministers. This leads to several problems. First, if the generic portraits assigned to neighbouring countries happen to be the same, players not familiar with the area can get confused. Second, some leaders had carefully crafted public images, and a generic portrait would not accurately reflect that. Lastly, some leaders, for example Mohammed Omar of Afghanistan (the leader of the Taleban) have worked very hard to not ever have a photograph or portrait done on religious grounds. In this case it would be fairly easy to have a shadowy figure portrait to not offend religious sensibilities, or use a generic best guess.

These cosmetic choices at the start lead to another problem, which is how to change or update things appropriately. In the Paradox self-published titles (Paradox Interactive 2007, 2009, 2010 etc.) along with SR:CW have scripting systems in place that as long as certain conditions are met, the flag will change believably, either to the historical flag or one that believably makes sense (for example, a communist British flag). Changing city names to reflect different ownership is again a system that can be scripted in, but has little overall value, though it does affect the historical accuracy of the game in some areas.

STRATEGIC IMPORTANCE ISSUES

Issues of strategic importance are changes that could substantially impact gameplay or historical accuracy. Much of this in essence revolves around the models and approximations used in any game, though for us are largely political and economic. Trying to boil down the industrial base of a country into "industrial capacity" or splitting production into civilian, industrial, and military and so on, are attempts at simplifying the real world for playability. These simplifications, however, run counter to accuracy, and so there are interesting and challenging trade offs that must be considered.

Geo-Political

Starting with the geo-political area, in our case of SR: CW, the first and biggest questions are the Soviet satellite states, the decolonization areas, and then parts of West Germany, specifically, the areas under French occupation that the French were trying to keep. In some of these cases the game must attempt to balance between modelling a territory that will be independent shortly, from a territory that is independent. For example, Canada and Australia are separate from the United Kingdom, but Kenya and Uganda, in 1949, were not. Indonesia was, at the start date of SR:CW (October 9, 1949) still a colony of the Netherlands legally, but in practice it was much easier to just make it fully independent and forget about the colonial struggle which the Netherlands was going to end in less than a year anyway. Places that were clearly colonial (for example Kenya and Uganda), as well as the Soviet satellite states, required a specific political model where they were separate, but limited in what they can do diplomatically. One of the first fan mods for SR:CW that we saw was to put all of the Soviet satellites into one big

[4] CÔTE D'IVOIRE http://www.iso.org/iso/country_codes/iso_3166_code_lists.htm accessed June 15 2011

USSR nation, and it seems to be quite a popular theme. This is clearly an area where some fans have disagreed with the original SR:CW model of the world.

The messy problem of the French occupation zone in Germany, and areas they wished to keep in this time frame is a difficult one. Unlike Indonesia, which was to be independent immediately, the Saarland was, until 1956, an area the French were trying to administer. It is also important, because the Saar is a major source of both coal and steel, and so determining who gets control can have fairly significant implications. We decided, though without much conviction, and with much disagreement, to put the Saar directly under the control of France. It is too small to justify as its own state (which then needs some way to be peacefully absorbed into West Germany), but after assessing balance we think it should be under Germany, something for a path. Some other territorial disputes, for example between Canada and Denmark, have little to no value even today, and the squabbling over them is mostly cosmetic and low intensity.

Making sure important places are on the map is an interesting problem. Gibraltar is just large enough to make one hex on the SR:CW map (supplied by NASA), where each hex is about 16x16km. Portuguese Macau and Goa, and British bases on Cyprus, and a few others are simply too small to show up. Yet as naval bases or air strips or the like they could prove quite valuable, especially in a situation like the Falkland's war. It is an interesting challenge to balance between representing something that is theoretically significant, while at the same time not bogging down the map with little dots of colonial history that in practice will not matter. The SR:CW map, while provided from NASA still needs to have terrain manually painted on with our tools. Satellite maps have river enhancement techniques, but we still miss several water bodies. . In all three Supreme Ruler releases, Rio Negro in Argentina has been completely missed, despite it being relatively important locally. None of the battles researched for the Supreme Ruler series mentioned it, and it does not show up on the satellite, even with enhancement for it to have been noticed when terrain painting[5]. Adding it in would require bridges placed across it historically accurately, and it is not readily apparent how militarily valuable it is. We tended to focus our attention on areas of the world that are most likely to be reflected in sales, or major conflicts.

[5] Thanks to Paradox Forum user *sonofliber* for repeatedly pointing this out.

Resources are another area that has received much discussion. Historically placed resources make for an interesting problem. The player knows Iran, Saudi Arabia, parts of Canada, the coast of the United Kingdom and so on all have vast, but then undiscovered, oil reserves. If they were put in game as we know them today but in 1949, the world should play out rather differently than if there was no way to plan for the future. On the other hand, not putting the resources in the game at all changes how the world would evolve. In the case of SR:CW and in general for historical games, the choice seems to be to have resources in reasonably historical locations and quantities (model permitting). In Europa Universalis III unpopulated (by Europeans at least) colonies do not get a resource they produce until colonized, with a random selection based on historical resources from the general area. That model keeps the macro-economic situation the same overall, presents the player with legitimate historical dilemmas (colonize A, or B, not knowing which will be more valuable), and adds some differences in replay value. In a game where the world is both defined and settled in advance, that could be somewhat problematic, especially with undiscovered major resources rarely on the borders between principle states.

The last really interesting geo-political phenomena are in the actual cold war, and how to model proxy fights, spheres of influence and so on. While we think insurgency is an interesting topic for the future, a simple model can at least accurately reflect the strategic effects, funding insurgents sows discontent and can change governments. Satellite states and colonies are bound in some way to their parent, and have resources drawn away. They become undesirable to keep when the resources they produce are not valuable enough. Countries can engage in proxy wars and lower intensity conflicts, fund insurgents, and spur political discontent.

Internal Political

With the world containing thousands of political systems, all with an ever-expanding slew of written and unwritten rules, it is simply not possible to try and correctly model country leadership everywhere, all at once. The player is empowered to make choices, even as a democratic state, much like a dictator. It is not much of a game if the player cannot make choices, and in the real world, making even a handful of meaningful choices can define a career. Rather than games viewing the politics as built from the people up, it has tended to be from the perspective of "I am the State" down, where automation serves to reduce micromanagement for the player rather than acting as a realistic counterweight to their desires. Here there are not likely to be any surprises with SR:CW. The model from previous games in the series, of cabinet ministers who can be variously tuned to automate in different ways, remains. It seems like a good compromise: automation for those who want in, with some granularity, and total control for those who do not. This does tend to mean, however, that a player playing as the United States is running the country more like Stalin than Truman, which is quite interesting and ironic from the perspective of historical accuracy.

SYSTEMS AND BALANCE ISSUES

Where the previous section, on strategic impact, tends to concern larger, more abstract concepts, there are detailed problems as well, that may balance against each other and so on. This area is very much about systems and balance and, in this paper, we focus mainly on two areas. First is units, which broadly encompasses any sort of combat or transport item in the game. In their most simplistic form these units have a cost and a power, both of which need to be decided on. Most challenging is defining the value of utility; for example, what is it worth to be able to carry another unit, or to carry three missiles rather than four, and so on. Secondly, we examine technology, a system which itself is highly abstract, but again, must attach specific numbers for cost and effect. Following these issues, we briefly touch on balance issues in economics and multiplayer gameplay.

Units

Balance is a non-trivial problem at the best of times (Carpenter 2003). Units have some cost, and some statistics defining their various attributes, abilities, and so on. Different combat systems naturally necessitate different statistics. In the case of SR:CW, units have a cost in manpower and money, production time (which is, in turn, resources, which are also money), and they occupy a limited supply of construction slots. They have several statistics and capabilities depending on the unit.

Hopefully the idea of *construction slots*, say in a shipyard, addresses some of the weird behaviour that can come about from a single industry system, where an area can switch from building a tank division to a battleship in a heartbeat. It also reflects national assets; this shipyard, aircraft factory, and so on use the rest of the more abstract industrial base, but are themselves real things which can be captured or destroyed, just like in the real world, and there are consequences that come with that.

Because our units and structures are based on the real world we are faced with a simple problem: the world is simply not balanced. Sometimes, for the same cost, one thing is just better than another. Some places simply cannot, did not, or will not build a particular type of unit (notably big ships), even though their competitors might.

The choice in the Supreme Ruler series is to go with as accurate a model as possible. Unit statistics try and reflect their real world values. Other games use more abstract notions (for example in Hearts of Iron (Paradox Interactive 2009) giving units hard and soft attack values). In the Supreme Ruler case, an F15 should reflect the real combat value of an F15. In the Hearts of Iron case, the type of unit is a cosmetic name and all third generation medium tanks for all countries have the same base statistics. This is again a simplification, and one that runs counter to historical accuracy, but makes it very much easier to balance.

Chemical and biological weapons deserve special attention (Outpost Gamez 2011). For our purposes, we are not interested in non-lethal or incendiary agents. Incendiaries are part of the regular unit model, and non-lethals do not require any special systems to implement for grand strategy. Strategically, since the end of World War 2 lethal chemical weapons were only used to much effect in the Iran-Iraq war. Despite vast stockpiles, neither prohibited[6] lethal chemical, nor biological weapons saw much use from major powers (Bismuth et al. 2004). Strategy games simply do not model phosgene grenades or cluster munitions directly, though those are also prohibited weapons. Because of their relative lack of overall significance, and unknown military value, BattleGoat chose specifically to not include them. One might expect these to also fall into the legal section, though the Victoria series of games (Paradox 2010 most recently) have chemical weapons at a strategic level without issue.

Technology

For simplicity, research and technological development in games tends to have some sort of centralized set of limits and focuses (say research centres, each researching on technology at a time), which is obviously not how real research works at all. But the model established over the years, of some centralized research plan in nearly every strategy game gives the player control and influence. This poses a number of challenges.

[6] The Geneva protocol, 1925, and the Chemical Weapons Convention, 1993, describe prohibited and allowed agents.

Dramatic new technologies (and implementations) should presumably be expensive. If they are inexpensive, clearly worth development, or "stack" in some way, a clever player can exploit the system. Much like historical resources, a player that knows fusion reactors are just a few technologies away, or the best source of oil is Saudi Arabia will play differently than if they did not have such future knowledge, and the AI cannot exploit that knowledge to disadvantage the player too much.

Balancing technologies against each other is another challenge. A technology that increases electricity from oil by 5% has some cost compared to a technology that increases electricity from nuclear by 5%. Here there is little historical precedent to go on. The Manhattan Project and the space programmes are some of the few technologies that have had clearly, publicly defined goals and costs (Charette 1996). However, all of the intermediate technologies involved in those developments are harder to define and quantify. The costs also have to tie into other systems. To use the current Queen Elizabeth Carriers in the United Kingdom as an example, how do you separate the research and development costs from the actual building? How much is going on in parallel? Examination of budget documents might reveal the spending difference, but not how much is in parallel, or how well the costs can be spread out by adding more build orders, such as a third carrier, or the change in cost of reducing the order to just one ship. If there is a model of industrial capacity or military goods how does one factor those into the costs for one of the ships? Unfortunately the answers here are implementation specific and deeply tied to fairly elaborate technology trees. Research costs should reflect the value it immediately creates, the research it opens up, and if it is separate from building an actual thing, then the costs must be reflected in multiple places. If research is tradable (or can be stolen), which is itself an approximation, then you are also trying to factor that in as well, and avoid what, in previous Supreme Ruler titles was affectionately dubbed "trading your way to the 21st century" where a crafty player could accumulate all of the research known by all the artificial intelligence-controlled nations right after starting.

Economics

Games tend to use their own model of industry, goods, and so on. Those systems tend to stand on their own. In this section, we will look at concepts that map to the real world, notably in terms of currency and debt.

Currency is a somewhat abstract concept. For gameplay and programming reasons, it is much easier to simply have one currency in a game. The real world of exchange rates is complicated, unless everyone is on the gold standard, but then gold becomes the currency. The problem, especially for SR:CW in the Cold War time period, is that currency values were governed by complex agreements (Bretton Woods for example), in addition to central banks intentionally valuing and devaluing currency, with fixed, and in many cases inappropriate exchange rates. The cost of trying to understand these systems well enough to model them is rather prohibitive.

Debt was an issue not expected in SR:CW. It seems appropriate in this age of austerity to discuss though. Previous titles modelled national debt in an abstract but reasonably authentic way. Countries in previous tittles had reasonably correct GDP, debt, and interest rates. Attempting to do the same in SR:CW posed some issues. While having correct GDP is straightforward, debt and interest are not. For many countries, notably colonies, their debt situation is quite complicated. They may or may not inherit a portion of their parent on independence; they may have their own debt initially; and so on. This also assumes we could find the relevant data at all, or it even existed (some of Germany's debt agreements were signed after the start of the game, for example). Coupled with this is the system of debt repayments between governments, which is and was quite complex.

Using historical debt levels was found to cripple gameplay, and calculating interest rates to vary over time with what was happening was quite hard. To try and get any source of money to move forward, the game essentially degenerated to trying to disband the army as quickly as possible, which is neither realistic nor fun for the player. It would reflect the overall strategy of demobilization, but did not capture reconstruction or economic expansion well. For launch, SR:CW erred on the side of giving the player more choice, and simply wiped every countries' debt. This, interestingly enough, creates its own problems. For example, the United Kingdom has a much larger GDP than Russia, with France close behind, and no particular reason to weaken themselves militarily in this scenario. As a result, with no debt in place, it becomes quite a challenge for Russia to catch up to the United Kingdom or stay ahead of France. Eliminating debt significantly shifts the balance of power, but modelling it properly proved prohibitively difficult to keep the game both fun and accurate. To reflect a relatively gradual reconstruction, SR:CW creates an artificial shortage of industrial goods, which are needed to make factories to produce both more industrial goods and other things.

Multiplayer Balance

In multiplayer gameplay, it is difficult to find multiple countries that one could say are “balanced” against each other. The closest three are probably France, the United Kingdom, and Italy all having comparable GDP and populations, with Turkey, Spain, and Germany being close, but outliers. Another possible pair is Egypt and Ethiopia (Central Intelligence Agency, data for 2011). There are a few others, but if you want them to be reasonably close to each other geographically, there are relatively few countries that could be called balanced based on population, GDP and resources. This creates an unfortunate trade off, as an equal balance provides more fair and enjoyable gameplay to players, but providing this balance would by historically inaccurate.

Starcraft 2 (Blizzard 2010), while not a historical game, deals with many of the challenges faced in balancing units. Some of these changes are discussed on the official Blizzard forums[7], some not. Testing if a unit is balanced is not trivial, and may involve things such as automated testing, or analysing real world player data. It is important to note that two sides (in the case of Starcraft, all 3 sides) can be balanced, but have individual units which are not, and those units will tend to be over, or under represented. A faction can also be deficient at a specific set of circumstances, on specific terrain for example.

There are numerous ways to assess balance, either through simulation, automated testing, or play testing. The goal with balancing a game is that on one hand, it should never be cheaper to build a better army, all else being equal than the nearest reasonable competitor. That presents the problem of how one defines “cheaper”, either as a percent of GDP, on a nominal basis, or purchasing parity and so on, and how much you want to factor in technology. In SR:CW the choice was made to stick to historical accuracy as close as possible for unit statistics. Some of those statistics, such as weapon range, travel range, mass, and so on are easily found, and the rest are chosen to reflect their expected capabilities given the combat model. This is not intended to be balanced; the feeling being that multiplayer is a relatively small segment of their customer base, and given the relatively limited set of options for balanced country play, it seemed impractical. Multiplayer gamers have proven inventive in trying to, with the historical model, come up with fair scenarios or self-enforced rules people can play[8].

[7] http://us.battle.net/sc2/en/forum/ as of June 2011

LEGAL AND SOCIAL ISSUES

SR:CW has nuclear weapons to kill millions, which are, bizarrely, not all that controversial in the ESRB and PEGI ratings applied to the game, whereas a depiction of direct person on person tactical violence warrant an older suitability. Supreme Ruler 2020 and SR:CW have a PEGI 7+ and ESRB E 10 for everyone rating, as does Hearts of Iron, but the Total War series rate T for teen with ratings variously for Blood and Gore, Violence, alcohol, and sexual references. However, there are still controversial choices, notably on country borders and who is defined as a colony or satellite state. This is a relatively complex topic.

Where is the border between India, Pakistan and China in Kashmir? Territory that is clearly disputed, but also clearly under the control of one party, The Falklands for example, are relatively easy – the controller is the owner. On the other hand, the more murky areas, especially in the colonial era pose other problems. Who is in charge of Rhodesia in 1949? Should it be independent in some way, a satellite state, or something else? The British reorganized the territory several times after 1949, and there is no particular reason that the current arrangement would have been the final one if different choices were made (by, in this case, the player). Algeria was part of France, to them an integral part, for several years after the start of the game, and a different evolution of history could have seen Algerian Independence play out very differently. Tibet is always a great source of animosity; is it a satellite, independent, part of or something else with Communist China (who are themselves an interesting case). There are a number of conflicts that were largely internally driven, by the people so to speak, rather than by the state. Assuming one would even want to, how do you model an apartheid state? What about Turkish and Greek populations in Cyprus? The list goes on and on. Often it is not clear what is the most accurate portrayal should be for a game, and when one factors in the sensitivity of these matters, these are formidable questions indeed. The best approach is often to select a model that is consistent with the accepted reality in the largest markets, or, if a developer is big enough, making different versions for each locale.

[8] These are not necessarily actual scenarios, but simply agreed rules or victory conditions they play to. For example, the first player as a European state to conquer Australia wins.

Trademark and copyright is another great legal question in historical games (Rosenthal 2009). Do car simulators need rights to use specific vehicles? Gran Turismo advertises licenced cars from top manufacturers. But then who owns the rights to the image of an F22 or the like? What about something historical, like a ship from 1940? How about a ship that is now a museum? In SR:CW the units are all modelled by the team, and many are fictionalized similar but not identical for gameplay reasons, so as to not have, for example, 15 different versions of what is basically the same aircraft. Typically, government equipment falls in the public domain, but one has to be careful on what is being depicted and when. In a strategy game, one must be careful about depicting military organizational units (such as regiments, platoons and divisions) as well, rather than just equipment. Basing a game on the activities of the Irish 1 Southern Brigade, and use of their flags and so on may run into conflict with the official use of those images.

Depicting real people poses a considerable number of problems. Homefront (Kaos Studios 2010) encountered legal problems in Japan and the Republic of Korea (South Korea) over portrayal of real people (Owen 2011). The game was outright banned in South Korea, so as to not antagonize the North, and changed in Japan to not be malicious to real people or countries (Parker 2011). Depictions of various fascist leaders risk getting a game banned, especially in Germany, and even if they are portrayed in a negative light (Stegbauer 2007). The political system in SR:CW is, for the most part, sufficiently abstract to avoid this, and starting in 1949, most of the world has changed. However there are still important people in the world from that period, and as the game gets deeper in, the player could find more and more people who are, in turn, relevant today if real people were portrayed.

Homefront also had to change the enemy they were trying to portray, which is obviously Communist China. Having the PRC as the enemy would get the game banned in the PRC, cause political tensions, and so on. For a game like Hearts of Iron, or Supreme Ruler, being banned by the People's Republic of China may actually benefit sales, due to the free press coverage, and the lack of existing sales in China anyways. For a product like Homefront however, with a major publisher like THQ they obviously chose a different route.

SANDBOX VS. HISTORICAL SIMULATOR

Where other Paradox-published games tend more towards historical simulators, with heavily scripted events to lead the world towards something like what actually happened, BattleGoat Studios has taken the route of setting up the world in something reasonably historical and letting the player have at it, to radically transform the way history unfolds. The goal here is to empower the player to make interesting choices if they want. That is prohibitively hard if they choose to play the Comoros, as they can choose to do, but a player as Indonesia, Nigeria, or Vietnam should have a lot of choice on how the world plays out, even if they were, in October 8, 1949 not entirely free. At the same time, the world should behave somewhat authentically, and trying to build a nuclear aircraft carrier as an independent Vietnam in November 1949 would seem somewhat unrealistic.

The aim for us was to build a world that as realistically as possible depicts the start date. Where a historical simulation diverges from a sandbox is in how they make the system evolve. Forcing France, and then the United States into the decolonization and war scenario in Vietnam would guarantee history evolves with major defining events for both parties. Doing so, however, may not have any connection to how the player is evolving the game. A historical simulator attempts to present the player with authentic historical choices, and then believable historical consequences. A sandbox aims to present the player, at least initially, with authentic historical choices, but then shape how the world forms through their choices, and have random, unexpected things happen. Imagine a World War 2 game where the Nazis never invaded Poland. Suddenly, it ceases to be much of a World War 2 as we know it. SR:CW lets the cold war go hot, the various smouldering insurgencies and proxy wars move and pop up in different places. The different styles require different tools. A sandbox needs a more general artificial intelligence, whereas a simulator needs more robust scripting, and artificial intelligence that behaves more historically.

PRE-LAUNCH THOUGHTS ON SR:CW

As with any game design, an enormous number of decisions are made before launch; some big, some small. One never knows how the customer base is going to respond to this work until it has been released and in their hands. In this section, we briefly summarize some of the choices made (as discussed in previous sections), for easy comparison to the post-launch reception.

On the cosmetic side, BattleGoat Studios has tried to be as authentic as possible with SR:CW. With events to change flags to historical norms, with names of places all as close to their actual names as possible. The overall gameplay model of building, production, and combat is very much the same as previous versions, so the presumption is that the target customer base will be satisfied with what works and what they know.

Where things are likely to get interesting is in colonies, satellite states and the new geopolitical models of diplomacy, and proxy wars and so on. How the player base will respond to some of these new features remains to be seen. No one launches a game hoping for a negative reception, and that is not likely, but it was interesting to see how the players respond and what they want tweaked for the major patches.

Multiplayer was another unknown. The Supreme Ruler series has supported multiplayer for some time, and it is a relatively small part of the audience, which makes it hard to know what exactly they want. There is not a major attempt to make a balanced multiplayer game; while a balanced multiplayer scenario probably could be made, the expectation is that players will prefer the historical route, or will add whatever they want through various mods to the game.

As of time of submission for review, a hot topic on the BattleGoat forums was the space race, and how technology will be tied into that[9]. The game certainly has technologies tied to the space programmes, and there is a fairly abstract model of orbiting satellites, and a whole space race victory condition, fortunately we had anticipated this one, though players would have liked a less abstract space race.

POST LAUNCH ANALYSIS

The main mechanism for feedback on SR:CW is on the BattleGoat and Paradox forums, with other feedback arriving through forums for various retailers, game websites, and other portals as well. Many of the issues that came up during launch have already been touched on elsewhere in this paper, such as the map, the economy, and some unit issues. Several areas deserve separate attention, however.

On the cosmetic side, Churchill and de Gaulle were not actually the leaders of the United Kingdom and France in 1949. The fans really noticed this issue, and pointed out a couple of flag errors we had as well. As mentioned earlier, the satellite maps missed a few geographic features that several users think we should have added.

Strategically, the diplomatic artificial intelligence seems to struggle with expanding spheres properly, especially as the United States, and it is not clear as a player how to do it. The United States should come into the time period friendly with a lot of people, but for gameplay reasons that momentum is not reflected; it would make the United States too powerful. Unfortunately, it creates a very poor Cold War when all of NATO joins the USSR and the United States does nothing about it.

Technology can be very unbalanced. Some units, especially 1960's and early 70's era do not really require any technology that did not exist in the early 50's. Instead, it just took a long time for people to bother developing them. This gap, where you can jump ahead in technology and militarily is easily exploitable.

The world is round, the map in SR:CW is not, and the Pacific is bigger than portrayed. For much of the Cold War, the threat of missiles being lobbed over the pole with bombers in hot pursuit just does not fit right on a flat map. SR:CW created a strategic deployment option where a unit can be sent anywhere without seeing how it gets there (which is itself a system the player needs to learn). This has confused several people, and required some tweaking to unit ranges from historical accuracy. The Pacific had to be shrunk down a little for some path finding reasons. It is still big, although it looks a bit odd.

The SR:CW model of the United Nations is a fairly abstract notion that governs world trade, which is not really what the United Nation does. Being unpopular with the United Nations in SR:CW makes it impossible to buy goods, yet there are several countries in the game that were not even part of the United Nations for decades (notably the People's Republic of China). In a game about waging war, it is a challenge to find a good role for an organization devoted to peace.

The United States has over 60 aircraft carriers in 1949, which is reasonably historical considering many are escort carriers. Unfortunately, this presents the player with a dizzying managerial task at the start of the game, and the artificial intelligence, which does not know how to disband units, can end up with an absurdly large navy of largely antiquated ships.

SUMMARY

As demonstrated in this paper, historical accuracy can be an interesting and important topic of discussion. Developers face many issues and challenges in this regard in the creation of their games, with far-reaching ramifications on the sale and reception of their games. This paper has highlighted many of these issues and challenges, using Supreme Ruler: Cold War as a source and reference for discussion.

[9] http://www.bgforums.com/forums/

One area we have not discussed is tactics and doctrines. Tactics and doctrine tie heavily into the artificial intelligence system, and are a separate topic. Larger strategies, like strategic bombing (and its various euphemisms such as area bombing, or precision bombing) may have various abstract implementations, but more specific strategies like infiltration are somewhat different. Here, a believable model should try and deal with a historical strategy that is not successful, and try to change. The reactive nature of strategies and how they work with the artificial intelligence system is a large separate topic that requires further attention and study.

REFERENCES

Games

BattleGoat Studios 2005 Supreme Ruler 2010 Published by Strategy First.

BattleGoat Studios 2008 Supreme Ruler 2020 Published by Paradox Interactive

BattleGoat Studios 2011 Supreme Ruler: Cold War Published by Paradox Interactive.

Blizzard Entertainment 2010 Starcraft II: Wings of Liberty

Creative Assembly 2011 Total War: Shogun 2 Published by Sega

Firaxis Games 2010 Civilization 5. Take 2 Interactive

Kaos Studios 2010 Homefront Published by THQ. http://www.spike.co.jp/homefront/ - specifically the Japanese version

Paradox Interactive 2003 Victoria: An Empire Under the Sun, Strategy First.

Paradox Interactive 2007 Europa Universalis III Self published

Paradox Interactive 2009 Hearts of Iron III

Paradox Interactive 2010 Victoria II

Polyphony Digital 2010 Gran Turismo 5 Published Sony Computer Entertainment

Not specifically listed: expansion packs or other titles in the series of the above games, major revisions due to patches or others in the same series, ports to different platforms.

Papers or Websites

Bismuth, C., Borron, S., Baud, F., and Barriot, P. 2004. "Chemical Weapons: Documented Use and Compounds on the Horizon". Appeared in Toxicology Letters, Volume 149, Issues 1-3, April 2004.

Charette, R. 1996. "Large-scale Project Management is Risk Management". Appeared in IEEE Software, Volume 13, Number 4, July 1996.

Moreno-Ger, P., Burgos, D., Martínez-Ortiz, I., Sierra, J., and Fernández-Manjón, B. 2008. "Educational Game Design for Online Education". Appeared in the Journal of Computers in Human Behavior, Volume 24, Issue 6, September 2008.

Stegbauer, A. 2007. "The Ban of Right-Wing Extremist Symbols According to Section 86a of the German Criminal Code". Appeared in the German Law Journal, Volume 8, Number 2.

Websites

Carpenter, A. 2003. "Applying Risk Analysis To Play-Balance RPGs". Accessed June 2011 from http://www.gamasutra.com/view/feature/2843/applying_risk_analysis_to_.php.

Central Intelligence Agency. 2011. "The World Factbook". Accessed June 2011 from https://www.cia.gov/library/publications/the-world-factbook.

China Daily. 2004. "Swedish Video Game Banned for Harming China's Sovereignty". Accessed June 2011 from http://www.chinadaily.com.cn/english/doc/2004-05/29/content_334845.htm.

North Atlantic Treaty Organization. 1949. "The North Atlantic Treaty". Accessed June 2011 from http://www.nato.int/cps/en/natolive/official_texts_17120.htm.

Outpost Gamez. 2011. "Supreme Ruler: Cold War Q&A". Accessed June 2011 from http://www.outpostgamez.com/outpostgamez/top-stories/668-supreme-ruler-cold-war-qaa.html.

Owen, P. 2011. "Japanese Version of Homefront Loses References to North Korea". Accessed June 2011 from http://www.gamefront.com/japanese-version-of-homefront-loses-references-to-north-korea.

Parker, L. 2011. "Kim Jong-il Axed from Homefront in Japan". Accessed June 2011 from http://www.gamespot.com/xbox360/action/homefront/news/6297959/kim-jong-il-axed-from-homefront-in-japan.

Rosenthal, D. 2009. "Legal Issues for Game Developers". Accessed June 2011 from http://gamedesignconcepts.pbworks.com/w/page/11829076/Legal-Issues-for-Game-Developers

Special thanks are extended to the whole team at BattleGoat Studios for collaborating with us BattleGoat Studios would also like to thank fans of the Supreme Ruler series for their continued feedback and support over the years.

Biography: Brian Srivastava is a PhD student in game development at the University of Western Ontario under the supervision of Dr. Michael Katchabaw.

Dr. Michael Katchabaw is an Associate Professor in the Department of Computer Science at the University of Western Ontario.

George Geczy is the Co-Founder and Lead Programmer at BattleGoat Studios. His first game, "Exterminate", was published in 1981, soon followed by the original text-based version of Supreme Ruler on the Radio Shack TRS-80 microcomputer.

SERIOUS GAMING AND TRAINING

Non Verbal Communication Assisted Serious Gaming Applications

Alan Murphy and Sam Redfern
Department of Information Technology
National University of Ireland Galway
a.murphy30@nuigalway.ie, sam.redfern@nuigalway.ie

KEYWORDS
Serious Games, Virtual Learning Environments, Nonverbal Communication, Collaborative Decision Making.

ABSTRACT

The aim of this paper is to provide a brief review of research in the field of nonverbal communication as applied to 'serious gaming' applications. Serious games are designed with the primary purpose of applying simulation technologies to non-entertainment or 'serious' applications as opposed to solely entertainment applications. Although such applications may entertain, their primary purpose is to teach, train or investigate. One fine example is Microsoft Flight Simulator which has been used as the first step in training real pilots. Organizations are constantly intensifying their efforts to engage with their workforces and serious games offer a powerful approach to the training and skills development of a workforce.

An overview of virtual environments shall be outlined, along with a study of the natural processes of communication that exist between individuals in a face to face environment. Examining such processes leaves us with a greater understanding of the nonverbal aspects of group dynamics that should be supported in serious gaming applications to provide for more enriched interactions. Through such richer interactions group members can be better equipped to collaborate, train and learn in these 'serious' simulations.

INTRODUCTION

Serious games are a product of the convergence of game based methods and concepts with other ICT technologies and research areas. They can be applied to a broad spectrum of application domains ranging from training and simulation in industries such as education, military simulations, health care, emergency response management and planning, city planning, engineering and many other societal relevant topics and business areas (Sawyer and Smith 2008). In addressing pertinent literature, this review will aim to highlight relevant research regarding the virtual worlds and interaction environments in which these serious games are built. Also addressed in this review are the communication processes which should be supported in these applications, in order to provide an efficient communication framework which will foster the improvement of the learning process. This review will focus on group based serious simulations where positive group dynamics and intuitive nonverbal interaction are beneficial for effective negotiations and decision making.

The virtual worlds in which these serious games are constructed have become a critical aspect of social computing in the past few years, due to the latest technological advancements in user interfaces, graphics and networking technologies. Access to such environments has had critical implications for business, education, training applications, medical treatments and for society in general (Messingera et al. 2009). In such three dimensional spaces, members can engage in rich interactions with other inhabitants simultaneously through graphical representations of themselves (i.e. avatars) (Messingera et al. 2009). Many organizations and individuals have realised the potential uses and cost savings this alternative reality could incur, by enabling them to simply "step into the internet" to interact with their peers (Jarvenpaa 2007). The quality of these interactions is strongly correlated with the quality of the structures and software agents put in place to replicate the quality of real life interactions.

In discussing the use of serious gaming applications for task training in businesses where teamwork, collaboration and decision making are central, intuitive communication is paramount. Ideally, the perfect collaborative environment would cater for and support all channels and means of communication that are at the disposal of two individuals having a conversation face to face. Unfortunately this is implausible due to the technical nature of such systems and so most only support voice and text based communication. Other channels like gesture, expression and head orientation should be controlled subconsciously in order to reflect a real life communication scenario. Slater et al. (2000) outlines a number of explicit user interfaces that provide conscious control of non verbal cues. Unfortunately, such interfaces fail to reflect real life interactions where gestures are a subconsciously controlled.

VIRTUAL TRAINING ENVIRONMENTS

A virtual training environment is a system which is designed to support the training of a group, faced with a collaborative training task, or of individuals faced with training objectives in many virtual settings. It is important to provide an understanding of the collaborative training environments which support such virtual learning applications. They are essentially distributed virtual reality systems which support multi user access. The following definition describes these Collaborative Virtual Environments (CVEs) or Virtual Learning Environments (VLEs) as "…computer-based, distributed, virtual spaces or set of places. In these places, people can meet and interact with others, with agents or with virtual objects" (Churchill et al. 2001, pg. 4). These environments can also vary in representational richness, from 3D graphical spaces, to 2D environments and to text-based environments. These systems have become increasingly popular for their usefulness in collaborative serious gaming applications in which the sharing of information is central. One such serious game is 3DiTeams' HumanSim (Virtual Heroes, 2007) which is used for medical education and the training groups. Fig 1 below provides a screenshot of the HumanSim application.

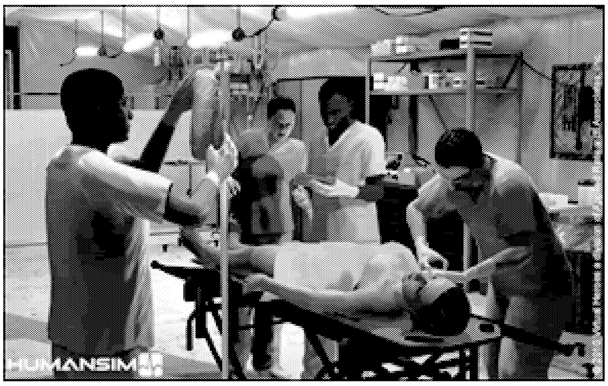

Fig 1: The training of medical procedures and effective group work in HumanSim (Virtual Heroes, 2007).

Differentiating Space and Place

In order to create a suitable environment for human learning it is important to distinguish between space and place. Churchill et al. describes the space as a "physical (or digital) volume" that can house objects and where events can take place. (Churchill et al. 2001, pg. 6) The place, however, is determined by how that space is used to create the environment in which inhabitants will collaborate and learn e.g. the space may be a sports hall but it may also be a place for special functions or other meetings. So a space essentially becomes a place when it is appropriated and given social meaning. However, it is important not to simply endeavor to slavishly replicate a place or to replicate reality. Such resource heavy replication might not particularly improve the interactive experience, but restrict the use of resources that can. Real life collaborative training of groups involves the complex exchange of information between collaborators and therefore according to Churchill et al. (2001, pg. 9) the following key features should be the ones to be optimized to allow for meaningful interactions: shared context, awareness of others, negation & communication and finally flexible viewpoints.

Awareness and Interactions

In multi user serious games, awareness not only refers to the intentional awareness of the activities of participants around you but also of peripheral activities exclusive of the current task context. Such awareness, in turn paves the way for rich interactions and negotiations between fellow trainees. In order for a serious game to be successful, intuitive negotiation with fellow trainees of task related activities and also task structure in terms of responsibilities is essential (Churchill et al. 2001, pg. 10). In order to support this natural negotiation, tools must be put in place to support natural communication within the gaming application. To understand what these systems need to achieve, a deeper look at the natural process of communication is necessary.

COMMUNICATION IN TRAINING APPLICATIONS

Essentially communication can be described as the process of information exchange. Foulger (2004) describes communication as "the process by which people…construct representations of meaning such that other people can interpret those representations". The reader can extract two important elements of communication from the above definition, namely meaning and interpretation. Communication can also be described as the process by which individuals interpret intended meaning, so without a meaning to interpret, and without the accurate interpretation of that meaning, the communication lacks integrity. In order for an individual to share meaning with another, there is always a certain gap to be bridged. When referring to serious games used for training groups, it is safe to assume that it is the communication structures that are put in place that will provide the bridges that will comprise the communication process between trainees (Foulger 2004).

There are two main categories of communication between individuals; these are verbal communication and non verbal communication. Ross Buck and C. Arthur Van Lear (2002) categorise verbal communication as the "intentional" use of language, be that signed, written or spoken and non verbal communication as the "non intentional" communication or underlying emotional states or meaning, through the processing of certain signals given "spontaneously" by the communicator. Social psychologists have argued that more than 65% of the communicational information exchanged during a face to face encounter is carried on the nonverbal band (Fabri et al. 1999). Such a measure is difficult to calibrate and this can also depend on the context of communication which has resulted in varying percentages of the verbal/nonverbal ratio. However, it is given that there is a need to provide support for such channels of communication when designing a platform for remote group training of individuals.

Nonverbal Communication in Collaboration

Extensive studies have demonstrated and proven the extent to which facial expression, head nods, hand gestures, posture, eye gaze and eyebrow raises can determine how most utterances are interpreted or add underlying meaning to the obvious instrument of speech. It is these forms of bodily cues that can provide a background context for verbal negotiations and interactions in training applications (Beebe 1979). Training games are commonplace in many industries today. Military officers have been playing war games for a long time now, to learn tactical and strategic skills, and to learn from their failings in these mental contests. Training simulations and group learning has also been used by businesses around the world to train specific skills before practice. It is important to outline the role the non verbal communication plays in such a business or training context. Recent developments by Stefan Marks (2009) highlight the necessity to expand the support for non verbal channels of communication in serious games which is "still limited". His work outlines the role that NVC can play in a surgical setting in serious games such as HumanSim and how gesture support can create a context for addressing people and objects (Fig 2). These concepts could potentially be applied to games such as America's Army (Virtual Heroes, 2007) which aims to train military personnel teamwork and decision making skills.

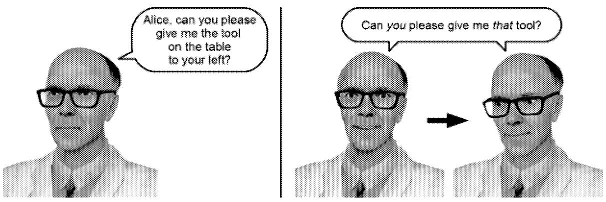

Fig 2: The non verbal addressing of an object (Marks, 2009).

Whatever the task aim, or training goal, intuitive interaction with other participant(s) is crucial in establishing a shared and common understanding of the task or aim at hand. Beebe (1979) argues that this "meta-communication" or communication about communication can serve to repeat, contradict, substitute, complement, accent or regulate verbal communication. Should industry officials or training participants wish to ascribe truthful meaning from their trainee counterparts; much care should be taken in also interpreting the non verbal messages in their proper context.

PROBLEM STATEMENT & CONCLUSION

One of the most important features of training simulations is the presence of goals and objectives. In order to reach these goals and objectives, decision making is invariably a crucial part of the process. Caird-Daley et al. (2007) explain that many decision making problems in a naturalistic environment involve more than a single decision maker. Decision making teams, training cooperatively in a serious game, can comprise a number of individuals co-located or geographically dispersed. This can present challenges in terms of ensuring a sense of shared awareness and understanding of task specific information. Further research in psychology and neuroscience as conducted Zhang & Li (2006) has identified the critical role of emotion in decision-making and social interaction. Such games should aim to support conveyable emotion through their avatars, thereby aiding the decision making process.

In conveying emotion and nonverbal communication, Churchill et al. (2001, pg. 10), note that the gestures which would so greatly compliment verbal communication, are "often hard to achieve with embodiments where nuanced subtle gesturing is not easily supported". To date many approaches taken to capture user data to recognize hand gestures have involved mountable sensors which are used for mapping a hand movement of to that of an avatar. (Fabri et al. 1999) In terms of capturing a participant's emotional state or facial expression, which would be a great asset to interpreting accurate meaning, much research has been based on capturing fundamental human emotions as described in the works of emotional psychologists Robert Plutchik (1962) and Paul Ekman (1980). Most prototypic development of emotion capture solutions as applied to serious games to date, have solely been trained to capture a static and finite list of basic emotions, lacking the dynamic capacity of capturing other more subtle, unread emotions.

In reality there are many more human emotions that can only be recognized through a constant monitoring of a participants' expression. Future research needs to address a more dynamic solution of monitoring all expressions, instead of searching for a trained few. The non-verbal channel of communication could be utilized to a greater potential, thus improving communication between participants of a serious game and carrying across a more natural process of communication, similar to that which exists between individuals in a face to face scenario. This enhanced nonverbal support, should in turn aid the decision making processes which are so central to most modern serious gaming applications.

FUTURE WORK

Given the preliminary nature of this research, there is much experimentation to be done. Quantifiable results are required in order to determine the usefulness of such dynamic emotion capture and its usefulness in group decision making. The design of such experiments should rely on strong reference from psychology literature in order to combine multi disciplinary research into a possible serious gaming contribution or application. Such an experiment may involve inducing emotion in a subject by undertaking a certain activity, while being concurrently assessed by a psychological standard for reading the emotion induced.

BIBLIOGRAPHY

Beebe, S. A., 1979, *"Non Verbal Communication in Business: Principles and Applications",* Annual Meeting of the Midwest Business Administration Association Chicago, Illinois, 1-27

Buck, R. and Van Lear, A., 2002, "*Verbal and Nonverbal Communication: Distinguishing Symbolic, Spontaneous, & Pseudo-Spontaneous Nonverbal Behavior*", Journal of Communication, Blackwell Publishing, 52(3), 522-541

Caird-Daley, Antoinette & Harris Dr. Don, 2007, "*Training Decision Making Using Serious Game*", Human Factors Integration Defense Technology Centre, 1-35

Churchill, E. F., Snowdon, D., Munro, A. J. (Eds), 2001, "*Collaborative Virtual Environments: Digital Places and Spaces for Interaction*", London, Springer Verlag

Ekman, P. 1980, "*The Face of Man: Expressions of Universal Emotions in a new Guinea Village*", STPM Press, New York

Fabri, M., Moore, D.J., Hobbs, D.J., Smith, A.B., 1999, "*The Emotional Avatar: Non-verbal Communication between Inhabitants of CVEs*", In Proceedings of the International Gesture Workshop on Gesture-Based Communication in Human-Computer Interaction (GW '99), Springer-Verlag, London, 269-273

Foulger, Davis. 2004, "*An Ecological Model of the Communication Process*", Brooklyn College/CUNY, Retrieved: 15 April, 2011, http://davis.foulger.info/papers/ecologicalModelOfCommunication.htm

Jarvenpaa, S., Leidner, D., Teigland, R. and Wasko, M., 2007, "*New Ventures in Virtual World*", MIS Quarterly, 1-3

Messingera, Paul R., Strouliab, Eleni., Lyonsc, K., Bonea, M., Niud, Run H., Smirnova, K. and Perelgute S., 2009, "*Virtual worlds - past, present, and future: New directions in social computing*", Decision Support Systems, 47(3), 204-228

Plutchik, R., 1962, "*The Emotions: Facts, Theories, and a New Model*", American Journal of Psychology, New York: Random House, 77(3), 518-522

Sawyer, B. & Smith, P., 2008, "*Serious Games Taxonomy*", Game Developers Conference, Florida, 1-54

Slater, M., Howell, J., Steed, A., Pertaub, D.-P., and Gaurau, M., 2000, "*Acting in Virtual Reality*", In Proceedings of the Third International Conference on Collaborative Virtual Environments, 103–110

Marks, S., Wünsche, B., Windsor, J., 2009, "*Enhancing Virtual Environment-based Teamwork Training with Non-verbal Communication*", In GRAPP, 361-366

Virtual Heroes, "*Healthcare*", Retrieved: 1st August, 2011, http://www.virtualheroes.com/healthcare.asp

Zhang, Hua & Li, Rui, 2006, "*Emotional Nonverbal Communication Based on Fuzzy Dynamic Bayesian Networks*", ICARCV 9th International Conference, 1-5

BIOGRAPHIES

ALAN MURPHY attended the Institute of Technology in Athlone, Ireland, where he studied for a B.Sc. in Computer and Software Engineering (2008). He followed this by studying in the National University of Ireland, Galway where he obtained a M.A in Digital Media (2009). He then worked for one year at web application and database development for a Galway based Production Company (2010). Alan has since returned to full time education in NUIG where he is currently studying for his PhD in Information Technology.

SAM REDFERN attended the National University of Ireland, Galway, where he studied for a B.A in English and Archaeology (1992), followed by a M.Sc. (1994) and Ph.D. (1998) in Information Technology. He has worked as a lecturer in Galway since 1996, and has since published in the areas of digital image processing, various types of A.I., graphics, collaborative virtual environments and serious games. He has been an independent game developer in his spare time since 1984, with games published on the BBC Micro, Amiga, PC, Mac, iPhone and Android.

Experimental Assessment of an Emotion Tracking Software Agent (ETA) for assisting Communicative Interactions of Multitasking Users in Groupware

Paul Smith and Sam Redfern
Discipline of Information Technology
National University of Ireland, Galway
Galway,
Ireland.
E-mail: dr.paul.t.smith@gmail.com
E-mail: sam.redfern@nuigalway.ie

KEYWORDS
CVEs, Automatic Facial Expression Recognition, Serious Games, Virtual Environments, Non-Verbal Communication, CSCW, Neural Networks, Groupware

ABSTRACT

Interactions such as discussion and negotiation in face-to-face work contexts strongly rely on non-verbal feedback. Such feedback provides indications of important negotiation states such as agreement or disagreement and understanding or confusion. The increasing popularity of groupware and its use by virtual teams for collaborative remote work necessitates the development of appropriate tools to manage the reality of distributed and remote work. Such remote collaboration is often hampered by a lack of social cohesion and such phenomena as participant multi-tasking. This paper outlines the experimental assessment of a proof of concept AI based software agent (Emotion Tracking Agent or ETA) for the real-time tracking of groupware user's facial expressions of emotion during virtual meetings. The software agent is designed as a novel approach to the removal of negative or unwanted effects of user multitasking and attention distracting behaviours in virtual collaboration and meeting environments.

INTRODUCTION

Groupware for remote collaboration has become increasingly important in recent years due to the growing number of businesses using distributed workers based in different locations around the world (Lojeski, Reilly and Dominick, 2007). Studies have established that companies such as Intel (70%), IBM (40%) and Sun Micro Systems (nearly 50%) already have high percentages of virtual and distributed workers (Lojeski, Reilly and Dominick, 2007). These workers can be based at home or abroad or may travel frequently making remote collaboration software the most convenient and economical means for these individuals to communicate with colleagues and clients.

In virtual and face-to-face meetings, one of the inherent problems described in the literature relates to the negative impact of multitasking in both traditional and virtual work environments (Appelbaum, Marchionni and Fernandez, 2008). Successful management of worker multitasking and the provision of appropriate solutions to specific problems caused by multitasking or inattention is of key importance to future work practices (Black and Hearne, 2008).

This paper briefly discusses the architecture and use of our prototype Emotion Tracking Software Agent (ETA) in tackling the negative impact of multitasking or task switching behaviours of users in virtual collaboration environments. The ETA is also used as a means to augment a user's ability to communicate his/her non-verbal displays of emotional information to fellow collaborators, enriching their communication in environments which traditionally tend to provide poor support for non-verbal communication (Ekman and Friesen, 1975).

The research area of serious games is a diverse field, encapsulating any usage of game technology for purposes other than its primary entertainment use. Our developed ETA is designed for integration within many forms of virtual interaction environments, many of which are based on existing game technology and graphic engines. The environment which our ETA is integrated for testing is itself created with the Torque Game Engine developed by Garage Games. Further discussion of our agent and its usage in serious games application is described in (Smith and Redfern, 2010).

In the later part of this paper we describe a number of tests designed to assess the ETA's usability and accuracy in recognising user's facial expressions. An implementation study is then described where a number of subjects were tested in a communicative interaction in a Collaborative Virtual Environment (VRCGroups) in which the ETA was integrated, in order to provide feedback on the effect of inclusion of the agent in a groupware environment.

ETA ARCHITECTURE

The software agent was developed on a Windows operating system and the initial code for the neural network used in the project was taken from (Chopra, 2009) and is open source, written in C++ which has been extended and edited for our purposes. The completed neural network consists of three separate layers: the input layer which receives numerical inouts, a single hidden layer and a final output layer in which a single emotion is identified as the dominant among Paul Ekman's universal prototypic emotions (Ekman, 1980) (see Figure 1). As illustrated in Figure 1 there are 15 nodes in the input layer representing the angles computed using an input representation scheme, 15 nodes in the hidden layer and 6 nodes in the output layer (one for each of the prototypic emotions proposed by Ekman). The input representation scheme was chosen after a series of experiments were carried out using the marker coordinates recorded from the user's face and an optimal scheme was determined through the use of cluster analysis which took the positions of the markers and computed 15 specific angles between them (see Figure 2).

The finished neural network was trained using 40 instances of each emotion (270 in all) and 10 instances were recorded separately for each emotion and used as validation sets for early stopping and prevention of over-fitting.

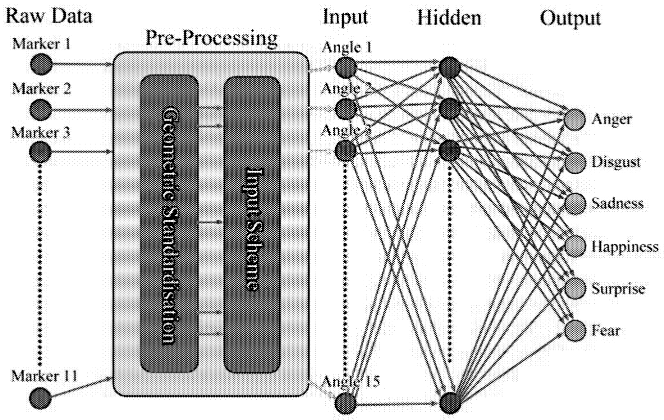

Figure. 1. The basic structure of the ETA neural network, showing how raw marker coordinate data is processed through the network.

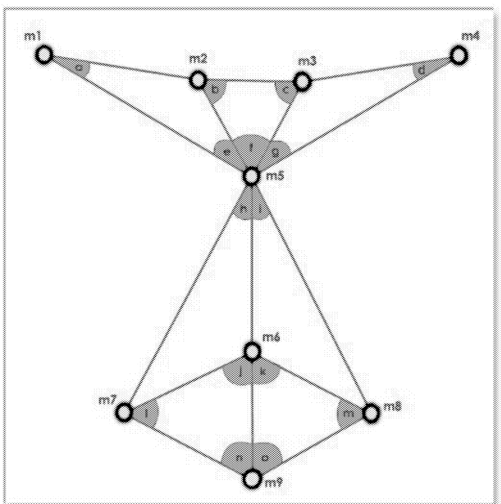

Figure. 2. Illustration of 15 angles used in current analysis for the Angle Input Representation Scheme. Angles are labelled in order from a to o.

The data which is sent to our emotion recognition neural network for identification is recorded in real-time using an Optitrack FLEX:C120 optical motion capture camera (see Figure 3). This camera is an integrated image capture and processing unit which uses a B&W CMOS imager to captures 120 frames of video per second and an onboard image processer which transfers marker data over standard USB to a computer for display and post processing. The camera then preprocesses the record image frame, removing light from none refelective surface which allows only the reflective markers to be detected in the correct lightening conditions.

Eleven tracking markers are placed on the user's face at specific position to facilitate feature tracking and the coordinates of these markers are computed by the camera and sent to the ETA for categorization. Figure 4 shows the markers placed on a user's face in their correct positions (Zhang, 1999). More information on the neural network behind our ETA is available in (Smith, and Redfern, 2010a)

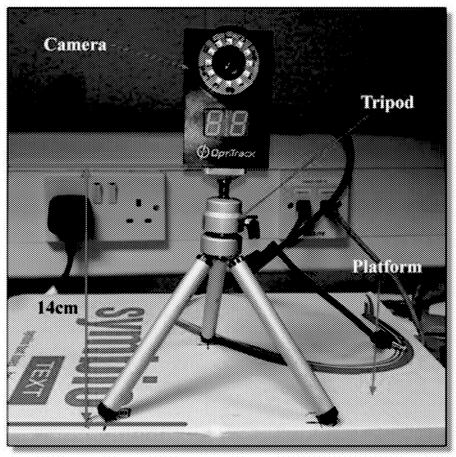

Figure. 3. Our chosen tracking solution, the Optitrack FLEX: C120 developed by Naturalpoint mounted on a standard tripod.

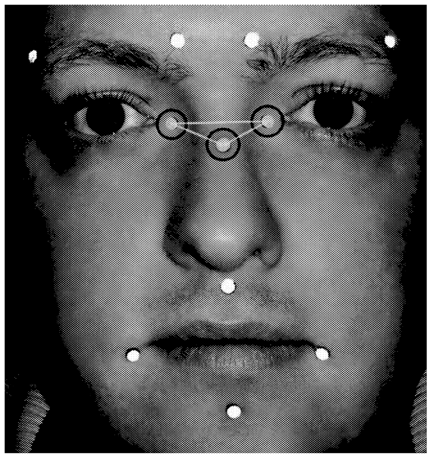

Figure. 4. Tracking markers placed in specific positions on user's face to facilitate feature extraction.

The ETA is developed in C++, and incorporates a Multi-Layer Perceptron (MLP), trained using back propagation (Stergiou and Signanos, 1996). Data is received from the Optitrack camera via the SDK. The MLP with backpropagation was chosen due to its prevalence in the facial tracking literature, and due to the flexibility it

afforded us in allowing users to record their own specific emotions and use them as training for the network (Pantic and Rothkrantz, 1999).
For the sake of experimental assessment of the prototype ETA, we integrated the agent within a Collaborative Virtual Environment (CVE) known as VRCGroups (developed previously by our research group) (Redfern et al, 2006). Inside this CVE, the agent tracks the user's facial expressions at all times and prompts participants of the emotional changes of fellow collaborators inhabiting the same environment.

ETA INFORMATION MODES

Since the aim of the software agent is to increase users' emotional awareness and attention in remote meeting or groupware environments, in VRCGroups when ETA functionality is enabled, the users are prompted with information relating to other participant's emotional states in three separate communication modes. These modes are vocal (a recorded message prompts the user of fellow collaborator's emotional states), textual (a chat message is sent to the user informing them of the emotion change of a fellow environment user) and visual (an emoticon icon appears over the head of the user whose emotional state has changed allowing easy identification of the user's avatar). The user also has the ability to specify which co-workers they are interested in receiving information about, from the ETA to ensure that they are not bombarded with unwanted information which may detract from the interaction.

EXPERIMENTAL ASSESSMENT OF ETA

The neural network which forms the heart of the emotion recognition system for our ETA was evaluated in a number of experimental tests designed to gauge its ability to recognise each of the 6 primary emotions. The network's abilities were also compared to the emotion recognition capacity of a group of human participants tested on videos of the same facial expressions as the network, in order to ascertain that it operated with at least the abilities of a human observer. This also ensured its performance was adequate for its real-time usage requirement, as a substitute for a user's eyes, in keeping track of the non-verbal communication of other users outside his/her field of view. The final testing of the software agent took the form of an implementation study where the ETA was integrated in the VRCGroups environment in order to evaluate its contribution to reducing the negative effects of multitasking among users in such groupware.

ETA Accuracy Experiments

In order to determine if the ETA was as (or more) effective at classifying emotion from facial movements than human subjects, two tests were carried out. Firstly, we sought to discover how well a human subject could correctly interpret an emotion from the facial expression of another human being recorded displaying a number of expressions on video. This test entailed the simultaneous recording of a specific user's facial movements using the FLEX:C120 camera and a digital video camera. The experiment sought to evaluate the recognition ability of the ETA neural network in comparison with human subjects. The videos recorded using the digital camera were displayed to a group of 15 subjects instructed to identify which of the 6 primary emotions (the 6 prototypic emotions proposed by Ekman were Anger, Disgust, Sadness, Happiness, Fear and Surprise) each video portrayed. The data recorded by the motion tracking camera (2D coordinates of facial markers) was sent as input to the ETA neural network in the same order as the videos were given to the experiment subjects. The results were then analysised and compared (accuracy was computed using percentage of correctly identified emotions from data and video).

Analysis of the results for the first stage of this test, which involved human subjects identifying emotions depicted in videos indicated that a high percentage of the test participants had a strong ability for emotional recognition. A 78% recognition rate was recorded suring this test which was in line with previous tests carried out in the literature such as the high average recognition accuracy of 89% published in (Susskind et al, 2007) and the relatively low accuracy of 69% published in (Zucker et al, 2008). The results appeared to be in the midpoint between both these previous studies which is verified further by a Chi Squared analysis of the results.

Analysis of the results of the second stage of this experiment, which involved the testing of the ETA neural network on the same data used in stage 1 to test human subjects, indicated that the network is strongest in the recognition of fear, surprise, happiness and sadness. The neural network displayed problems mainly with data sets representing anger and to a far lesser degree disgust. The overall regognition rate recorded during the test was 96% (that is 96% of all 270 facial expression were correctly identified), indicating that the neural network was superior to the performance of the human subjects in stage 1. This result was again verified statistically using a Chi Squared analysis.

Implementation Study

After accuracy of the ETA neural network was determined, an implementation study was then conducted with the objectives of illustrating the use of the ETA agent in its intended role and performing an evaluation of the completed prototype agent. The study involved evaluation of the provided benefits of the ETA software agent to a multitasking user's understanding of information presented to him/her in the course of the study and his/her perceived effectiveness of the agent in enhancing emotional awareness and attention in the virtual test environment. The study involved the integration of the agent into VRCGroups. VRCGroups is a fully functional LAN based collaborative interaction environment which provides text and voice chat facilities along with application and work sharing tools for user collaboration.

By analysing the experiences of a number of individuals in the VRCGroups environment, both with and without the use of the ETA, we intended to measure their emotional awareness under a multitasking remote meeting situation and evaluate the impact of the agent on their communication abilities in such environments.

The study comprised of two stages, involving a number of participants engaging in a remote meeting requiring them to juggle multiple tasks in parallel while simultaneously paying attention to another user's presentation. The first of these stages required the participants to complete all tasks without the help of the ETA and the second repeated these tasks a month later with the ETA functionality included (a month between stages monimised the learning effects which may have skewed the results of the study). A mixed quantitative and qualitative questionnaire was used to determine a participant's emotional awareness in both stages and to determine the participant's perception of the meaning of presented messages during the study. Another survey was used at the end of the second stage to allow participants to give feedback on the perceived benefits or disadvantages instigated by the introduction of ETA functionality. A side-task in the form of a short ten question IQ-style test was also given to the participants to complete in parallel with the main emotion identification and perception survey to induce multitasking. The results of this side-task were not important but the effect it had on the participants of the study was imperative.

Implementation Study Results

During the analysis of the study results a number of interesting observations were made regarding the effect of the ETA on subject's attention and awareness in a remote collaborative and communicative interaction. During the study the subjects were required to identify the emotions portrayed in the voice of the presenter (who relayed information to the subjects from the CVE) at set points in the course of the each stage. The subjects were also instructed to describe the meaning of what the presenter was saying at these set points in order ascertain how well they understood the underlying messages portrayed. There were 19 of these points in total during each stage of the study. In the first stage, where the ETA was not used to help participants, the overall average emotion identification accuracy observed was 60 percent, which is considerably lower than the accuracy observed in the previous experiments from facial expressions. The results from the emotion identification portion of this stage are shown in Table 1 in terms of correctly identified, logically misidentified (such as anger mistaken for disgust), entirely incorrect and totally misidentified emotions. Although the number of participants in this study is too few to make any definitive conclusions regarding the ability of humans to recognise emotion from speech, this result still implies that the level of emotion conveyed from a person's voice is considerably lower than that conveyed through facial displays.

Table 1: Compiled results for individual subjects during initial emotion identification stage of Implementation study

	Total Correct	**Total Incorrect**	**Logically Incorrect**	**Incorrect entirely**
Subject 1	11	8	3	5
Subject 2	8	11	2	9
Subject 3	10	9	3	6
Subject 4	13	6	3	3
Subject 5	15	4	2	2

Emotional Awareness and Understanding Observations

It is clear from the results displayed in Table 1 that the best performing participants of the study were Subjects 4 and 5, who incidentally were both of Chinese nationality. From further analysis of their descriptions recorded during this study stage, it became apparent that although their emotion accuracy from vocal tone was superior to the remaining subjects, their understanding of the underlying meaning of the messages was in fact inferior. This was likely due to the fact that English was not their first language which may have left them at a disadvantage. Comparisons made between these subject's first stage descriptions and those recorded in the second stage, show a considerable increase in understanding, suggesting that the inclusion of the ETA was of significant benefit to these two subjects. Although the small number of subjects used in this study is not enough to make definite conclusions, this result still suggests that there may be possible augmented usefulness of such an emotion tracking agent in online interactions for non native English speakers.

In relation to emotional awareness, a number of subjects remarked in interviews that once the ETA informed them of the emotion being portrayed on the presenter's face during the interaction, they were more aware of that emotion in the presenter's vocal tone and the emotions in his voice seemed clearer and more pronounced. It was also noted that the ETA made it easier for them to recognise and distinguish problem emotions such as the negative emotions of anger, disgust and fear which were identified by subjects as being harder to differentiate or detect from the presenter's vocal tone alone in stage 1 of this study. Positive emotions such as happiness and surprise were noted as easier to recognise from vocal tone than their negative counterparts.

The effect of the information offered by the ETA was evident from the descriptions given by the subjects in the second stage of this study. This was seen in the subject's use of the names of emotions prompted by the ETA in their described meanings recorded in this stage.
Analysis of participants' descriptions in both stages of this study also indicated that certain linguistic modes were unfavourably influencing and complicating the subject's interactions. The most dominant of these linguistic modes is the manifestation of sarcasm in the vocal tones of the presenter's voice.

Attention and Certainty Observations

The use of more concise and detailed language during the second stage of this study by the majority of subjects also indicated an increase in understanding of the underlying message portrayed by the presenter. The usage of words such as "might be", "seems", "maybe" or "it appears to be" are prevalent in stage 1 giving the impression of a lack of certainty and sureness in the subject's descriptions. During the second stage it was clear from the predominance of terms such as "definitely" or "does not" or "is not" or "is", that there was a clear increase in the subject's confidence and certainty in their answers. This improvement in the certainty levels of the study participants suggests a positive reaction to the inclusion of the ETA which complements the previously observed increases in the subject's understanding and emotional awareness.

It was also observed that in stage 2, subjects tended to more often reference multiple elements of a section in their answers, and appeared to be more capable of describing the presenter's attitude to these individual elements; this was not a phenomenon observed to such a degree in stage 1. An increased ability to break up the message into separate informational strings also indicated an increase in understanding of the underlying meaning which may be due to an improvement in the subject's attention to particular details of the presented information. Many times in stage 1, the descriptions given by subjects, were unspecific and used general terms such as "this statement" to refer to the entire message as a whole. Such language use implied either the participants were not paying full attention or they were unable to adequately understand the information given.

Multitasking Observations

It became evident the further the implementation study progressed, that the experimental setup used may not have simulated multitasking to a sufficient degree since some subjects tended not to task switch, but rather to finish one task before starting another. This may have been due to the amount of concentration required for completion of the main task. This was apparent in the fact that some subjects noted in interview that the first stage without the use of the ETA, caused them to have to concentrate more on the presenters information task but in the second stage with the help of the ETA's prompting, they were capable of concentrating on the side task more due to the abundance of emotional information provided which made the understanding of the meaning of the presented information much easier. Subjects stated that the juggling of two tasks was much easier if one of the tasks was aided by the ETA and it was clear when the increased accuracy of the subject's descriptions in stage 2 is considered, that the overall interaction seems to have benefited significantly.

Another remark made by subjects in interview indicated that concentration was never really a large problem in either stage but they did note that it was definitely easier to focus and concentrate in the studies second stage. Some subjects revealed that their strategy for dealing with the side-task in the first stage consisted of choosing the easiest of the given questions to perform first, in order to allow more time to concentrate on the main presentation task. This was apparently not a problem during the second stage and most subjects stated that they tended to go from the first question sequentially to the last due to the lack of need to concentrate as much on the main task to the degree required in the initial stage.

CONCLUSIONS

We believe the the ETA solution has been successfully demonstrated to allow remote collaborators to more easily perform multiple tasks when one or more of these tasks involve a communicative interaction supported by the ETA. The results of this work and experiments have illustrated that the application of such an emotion tracking agent to remote collaboration tools has the capability to increase collaborating user's understanding of vocal communication and increase their attention and ability to concentrate on the ETA supported interactions while engaged in side-tasks. Increased communicative abilities between collaborating users will help bridge the gap between real world and virtual world interactions and if such an agent is developed further to a commercially available product built on commonly available hardware then it would represent one step closer to ensuring the continued usage and growth of the remote collaboration industry.

The research related to the tracking and recognition of non-verbal displays of emotion carried out during this research project has demonstrated the importance of facial expression in the portrayal of emotion and meaning in an online communicative interaction between two or more collaborating persons. It was evident from observations made during the implementation study that without the accompaniment of facial expressions during a communicative interaction, the listening parties may exhibit difficulties in extracting the full meaning of vocally communicated messages. Clear improvements to user's understanding and confidence in the correctness of extracted meaning were observed upon the introduction of the ETA prototype into the communicative process. During the implementation study it was also observed that the non-verbal feedback provided by the ETA adds to a user's ability to more clearly understand messages from other participants where the person's facial expressions are not directly in view of the user.

FUTURE RESEARCH

A wide range of fields have been drawn upon and applied to this research and as a result a wide variety of important research problems have been uncovered. A number of these apply primarily to remote collaboration itself, while others are of more general interest to emotion recognition, non-verbal communication interfaces and cognitive science and psychology applications.

Uncovered limitations from assessment of the chosen research methodologies have highlighted a number of additional research studies which if conducted may benefit further assessment of the developed ETA in its use for tackling remote collaborator multitasking. Through analysis of the results, the implementation study appears to have been unsuccessful in its ability to fully simulate remote worker multitasking. To better analyse the agent's benefits to multitasking remote interaction participants, a more detailed case study is proposed in which the ETA is employed in a real collaborative research situation over an extended time period.

In order to further assess the effects of the ETA on multitasking behaviours, it is proposed that a more realistic case study is performed in future work involving the use of the ETA in a full-scale online meeting between participants collaborating on a realworld unscripted task. Such a case study will also provide insight into the nature of current remote collaboration and help measure the extent in which multitasking truly effects such interactions.

Another more obvious direction to take in furthering this research is the replacement of the optical motion capture device with a webcam using robust image processing techniques to provide feature extraction and input for the ETA neural network. This would not only be more practical and less intrusive but also more economical if such an emotion recognition agent is to be eventually developed into a commercial product. Use of the ETA without specialised equipment beyond those required for remote work would minimise the boundaries which may arise from the extra expense and expertise needed to operate specialist technology. Much work has been done in developing efficient image processing algorithms for facial expression recognition in recent years and further research into the application of these algorithms for the ETA would undeniably benefit the future of our work.

It is also evident that future implementations of the ETA should apply greater focus on the support for tracking and recognition of non-universal emotions and states such as confusion, agreement and (dis)interest. The primary emotions used in the proof of concept prototype agent are od limited usefulness in real-life collaborative interactions, and thus research in order to extend the emotion recognition architecture to better deal with such states and emotions would be of great benefit.

REFERENCES

Appelbaum, S.H., Marchionni, A. & Fernandez, A., 2008. The multitasking paradox: perceptions, problems and strategies. Management Decision 46(9), 1313-1325.

Black, J. & Hearne, S., 2008. Effective leadership of international virtual project teams, PMI Global Congress, Denver, Colorado, USA.

Chopra, P., 2009 retrieved on 3 September 2009 from http://paraschopra.com/

Ekman, P., Friesen, W.V., 1975. Unmasking the Face. Prentice-Hall Inc, New Jersey.

Ekman, P., 1980. The face of man: Expressions of universal emotions in a New Guinea village. Garland STPM Press, New York.

Lojeski, K.S., Reilly R. & Dominick, P., 2007. Multitasking and innovation in virtual teams. Hawaii International Conference on System Sciences (pp. 44b)

Pantic, M., Rothkrantz, L.J.M., 1999. An Expert System for Multiple Emotional Classification of Facial Expressions. In: Proceedings of the 11th IEEE International Conference on Tools with Artificial Intelligence, pp. 113-120

Redfern, S., Colhoun, M., Hernandez, J., Naughton, N., Noonan, D. 2006. "Distributed Virtual Reality Learning Environments", Chapter 13 in A. Cartelli (ed.) "Teaching in the Knowledge society: new skills and instruments for teachers", Idea Publishing Group, 2006.

Smith, P., Redfern, S., 2010. Facial Expression Tracking for Remote Conferencing Applications: An Approach to Tackling the Disadvantages of Remote Worker Multitasking. In Proceedings of the Second International Conference on Games and Virtual Worlds for Serious Applications, pp. 91-92

Smith, P., Redfern, S., 2010a. Emotion tracking for remote conferencing applications using neural networks. In Proceedings of the 21st National Conference on AI and cognitive science, pp.240-249, National University of Ireland, Galway.

Stergiou, C., Signanos, D., 1996. Neural networks. Final report on neural networks, SURPRISE 96 Journal 4. Department of Computing, Department of Electrical and Electronic Engineering, Imperial College of Science Technology and Medicine, Queen's Gate, London, UK. retrieved on June 28 from http://www.doc.ic.ac.uk/~nd/surprise_96/journal/vol4/cs11/report.html#The Back-Propagation Algorithm

Susskind J. M., Littlewort G., Bartlett M. S., Movellan J. & Anderson A. K., 2007. Human and computer recognition of facial expressions of emotion. Neuropsychologia 45, 152-162.

Zhang, Z., 1999 Feature-based facial expression recognition: Sensitivity analysis and experiments with a multilayer perceptron. International Journal of Pattern Recognition and Artificial Intelligence, 13(6): pp. 893-911.

Zucker, U., Radig, B. & Wimmer, M., 2008. Facial expression recognition-A comparison between humans and algorithms. Retrieved on January 7 from http://www9-old.in.tum.de/people/wimmerm/lehre/sep_zucker/sep_zucker.pdf

A GAME SYSTEM APPROACH FOR TRAINING AND EVALUATION: TWO SIDES OF THE SAME COIN

Claudio Coreixas de Moraes[1]
Daniel de Vasconcelos Campos[2]
Roberto de Beauclair Seixas [2,3]
Michael Aaron Day[1]

[1]Naval Postgraduate School – NPS
1 University Circle, Monterey, California 93943, USA
e-mail: ccoreixas@nps.edu

[2]Computer Graphics Technology Group – TECGRAF
Pontifícia Universidade Católica do Rio de Janeiro – PUC-Rio
Rua Marquês de São Vicente 255, Rio de Janeiro, RJ, Brasil 22453-900
e-mail: rbs@tecgraf.puc-rio.br, danielaer@ig.com.br

[2,3]Institute of Pure and Applied Mathematics – IMPA
Estrada Dona Castorina 110, Rio de Janeiro, RJ, Brasil 22460-320
e-mail:rbs@impa.br

KEYWORDS

Training systems, tracking, serious game, virtual simulation.

Abstract

Imagine yourself trying to interpret a theoretical sequence of actions for a given task from a procedures manual. The task implies doing a mental simulation of the spatial representation of all the theoretical actions. Visualizing complex spatial tasks is hard; practicing them in real life for the first time is even more complicated and many times involves correction via feedback from an instructor. We found that two systems that are under development as research projects that can be matched to improve instruction for cognitively complex tasks. Our objective in this paper is an attempt to combine a command and control system and a game-based ship simulator which will create an architecture for a powerful Live Virtual (LV) simulation system. Using this framework, we can bridge the gap between practical and theoretical instruction by providing instantaneous feedback to the instructor and After Action Review (AAR) to the student.

INTRODUCTION AND GOALS

The instructional cycle involved in a navigation and ship handling learning environment, such as a Naval Academy, requires many steps. The first happens inside a classroom where students make first contact with the theory behind how to handle a ship. In a typical a Naval Academy class students' ages range from 18 to 22 years old; a group of young minds thirsty for more action than a slide show can provide inside a classroom. It is difficult to convey the principles of a large complex system floating in the water without brakes and tires moving on a imaginary highway in a classroom [7] [2]. In order to make the sometimes boring components of navigation and ship handling training understandable, hands on training is required. Most of the Naval Academies worldwide adopt hands on training aboard small Yard Patrol Boats (commonly known as YP) specifically designed for this type of instruction. They function as floating labs were students can experiment and put into practice the abstract concepts of the physics of a ship, and the procedures and rules of the road to safely move a ship from point A to B. Simulation can fill the gap between classroom instruction and hands-on training. This is exactly the concept that drives the design of Yard Patrol Simulator (YPSim), a game-based training system that is under development at the MOVES Department of the Naval Postgraduate School (NPS, Monterey, CA - USA) as a Master's degree project. On the other hand, the Command and Control Support System (CCSS), another Master's degree project developed at the PUC-Rio (Rio de Janeiro, RJ - Brazil), is a tool designed to provide Command and Control (C2) information in a 2D representation [4]. The CCSS design is based on the OODA loop (Observe-Orient-Decide-Act) [3], and presents a virtual environment rich with information necessary for decision makers act on the observation and orientation steps of the process. The type

of information that CCSS primarily handles is for battlefield situations, where unit positioning, weapon range, communications range, and valuable resources can be monitored. Managing this whole set of data rapidly can be crucial and decisive in the course of actions of a military campaign. The CCSS tool is a facilitator in the process, using computational and graphical techniques. To see the value of combining CCSS with YPSim into a new type of VE training tool, it is necessary to understand instruction from two geographically isolated perspectives. From the YP Boat perspective we have an instance of the CCSS onboard the YP, registering and streaming data from GPS, Radar, AIS, engine, rudder and audio/video. From the onshore classroom perspective we have a group of instructors and/or other students observing the training exercise in a close to real time 3D graphical representation of the YP, its state (engines and rudders), other contacts, and audio/video from the bridge, presented via an instance of YPSim. This architecture can optimize the flow of information feedback from the instructor, allow students to share experiences, and facilitate performance evaluation and AAR during a debriefing phase. Allowing the classroom instructor to view a virtual appraisal in a Live Virtual (LV) simulation is something new. Due to physical limitations of the YP and time constraints involved with being onboard, most of the time the classroom instructor is not present at the hands on training exercises, breaking a valuable link in the learning process. The integration between the two subsystems is possible via a 3G network connection using the commercial cellular infrastructure. The nature of the hands on training missions onboard the YPs allows good 3G coverage (less than 10km from urban coastal areas). The data exchange is handled by UDP packets in a Client/Server connection in a close to real time transmission rate.
This work will represent a completely new dynamic in the instruction of navigation and ship handling, where classroom instructors are able to effectively participate and observe students' application of understanding theory.

RELATED WORKS

The use of Open Source software to develop visual simulation designed for naval training was described by Salvatore [14] and Ernst [5] in two Master's degree theses at NPS. The Live Virtual (LV) simulation concept explored in this work is derived from Noseworthy's [10] understanding of Live Virtual Constructive (LVC) simulation. The core of this paper is almost a continuation of Fruchey's efforts to describe the possibilities of integration between LVC simulators for training [6]. In his work, Fruchey describes a very large scenario for battlefield training simulation where LVC instances would be connected to improve the training capabilities of the exercise. He believed that collecting sensor information from real components in the simulation would be a valuable tool for instructors' and students' analyses during an exercise. The challenge, described by Fruchey as "the most critical", would be the integration of all the components exchanging the data collected. Our work is a simplified extension of Fruchey's approach.

TASK ANALYSIS

It is necessary to provide an overview of what kinds of tasks are performed during hands on training onboard the YPs. Having this concept cemented in the reader's mind will lead to a better understanding of why the proposed integration is important. In order to achieve this goal, we decided to design a very brief task analysis of a general basic instruction at sea, using the Brazilian Naval Academy's (BNA) framework as a base case.
The YPs are the laboratory where students will experiment and effectively learn how the ship's dynamics work, the forces and responses coming from the engine and rudder, how wind and sea currents will affect ship's behavior, and so on. We can observe an example of this kind of vessel in Figure 1. The common situations trained onboard the YPs at the Brazilian Naval Academy are Navigation, Man Overboard (MOB), Mooring, Anchoring and Replenishment at Sea (RAS). These represent the very basic skills that a young line officers need to be proficient in when graduating from any Naval Academy. The focus of the present work is not on the use of YPSim as a tool for the student, but helping the instructor in visualizing and virtually participating in the hands on training process.

Figure 1: The Brazilian Naval Academy's YP (left) and midshipmen at hands on training onboard (right).

Among the training situations listed above, hereafter called "missions," the restricted waters Navigation is the most interesting. Understanding what kinds of tasks are performed, by both student and instructor, during the Navigation mission will guide the analysis of the proposed system architecture (further detailed). Even though students go onboard as a group with different roles for each one of them (representing a navigation team at a ship's bridge), this paper is more focused on the Conning Officer duty. The Conning Officer role is played by one of the students (usually senior grade) who is responsible for maneuvering the ship. He will issue commands for the helmsman and lee-helmsman concerning engines and rudder respectively. The major goal of the student during Navigation mission is to move the YP from point A to point B, following a given route and respecting the rules of the road applied to general navigation to avoid collision with other contacts. The Conning Officer (our student) will receive suggestions for course and speed from the navigation team (who are responsible for accurately establishing the YP's position on a nautical chart). He/she then decides to act or not by changing engine and rudder status, reflecting on speed and course changes in order to achieve his/her current sub-goal: reaching the next waypoint in the route or avoiding contacts deemed risky. This process is continuously repeated until sub-goals are achieved and eventually the YP reaches the end of the navigation route.

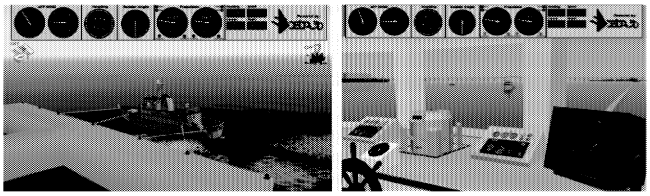

Figure 2: Yard Patrol Simulator

At the other side of the instruction process is the instructor. His/her role during the hands on training is a valuable resource to the student in case of doubt or an emergency situation. The instructor's role is not limited to correction of the student's actions or assistance during an unexpected situation. The instructor also observes the student executing his/her plan for conducting the YP from point A to point B. The expertise of the instructor gives him/her sufficient background to qualitatively judge the performance of the student. After every training mission onboard the YP, there is a rich list of topics (both positive and negative) that could, and should, be explored by instructor in an After Action Review (AAR) debriefing. Some cognitive scientists identify this phase of training as one of the most important in the learning process, when the student is able to explore mistakes made during intense cognitive processing and correct his/her actions in the future [1] [12]. Positive reinforcement coming from the instructor is also present during AAR and serves as an important motivational component. One of the biggest problems for conducting the AAR onboard the YP is the lack of a visualization environment where both instructor and student can review a specific situation early in the mission and observe detailed information, such as a turn that was made a few seconds after the ideal time. The heavy cognitive workload that a student, especially a novice, is subject to during missions makes it even harder to remember and associate exact situations with the reinforcements coming from the instructor. The use of a VE representing the mission scenario during AAR would be a valuable tool that could help the student's recall, using spatial information for an enhanced debriefing. CCSS is a system that is primarily designed to control the flow of information in a saturated complex environment, such as a battlefield [4]. Using CCSS' functionality aboard the YP, instructors would be able to filter, organize and share training information, addressing some common issues faced by both instructor and student. Both instructor and student actions are following an OODA cycle that could be enhanced using CCSS. Another important aspect of the tasks conducted by the instructor during hands on training is real time feedback. This type of reinforcement is provided by the instructor a short time after or before a given situation. Usually real time feedback is given in situations where safety is a concern and a student's actions lead to negative training.

ARCHITECTURE

This section introduces a detailed description of the YP-Sim and CCSS systems, and the proposed Integration Interface. The operation, capabilities and limitations found in both systems will be discussed, focusing on their use as an instructional tool as an integrated package.

Yard Patrol Simulator

Yard Patrol Simulator (YPSim) is a game-based system that is under development as part of a Master's degree thesis at the Modeling in Virtual Environments and Simulation (MOVES) Department at the Naval Postgraduate School (NPS). The application is designed to provide a 3D Virtual Environment using Delta3D [8] [14] [5] open source game engine with a real time physics model of a Naval Academy Training Ship having multiple scenarios as shown in Figure 2. The software's main use is focused on single-user missions where Naval Academy midshipmen (the primary user population) can practice and experiment navigation and ship handling concepts previously acquired in classroom instruction. YPSim hardware requirements allow it to be used on a regular PC with graphics cards supporting OpenGL 2.1 or above and CPU greater than 2.0 GHz. Users can practice some of the maneuvers they will be tested on during the hands on part of their education. From a regular navigation mission, where you can learn about ship dynamics and rules of the road, to a complex Replenishment at Sea (RAS) type of maneuver, YPSim offers an experimental lab where midshipmen can learn practical concepts at a level between classroom and hands on training. Regular LCD monitors are used to provide visual output to the system while regular keyboards/mice are used as default input devices. Using UDP packets under a Client/Server architecture, YPSim provides multi-player capabilities, allowing students to play missions where two or more units are re-

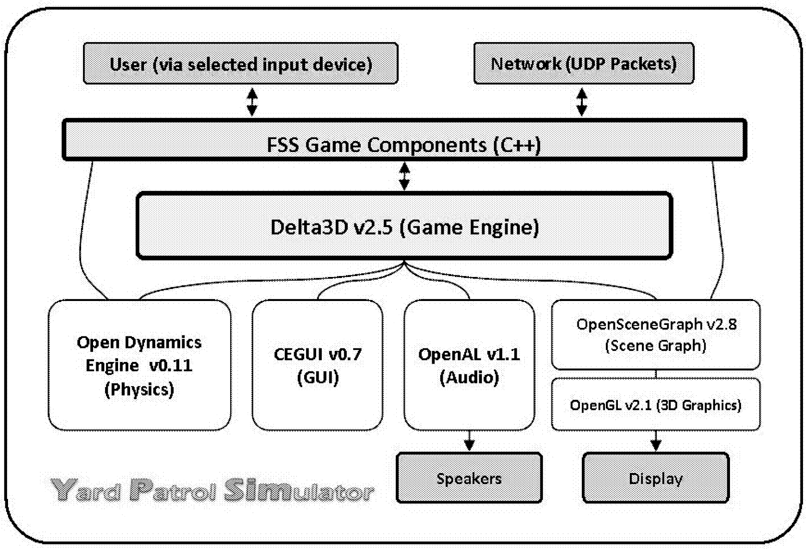

Figure 3: YPSim Overview

quired (like the RAS mission).

Features of YPSim of special interest to an instructor are as a remote 3D scene representation of hands on training. YPSim has a built-in radar module that simulates the YP's navigation radar, being able to receive and plot Radar/AIS (Automatic Identification System) contacts information coming from the real YP. In this way, instructors would be able to share the same contact lists, facilitating the understanding of student's current plan of action. Another valuable source of information is the camera positioning flexibility provided by YPSim, where user can toggle between 4 modes: first-person (inside the bridge), third-person following the ship at a fixed position, third-person orbiting around the ship, and third-person using a fly motion model. Analyzing the scenario from different perspectives is a very important feature regarding AAR in a debriefing phase, when instructors can point out students mistakes from different perspectives. Accurate 3D models of the area of operation can represent position of aids to navigation (buoys, lighthouses, alignments, and land marks) replicating the real world scenario. YPSim uses OSGOcean to render the ocean surface and affect ship's dynamic response to swell. A situation display constantly presents information about the engines' RPM, rudder angle, heading and depth – providing a good understanding of the ship's status.

Command and Control Support System

The Command and Control theory of John Boyd [3], a 20^{th} century military strategist, allow us to introduce computing techniques that are able to speed up the OODA loop (Observe–Orient–Decide–Act), especially on observing and orienteering steps. Command and Control Support System introduces a low cost framework capable of monitoring people, vehicles, boats, or any other elements of interest, almost in real time. The goal of this design is to gather and present the best possible information for decision makers.

For a better understanding about Command and Control Support System, it is essential to explain its pipeline. Looking at the Figure 4 we can identify two main blocks: monitored element(s) and stateful station. The monitored element contains some portable electronics devices such as a small computer, a 3G modem and a GPS logger.

As the proposed framework must be able to monitor from military vehicles or vessels to a soldier, we had to look for hardware that possessed the following characteristics: (a) light, (b) small, and (c) good battery life

So, we found on the market the Igot–U GT–600 GPS logger, the PC2 computer, the Tekkeon MP3450 R2 battery, and the MIMO monitor according to these requirements.

We should note that these characteristics are mainly required by "monitored elements", as the "stateful station" could be installed in a room. In this case, the devices used need not be light or use external batteries.

Figure 7 shows the hardware used by Monitored Element(s).

The main idea was to write a script, in our case we use Lua [9], able to do the parsing of National Marine Electronics Association (NMEA) data, received by the GPS logger, and transmit/store a record with information such as current speed, latitude, longitude and travel direction. After the parsing we insert all this information in two databases: local (monitored element) and remote (stateful station). The connection with the remote databases is established through a 3G modem. Actually, any remote connection technology can be used (wi–fi, radio, satellite...).

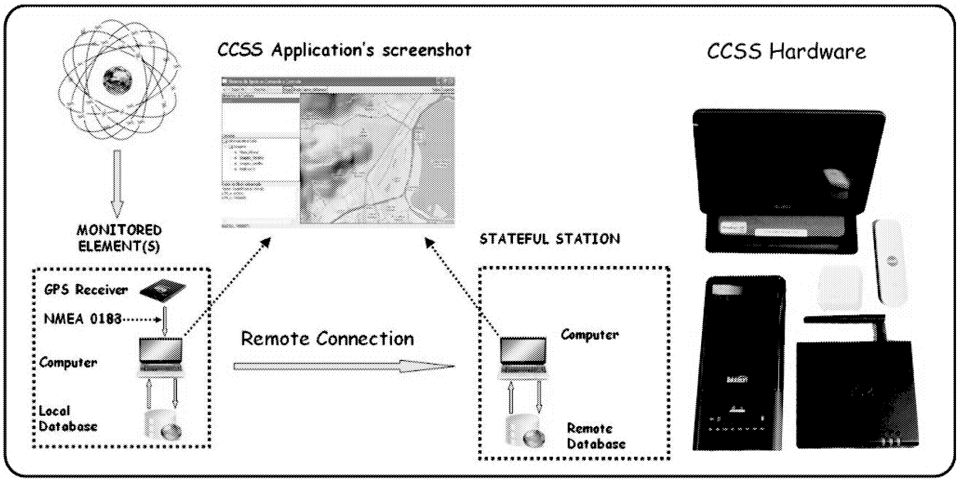

Figure 4: Architecture of the system to support C2

Using a specific application built to handle such data, both members of the system (monitored element and stateful station) can access a display that shows an icon representative of the georeferenced monitored element. The framework is still able to keep audio and video communication between the monitored element and the stateful station through any communication software such as Skype, MSN, Ekiga...
An example use case can be found in [4]. In this case, the application developed has the goal of providing, in a unique environment, some relevant information to the high command, such as:

- 1– Sketch graphically, in interactive time (near real time), the monitored elements of a force deployed in theater operations as well as the information about speed and direction of travel of these elements;
- 2–Provide information graphically, about the line-of-sight, the weapon range and the radio range of the monitored elements;
- 3–Present thematic layers with various coordination and control measures used in military operations, such as goals, boundaries, landing beaches, among others;
- 4–Record in a database positions occupied by monitored elements allowing the reconstruction of their itinerary;
- 5–Enable virtual position simulation of monitored elements, allowing the establishment of retransmission stations.

We can observe a screenshot of this application on Figure 5.

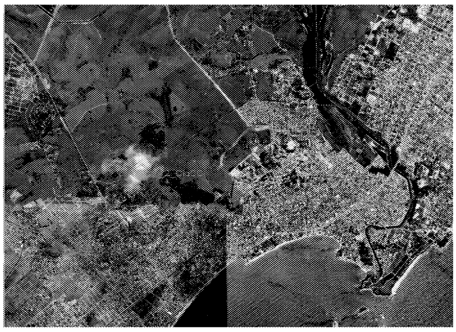

Figure 5: Weapon range of 2_CiaCC(in red)

Integration Interface

Despite both systems' (CCSS and YPSim) network capabilities, they are not primarily designed for this interoperability mode. YPSim and CCSS do not share the same network protocol for message traffic and this is the major issue to be addressed in this work. Another consideration needs to be made regarding both applications' handling new data fields that are required for this integration. In this subsection we will present solutions for a common protocol and what changes need to be made in both applications, such that the integration is possible for this specific use as a training tool.
The first consideration is about the audio and video connection. CCSS relies on freeware VoIP applications such as Skype or MSN to handle this communication between instances [4]. This work is not intended to modify this simple, but efficient solution preserving the same concept to the proposed architecture. Regarding the YP state data package (position, course, speed, engine RPM, rudder angle, etc), CCSS originally used TCP socket connections between its instances for messages exchange. This approach assures reliability in the network but has the drawback of higher latency, a big issue for real time simulations [11]. YPSim, at the other hand, adopts a UDP socket connection using a very simple and fast protocol composed of two types of data packets in a Client/Server connection. It is clear that both systems cannot talk to each other without changes in the

network implementation following protocol standardization. Between the reliability and high latency of TCP and uncertainty and low latency of UDP, we opted to use UDP sockets in our integration. The system relies on a Client/Server architecture with very simple packets. The Server, represented by an instance of YPSim running in a classroom or office, opens UDP port 3000 and waits for a connection. The Client, represented by an instance of CCSS running on board the YP during the training exercise, requests a connection by sending a CONN_REQUEST packet containing its ID number. Once the Server acknowledges the request and accepts it by sending a CONN_ACCEPTED packet, the Client starts sending data packets containing information for the virtual simulation on YPSim. In order to represent the VE instance of the training exercise that is happening on board the YP, CCSS sends out two types of data packets: YP_DATA and CONTACT_DATA.

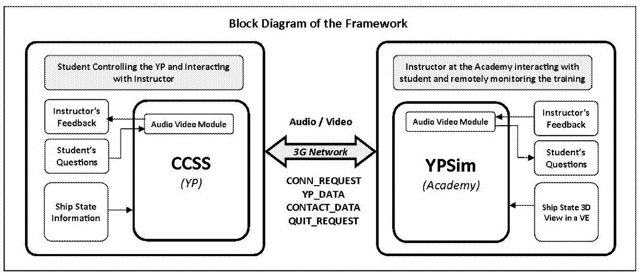

Figure 6: A block diagram representation of the integration

The first packet, YP_DATA, holds information about the YP state at a given moment that will make possible the virtual representation of the YP in YPSim. The virtual simulation needs information about what is happening at that moment inside the YP concerning the navigation/ship handling exercise. In order to do that, this packet contains the following fields:

- Header (packet name + YP's international call sign);
- YP's position (LAT/LONG);
- YP's heading (0-359);
- Course (0-359);
- Speed (in knots);
- Turning rate (in degrees/sec.);
- Linear Acceleration (single value for speed);
- Angular Acceleration (single value for heading);
- Engine RPM (two floats, one for Port Engine and one for Stbd Engine);
- Rudder Angle (one float, assuming both rudders acting in sync);
- Apparent Wind (two floats, one for speed and one for direction).

YPSim uses a transmission rate of one data packet every 15 frames, or two packets per second considering an average of 30 frames per second under normal operating conditions. This heartbeat rate [13] was proven to be satisfactory in previous tests of YPSim thanks to the low velocities (linear and angular) involved in the simulation and second order dead reckoning algorithm used (accounting for velocity and acceleration) [13]. Using the information provided by CCSS running on board the YP, the instructor is able to understand and better visualize the YP's location, its movements, and governing actions applied by the student (rudder and engines). The instructor virtually follows the student's actions and qualitatively evaluates if his/her decisions meet the best practices according the theory of ship handling and navigation. The apparent wind information also provides a valuable resource to better understand the environment where the exercise is taking place.
The second packet, CONTACT_DATA, holds information about other ships in YP's vicinity. This information can be extracted primarily from an Automatic Identification System (AIS) equipment, when installed on board the YP. Information about contacts close to the YP help the instructor to visualize the scenario and understand student's intentions while maneuvering. This data packet can be less complete and contains only the basic information necessary to make corrections in YPSim, applying only a first level of dead reckoning via velocities (speed and turning rate). The following fields will represent our CONTACT_DATA packet:

- Header (packet name + contact's international call sign + type of ship);
- Dimensions (length and beam);
- Position (LAT/LONG);
- Heading (0-359);
- Course (0-359);
- Speed (in knots);
- Turning rate (in degrees/min.).

A secondary source of contact information, for building a CONTACT_DATA packet, can be navigation radar equipped with Automatic Radar Plotting Aid (ARPA) capabilities. When using Radar as a data source, the Tracked Target Message (TTM) NMEA sentence is parsed and position is converted from bearing and range to Lat/Long. TTM sentences coming from radars do not provide as detailed information about the target as the AIS does, therefore these fields (call sign, type, dimensions and turning rate) in this package will be filled with 9, flagging radar as the source. Because of the nature of the source of information (AIS or Radar), contacts are not updated as frequently as the YP itself. This implementation will consider a fixed heartbeat of 5 seconds

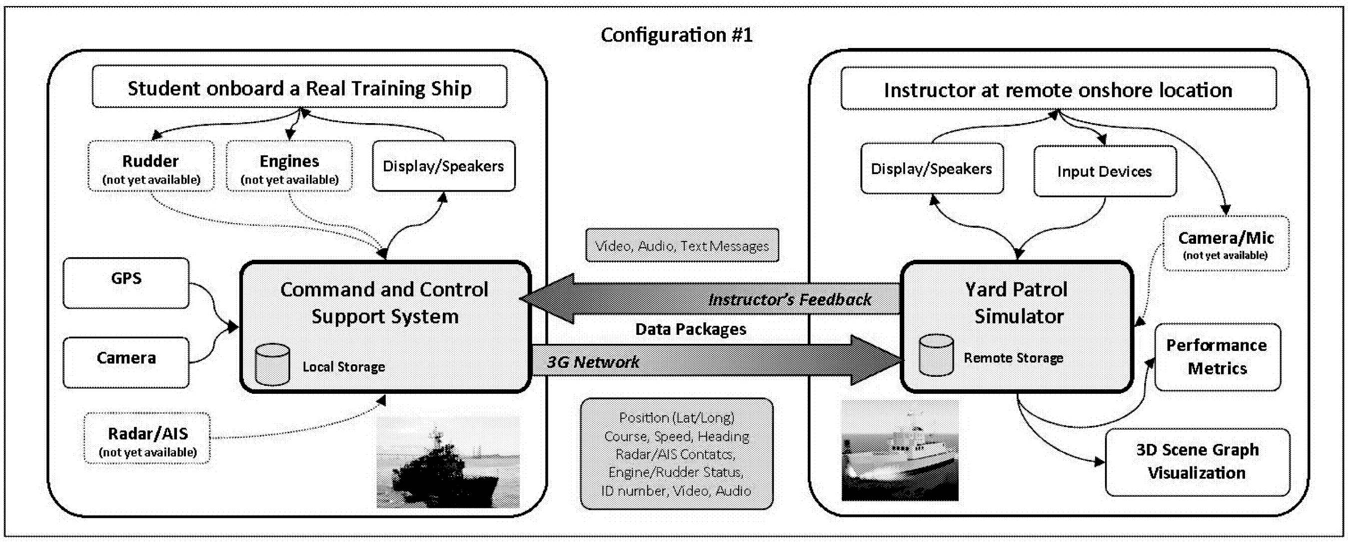

Figure 7: The integration between CCSS (student onboard the YP) and YPSim (instructor remotely onshore)

between data transmissions of the same contact. Position and orientation of contacts between updates are calculated using a first order dead reckoning algorithm that uses only course and speed as parameters [13]. Contacts that are not updated after a timeout of 120 seconds will be automatically removed from the virtual simulation running on YPSim.

Another auxiliary packet is required to remove entities, both YPs or contacts, from the simulation. The QUIT_REQUEST packet is responsible for this job, transmitting to the Server the ID of the entity to be removed, i.e. entity's call sign. Upon receiving this packet, the Server (YPSim) automatically removes the entity from the world without sending any acknowledgement message back. After sending the QUIT_REQUEST packet, the Client automatically stops sending updates relative to that contact or itself. The QUIT_REQUEST is just an attempt to advise the Server that the contact's updates will cease. If the Server does not receive this UDP packet, a timeout will fire and the entity will be removed after 120 seconds of inactivity.

CASE STUDY

A typical application of the proposed framework is for a student that needs to learn how to execute a precision anchoring task. As the Conning Officer of a YP, he/she learned a series of complex procedures to correctly move the YP from a given position A to the anchoring position B. Ship handling manuals could vary a little on how to execute this maneuver, but the precision anchoring task will be composed of basic navigation (as described in Task Analysis Section), with systematic speed reductions as the anchoring position approaches. A set of verbal instructions must be passed to the Boatswain, located in the YP's bow, reporting readiness for anchoring as the ship approaches the goal position.Suppose a precision anchoring task is executed on board a given YP under local supervision of the YP's Commanding Officer (CO) and remotely monitored by the student's classroom instructor located at the Naval Academy's navigation laboratory (NavLab). One week before, the Naval Academy Instructor taught all the procedures of the precision anchoring task to student and his classmates. The instructor is not confident that his class mastered this topic, since it has some complexity that is hard to memorize from classroom slides and procedure manual pictures. This week students will be tested on this topic during a hands on training exercise on board the CO's YP. The instructor and a group of students will be able to remotely observe this activity by using a version of YPSim that is installed at the academy's NavLab. This instance of YPSim is linked to a version of CCSS running on board the YP, transmitting data for position, engines, rudder, apparent wind and other contacts tracked from Radar or AIS. YPSim and CCSS will also be connected by audio and video streamed using the same link, a 3G commercial cell phone network. The instructor hopes to gather enough information to evaluate how well his instruction was retained from one week before, focusing on improvements for future classes. He also wants to give a chance to highly motivated students from other classes to remotely observe this maneuver, hoping to trigger discussions about the actions taken on board. Even though the YP is 5 Nautical Miles (NM) from the Naval Academy, the instructor can interact with his student and the YP's CO.

Imagine now all of the many actions that occur in a given training exercise on a given day. In a debriefing with the instructor the next day of class, it is unlikely that a sufficient part of the instruction will be retained. It is true that good and valuable feedback is provided by the YP's Commanding Officer during the hands on training or even minutes after the exercise. The point explored in this framework is the possibility of recording exercise data in a database for later debriefing and a possibility of interaction in real time between the classroom instructor and his/her student, a key link in the instructional process. The instructor has now a chance to (at any later time) evaluate the level of instruction provided in the classroom, check if the methods used fit the hands on training expectations and also qualitatively evaluate how the students transfer the knowledge from the theory to the practice.

CONCLUSIONS

This work presents a framework that integrates two conceptually different systems that could work for the same purpose: improving navigation and ship handling instruction at Naval Academies. Integration between CCSS and YPSim is possible using UDP socket connections via a 3G cell phone network with a few changes both of the existing software infrastructures. An integration interface can be created, making data transfer possible from a YP engaged in a training exercise to the monitoring station at the Naval Academy. It is important to reinforce that the Live Virtual (LV) simulation framework presented in this work is generalizable. In this work we used an existing ship handling and navigation simulation (real YP + YPSim), however environments such as medical, battlefield or flight simulations, could also benefit. Virtual representation of the monitored student during hands on training expands the eyes of the instructor to a new horizon that extends beyond the walls of a classroom, providing a better instructional environment for complex tasks.

FUTURE WORK

The concept proposed in this work needs experimental proof. Higher levels of interoperability with other Live Virtual Constructive (LVC) simulations could be achieved using DIS or HLA protocols, instead of a simple UDP. CCSS is not ready to handle AIS and TTM NMEA sentences, but an extension of the original parsing capabilities of CCSS could be made. There is no reason not to have multiple Client connections with the YPSim Server, although this work was focused on an architecture using a single instance of CCSS connected to YPSim. The use of multiple instances of CCSS and YPSim could be useful in future work exploring tactical maneuvers exercises, where more than one YP is engaged in the same task.

References

[1] Annet, J. 1989, "Training Skilled Performance. Acquisition and Performance of Cognitive Skills".

[2] Barber, J. A. 1934, "Naval Shiphandler's Guide".

[3] J. R. Boyd. *The Essence of Winning and Losing.* 1995.

[4] Campos, D. V.;Seixas, R.B. Command and Control: A low cost framework to remotely monitor military training. In *Spring Simulation Multiconference – SpringSim'11*, 2011.

[5] Ernst, R. B. 2006, "The Design of a Stand-Alone Division Tactics Simulator Utilizing Non-Proprietary (Open Source) Media and Iterative Development"

[6] Fruchey, D.B. 1994 , "Distributed simulation for live training"

[7] Hooyer, H. H. 1983, "Behavior and Handling of Ships".

[8] Delta3D 2011, http://www.delta3d.org. Retrieved on 2011-04-13.

[9] R. Ierusalimschy. *Programming in Lua.* Lua.org, 2006.

[10] Noseworthy, J. R. 2008, "The Test and Training Enabling Architecture (TENA) - Supporting the decentralized development of distributed applications and LVC simulations"

[11] Sales, L. M., Almeida, H. O., Perkusch, A. 2008, "On the performance of TCP, UDP and DCCP over 802.11g nNetworks"

[12] Percival, F., Ellington H., Race, P. 1993, "Handbook of educational technology"

[13] Ryan, P., Oliver W. 2006, "Modelling of Dead Reckoning and Heartbeat Update Mechanisms in Distributed Interactive Simulation"

[14] Salvatore, R. B. 2005, "Using Open Source Software in Visual Simulation Development"

MOBILE GAMING

GPS GUIDED AND TOUCH SCREEN NAVIGABLE 3D RECONSTRUCTION OF AN ANCIENT ENVIRONMENT ON iPHONE AND iPAD

Gavin Duffy, Daniel Heffernan,
Eoghan Quigley, Paul Smith
RealSim Games
Unit 201 Business Innovation Centre
Upper Newcastle Rd
National University of Ireland, Galway
Galway
Ireland

Heather King

Office of Public Works
51 St. Stephen's Green
Dublin 2
Ireland

E-mail:{gavin | eoghan | paul }@realsim.ie
E-mail: me@daniel.ie
Email: heatheraking@eircom.net

KEYWORDS
3D Virtual World apps, iOS, Heritage and Education apps, OGRE, Objective-C, C++ and C

ABSTRACT

The smartphone, enabled with GPS, accelerometer, compass, camera and 3D graphics capability has transformed how we use our mobile phones. One area that smartphone and smart 3D content can add significant value to is the area of communicating history and heritage. We can now develop apps that are akin to virtual time travel experiences. This paper discusses the deployment of the very first GPS guided full 3D reconstruction of an ancient environment on to iPhone and iPad, the Lost City of Clonmacnoise.

INTRODUCTION

Communicating the life and environments of past generations has until recently been largely done through written text. A picture says a thousand words because it is our visual sense that best helps us to understand concepts. First hand artistic records of historic environments are few and far between and so we rely on skilled artists to visually translate the interpretation of historians and archaeologists. Until recently, such interpretations have been still images. In the last 10 years, digital artists have been able to animate their 3D reconstructions and produce movie-like animations of past environments. Such animations are however passive viewing experiences. The gaming industry revolves around engagement and so has developed technologies to allow people to interact with highly realistic 3D virtual environments. When high fidelity game engine graphics, mobility, artistic skill and historical insight are combined, one can create a highly engaging way of communicating a past environment.

THE SMARTPHONE AS A HERITAGE AID

Heritage sites have been using technology to enhance visitor experiences long before the development of smartphones. The most common of such aids have been headsets with pre-recorded audio narratives which tell the story of an historical location. The visual screen associated with a smartphone now allows a picture to be relayed with that audio narrative. An application or app for short, is normally developed containing visual and audio information about the historical location.

Presently there are two main types of apps currently used for the communication of history and heritage, digital book apps and augmented reality apps. The app discussed here forms a new class of app, interactive 3D virtual world apps.

Digital book apps

A good example of such an app would be the Clare Ecclesiastical Trails iPhone and iPad app [1]. This app has transferred the information from a booklet of the same name (Harbison, 2007) about the rich collection of churches in County Clare, into a simple but engaging app. However, it has made use of the phones GPS and also provides an interactive map which can give the user direction and distances to points of interest.

An enhanced version of this sort of app would be the Dublin City Walls iPhone app [2] which has also incorporates voiceover and some pre-rendered 3D animations. The animations show the user how Dublin would have looked like at a point in time from a select number of locations.

Augmented Reality apps

These are becoming very popular and are simply layers of information superimposed on the live camera view of a user’s phone. The information relates to what the user is seeing through the camera. An innovative use of such technology would be the Haunted Planet apps [3], developed by the National Digital Research Centre associated with Trinity college in Dublin. They use ghostly characters superimposed on the live camera view to tell a story and instruct the user to collect clues as they travel around a site.

Figure 1: A view of the 'Lost City of Clonmacnoise' app in operation at Clonmacnoise. The app shows the user what would have been there some 1000 years ago, correct for the user's location and direction.

Interactive 3D Virtual World apps

This is the new category of heritage app discussed in this paper. It has not yet become established largely due to the technical challenges that a phone has in rendering 3-dimensional virtual real world environments in real-time. It does not rely on a phone camera as the entire ancient environment has been re-constructed in 3D. The virtual environment is geo-referenced enabling the use of the phones GPS and gyroscope. This means that when a user is on site, their movement on the ground controls their movement through the virtual environment on screen. When a user is not on site the user can disengage the use of GPS and use simple touch screen controls. This is its major advantage over augmented reality apps which require users to be on site as they require the live view from the camera to add context to the background. This sort of app offers far more user engagement than a digital book app and allows a user to virtually explore an ancient world. One can explore either aided by helpful guides or unaided where the user can enjoy the process of discovery as tourists do in the real world.

MAKING THE LOST CITY OF CLONMACNOISE APP

Clonmacnoise today bears little or no resemblance to the bustling early medieval city that existed between the 9th and 11th centuries. Today it is a quiet rural township in the centre of Ireland. A 1000 years ago, however, it was on the cross-roads of the two most important travel routes in ancient Ireland, the River Shannon and the Eiscir Riada (a long winding glacial esker ridge which traverses the country east-west). The Monastic City of Clonmacnoise had a Cathedral and up to 9 churches, the ruins of which seven remain today. The settlement was spread over several kilometres with a population of several 1000 people. It went into decline, however, from the 12th century onwards, with successive raids by the Normans and eventually Cromwellian forces in the 16th century ransacked Clonmacnoise. It never recovered from this and since then it has never again functioned as a working monastery.

The challenge was to reconstruct this *Lost City of Clonmacnoise* and make it available on an engaging and informative iPhone and iPad app. The app development process involved 3 main elements;

1. Digitally re-constructing in 3D the City of Clonmacnoise
2. Optimisation and Level of Detail work.
3. Porting the model on to OGRE, an open source 3D rendering engine.
4. Adding user interface elements, which include pop-up information on some selectable buildings.

Reconstructing the City of Clonmacnoise

This phase involved 3 main stages;

i) Research and consultation with known experts on the history and archaeology of Clonmacnoise to assemble a layout map (Murphy 1998, MacGowan 1985, and Manning 1998) see Figure 2.
ii) Graphically model in 3D all known buildings, tools and artefacts from the time. This was a slow iterative process with significant consultation with experts to insure that the buildings represented the best accepted knowledge of the time.
iii) Assemble a geo-referenced terrain surface and re-create surface textures of rough grass and roadways and minor pathways

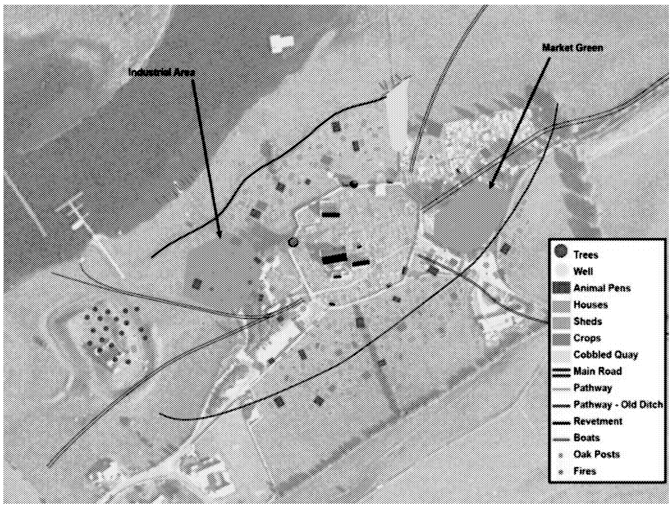

Figure 2: A schematic interpreted layout of the inner core of Clonmacnoise overlaid on an orthographic photo of the area.

Optimisation and Level of Detail

The memory limitations of the iPhone and iPad are such that the efficient use of level of detail (LOD) was absolutely necessary. It was also an aim to keep the frames per second (fps) as constant as possible. On average 20fps were achieved with no less than 10 fps at any given time. Each 3D object in the app contains four LOD as shown in Figure 3 below.

Figure 3: Four levels of detail for a typical 9th century round house at Clonmacnoise.

iOS DEVELOPMENT

The *Lost City of Clonmacnoise* app is written for iPad, iPhone 4, and any iOS device of the same or later generation. It is written in a mixture of Objective-C, C++ and C and uses many different frameworks and libraries to achieve its functionality.

In this section, the technology used will be discussed, followed by the various stages of development and finally challenges encountered in writing a 3D app for iOS and some of the solutions devised will be presented.

Survey of Graphics Engines

There is only one way to render a 3D environment on iOS: OpenGL ES [4]. However, a programmer can decide how much support is available by choosing appropriate libraries. Some libraries provide full game engines, including audio engines and physics simulation engines, and others provide only simple graphics engines.

For example, the PowerVR Insider SDK, provided by Imagination (the maker of the iPhone's GPU), provides support for 3D arithmetic, loading 3D model data, loading texture data and text rendering. This SDK is relatively simplistic and still requires the user to interact directly with OpenGL.

At the other extreme are game development tools such as Unity [5], which go beyond simple 3D programming support and include features such as 3D world building tools, user input processing, audio output, scripting, and app publishing tools.

In the middle of this broad spectrum various commercial and open source graphics engines exist, such as OGRE [6] and Irrlicht [7].

The requirements for the graphics engine were as follows:

1. Strong iOS Support Many great engines do not support iOS or OpenGL ES. Others have been ported to iOS but are poorly maintained.

2. 3ds Max Integration As 3ds Max is the tool used to create the Clonmacnoise environment, it was important to choose an engine that allowed the environment to be exported from 3ds Max with ease.

3. Customisation and Flexibility The ability to quickly fix bugs and make improvements through updates is essential for any app. Open source libraries provide this freedom, as the programmer does not have to rely on the library's vendor to make the necessary changes.

OGRE was identified as best meeting these criteria. Low-level libraries such as the PowerVR SDK do not provide any integration with 3ds Max, and closed sourced, high-level tools with limited programming APIs such as Unity do not provide the flexibility required. Ogre strikes a good balance in this area, and compared with other open source options it is well maintained by its community and has a good variety of commercial and free third-party tools and extensions available.

Designing the Content Pipeline

The content pipeline is the set of processes that are applied to an artist's content to compile it into a format loadable by the application at runtime. In this project we decided to use the commercial third-party library OgreMax [8] to export content from 3ds Max to files. These files are then processed by a set of Ruby scripts that identify and remove unused materials, as well as to convert textures to the required dimensions and format. These resources are then output to a directory that is archived into a ZIP file to be loaded by OGRE.

Handling User Input

To encourage a high level of interactivity, many different input sources were used. These are listed below in detail.

1. Touch The user can touch terrain to mark their desired destination. The app's camera will then begin moving to the position on the ground that the user touched. To move the camera orientation the user can drag a finger across the screen.

2. Accelerometer The user can opt to use the accelerometer in place of touch to change the camera's direction. By using the accelerometer, the direction of the force of gravity on the device can be measured and translated into a rotation on the horizontal axis (i.e., the user can look up and down).

3. Magnetometer To allow the user to look about the vertical axis (i.e., left and right), the magnetometer is used. This is a digital compass embedded in recent iOS devices that can give the device's azimuth as it rotates. The iOS SDK provides functionality to approximate true north based on the magnetometer's reading of magnetic north and the GPS's reading of the user's position.

4. GPS GPS input is used to control the user's position automatically. When using this functionality, the user does not touch the screen to move around in the environment, but as they walk about their position in the 3D world is constantly updated to reflect their position in the real world.

Terrain and GPS

As mentioned previously, there are two ways for the user to indicate their position in the world: direct touch input, or GPS input. Direct input is relatively easy to implement. When a user touches the screen, a raycast is performed to discover where in the environment they touched. This point is then set as the camera's position, and is transitioned to with a walk animation.

The GPS input is more challenging because GPS coordinates must be translated to a position in the 3D environment. First we define a coordinate to be the origin of the virtual word. The origin (0,0) was chosen to be an easily identifiable point on the ground, in this case, the northwest corner of the cathedral.

Degrees of latitude are parallel so the distance between each degree remains almost constant at approximately 111 kilometres apart. The range varies slightly between the equator and the poles due to the earth's slightly ellipsoid shape but the effect is so small over just a few hundred metres to render its effect insignificant for the purposes of geo-referencing the virtual city of Clonmacnoise.

However, longitudinal angles result in different ground distances depending on their latitude: one degree at the equator is a 111.321 kilometres which gradually shrinks to zero at the poles. However given the small longitudinal area covered by the 3D model, it was decided to use a constant as with the latitudinal conversion. The constant used is exact at the origin, but does introduce a slight decrease in the models longitudinal accuracy as the user moves north or south of the cathedral. This error, however, is well within the +/-10m accuracy of an iPhone's GPS sensor.

Equation 1 below shows the calculation for approximating the user's position in the virtual world. *pos* refers to the user's position: the coordinates read from the GPS sensor. *offset* is the latitude/longitude of the origin of the 3D environment. *k* refers to the pre-calculated constants discussed above.

$$(x, y) = \left(k_{lng}\left(pos_{lng} - offset_{lng}\right), k_{lat}\left(pos_{lat} - offset_{lat}\right)\right)$$

Equation 1 Calculation of the user's position in the virtual environment from the GPS coordinates

Of course, these calculations only provide the user's position on a flat plane. In order to calculate the user's position in a 3D world it is necessary to calculate an appropriate Z value too. This could be done by performing a raycast of every frame to find the height of the terrain under the user but this is a very expensive operation to perform. Instead, a height map was generated from 3ds Max which is loaded by the app at runtime. The user's position on the horizontal plane is used to identify a pixel in the height map image, and the colour value of that pixel determines the user's height, where black is low and white is high as shown in Figure 4.

Enriching the Environment

Graphical content alone is not enough to make a virtual world come alive. Audio and text were added to create a true multimedia experience. Audio is powered by OpenAL and comes from many sources in the environment: gospel readings in Latin can be heard coming from the cathedral; the sounds of hammer and anvil can be heard from workshops; and farm animals make sounds from their pens.

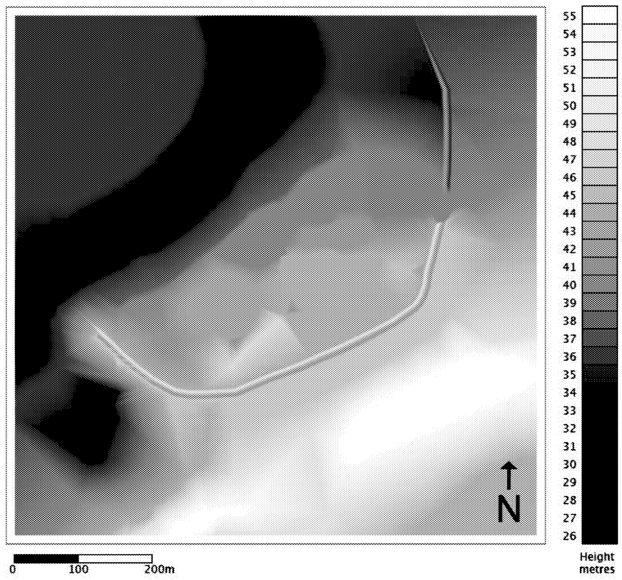

Figure 4: The height map which approximates the terrain elevations at Clonmacnoise, used to determine the user's vertical height at any position in the environment

Some objects in the world contain multimedia annotation. When these objects are tapped, the user has the option to display a multimedia document describing the object. These documents are written in HTML and are limited only by iOS's very advanced web browser engine (WebKit). These documents are loaded and displayed within the app and are stored offline so that network connectivity is not a prerequisite for enjoying the tour.

Optimising Performance

The final stage in iOS development was optimisation. Four main categories were identified as targets for performance enhancement:

1. Frame-rate The biggest optimisation challenge was frame-rate optimisation. This required creating LOD (Level of Detail) meshes for each object in the world, as previously. Furthermore, the far clip distance was set to an empirically determined value to skip rendering of far objects. Fog was added to make this clipping appear more natural.

2. Memory usage As the environment is quite large, original prototypes often crashed when the iPad/iPhone ran out of memory. By reducing the size of texture data, the amount of memory used was greatly reduced. We also decided to pre-process texture data into a format that can be loaded directly into video memory. By doing this the system memory does not need to be used to decompress and convert texture data before loading to video memory.

3. Load time Load time may take up to 20 seconds, and as the scale of the environment is not large enough to facilitate a lazy loading of resources, there were no realistic solutions to reducing this load time. By adding a loading screen, however, with real-time updates of percentage of content loaded, it is hoped that users will have more patience with the software.

4. File size The App Store only delivers apps under 20MB to devices connected via a mobile network. Larger apps require a Wi-Fi connection for deployment. With this in mind it was very important to keep the file size below 20MB. To do this support for the ARMv6 instruction set required by old iOS devices was removed, duplicate geometry was instanced to remove redundant mesh data, texture resolution was scaled down, image resources such as loading screens and maps were optimised, and the length of audio was limited.

CONCLUSIONS

Although testing has been limited to a small number of private beta testers so far, the overwhelmingly positive reaction thus far indicates that this type of smartphone heritage app provides a new way for people to engage with history, be it for tourist or educational purposes.

It does however require a lot more effort to put together than the existing digital book or augmented reality apps, and whether the users feel that this is merited in the enhanced experience waits to be seen.

From a technical standpoint, libraries such as OGRE significantly ease the development process but iOS devices still have comparatively limited hardware that demands a lot of work in order to deliver an enjoyable user experience, especially with such an ambitious project. However, with an intimate knowledge of iOS, the iOS SDK and supporting frameworks, and the C family of languages supported by iOS, it is possible to achieve impressive results.

FUTURE WORK

The app is currently set as a virtual ancient world explorer tool. Adding game play elements to the app was explored but it was felt that the scale and dimensions of the virtual environment was not conducive for an engaging game. For this reason, it has been decided to leave it as a virtual ancient world exploration tool. A separate 2-dimensional resource management game based on Clonmacnoise, is currently being created for Facebook which will eventually have links with the app and allow users to go from educational to gaming experience and visa-versa.

REFERENCES

Harbison, P. 2007. "*A guided tour of Ecclesiastical Treasures in Co. Clare.*" Dept of Environment, Heritage and Local Government

Murphy, D. 1998. "*Excavation of an early monastic enclosure at Clonmacnoise*". Clonmacnoise Studies Vol. 2, edited by King H.

O'Connor, K., and Manning, C. 1998. "*Clonmacnoise Castle*" Clonmacnoise Studies Vol. 2, edited by King, H.

MacGowan, K. 1985. "Clonmacnois" Kamac Publications.

Manning, C., 1998. "Clonmacnoise" OPW Publication.

WEB REFERENCES

[1] http://itunes.apple.com/us/app/clare-ecclesiastical-trails/id402159308?mt=8, eDymanics Limited, Clare Ecclesiastical Trails: iPhone application, 2011.
[2] http://itunes.apple.com/ie/app/dublin-city-walls/id388288868?mt=8, Noho Ltd, Dublin City Walls: iPhone application, 2010
[3] http://www.hauntedplanet.com, Haunted Planet
[4]http://www.imgtec.com/powervr/insider/sdk/KhronosOpenGLES2xSGX.asp, Khronos OpenGL ES 2.0 SDKs for PowerVR SGX, 2011.
[5] http://unity3d.com/, UNITY: Game Development Tool, 2011.
[6] http://www.ogre3d.org/, OGRE Open Source 3D Graphics Engine, 2011.
[7] http://irrlicht.sourceforge.net/, Irrlicht, 2011.
[8] http://www.ogremax.com/, OgreMax, 2011.

BIOGRAPHIES

GAVIN DUFFY is a geophysicist by training but became involved in 3D simulation in 2006 when he set up RealSim Limited. The digital reconstruction of Clonmacnoise began when RTE, the National Irish television broadcaster, commissioned their visualisation services for their 'Secrets of the Stones' documentary series. In 2009 RealSim teamed up with iPhone app specialist Daniel Heffernan, to work on the app presented here.

DANIEL HEFFERNAN is a software developer who has been developing apps for the iPhone since it first came out in 2007. He held the number 1 top paid position in the Irish App store continuously for 40 days in 2009 with his Eircom Wi-Fi password app "dessid". He is currently pursuing a research Master's degree at the University of Tokyo, Japan.

EOGHAN QUIGLEY is a 3D graphic designer who has over 10 years experience working in the gaming industry. In 2002 he was nominated for a BAFTA award for his work on children's animation Taz: Wanted. He joined RealSim in 2008 and has been the main 3D graphic designer on the Clonmacnoise project.

HEATHER KING was chief OPW archaeologist at Clonmacnoise for over 20 years. She has imparted all her knowledge on how she feels the Monastic City of Clonmacnoise would have looked, based on her excavations and research.

PAUL SMITH is a programmer and graphic designer joining RealSim in 2010. He is currently the lead programmer and artist working on a medieval resource management game. The game is styled on the Clonmacnoise environment and it is hope it will complement the iPhone app when it is launched later in 2011.

.

Open Device Control: Human Interface Device Framework for Video Games

Kosuke Kaneko[†], Yoshihiro Okada[††], Hiroyuki Matsuguma[†††]
[†, ††]Graduate School of Information Science and Electrical Engineering, Kyushu University
744 Motooka, Nishi-ku, Fukuoka, 819-0395 JAPAN
[†††]Graduate School of Design, Kyushu University
4-9-1 Shiobaru, Minamiku, Fukuoka, 815-0032 JAPAN
E-mail: [†]pumpkinkaneko@gmail.com, [††]okada@inf.kyushu-u.ac.jp, [†††]kuma@design.kyushu-u.ac.jp

ABSTRACT

This paper proposes a framework for human interface device for Video Games. Generally, a certain input device is designed to transmit its input data to its designated output devices. However, in some cases, it would be necessary to transmit the input data to other types of devices besides such designated output devices. For such situations, the proposed framework provides common protocols to exchange data among input and output devices based on Server-Client mechanism. Therefore, even when different developers individually make programs for each input and output devices, with the use of the framework, data communication among them will be possible without any modifications of the programs. To clarify the usefulness of the framework, this paper shows a serious game example, i.e., a rehabilitation game, developed using the framework.

KEYWORDS

Game Development Framework, Human Interface Devices, Serious Game, Rehabilitation

INTRODUCTION

Recently, various types of mobile devices like iPad, iPod and Android tablets besides smart phones have become available and popular. Since these devices have various functionalities, it is possible to use them in many cases instead of PC. Furthermore, these devices can be used as human interface devices (HID) such as a mouse device, game controllers like Wii Remote Controller because they have various types of sensors. This usage is technologically possible but practically not easy because originally these mobile devices are not considered to be used as such. For example, usually we cannot use iPod touch as a touch interface device of PC. Opposite cases have the same situation that usually we cannot use Wii Remote Controller as an input device of iPad. To do so practically, we have to create an interface program. If there is a very transparent interface program among such PC, mobile devices and human interface devices, we would be able to develop exciting and interesting applications more easily using such interface programs. In order to achieve this, we have been developing a HID framework, Open Device Control (OpenDC) as shown in Figure 1. OpenDC provides a dedicated framework for helping programmers to more easily develop application software like video games that support various types of human interface devices. In Fig. 1, input devices mean human interface devices like Wii Remote Controller and iPod touch that can capture user operation event data, and output devices mean display devices like PC, iPad and iPod on which any application software runs. The goal of our proposed framework is to provide transparent communication mechanisms among input and output devices as higher level programs. To do so, the standardization of communication protocols is significant.

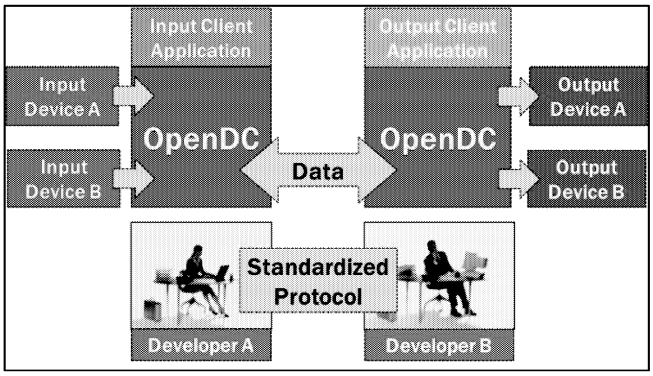

Figure 1: Framework of Open Device Control

In this paper, we propose such standardized communication protocols and several modules provided by our proposed framework to support the protocols. We also clarify the usefulness of the proposed framework by showing an actual application program that uses the framework.

First of all, we introduce our serious game project that motivates us to develop OpenDC. Serious game means one of the game genres. A serious game is a game designed for a primary purpose other than pure entertainment. This game genre includes educational games, training games, rehabilitation games, and so on. Last year, we tried to make one rehabilitation game as an achievement of our project to support rehabilitation training of actual patients. Rehabilitation training is a painful and tedious routine because most of the trainings are simple repetitive actions. Therefore, rehabilitation therapists in our project team wanted any method that enabled to make their patients do rehabilitation training with delight and joy. We thought a serious game might be one of the solutions for that and tried to make such a serious game. The purpose of the project team is to investigate how serious games are efficient for the rehabilitation of patients. We have a plan to develop various serious games for the rehabilitation training. Such serious games always need certain types of input devices for capturing patients' actions because in conventional rehabilitation training, in order to make a rehabilitant effectively move his/her body, a therapist uses some real devices like ring-toss and balance ball for their patient. Thus, we have to use various types of input devices to capture such a body action data as digital data in corresponding serious games. This motivated us to develop OpenDC.

The framework of OpenDC considers the following remarks.

1. OpenDC exchanges any digital data among input and output devices
2. OpenDC provides developers with a transparent framework for communications among input and output devices
3. OpenDC should support simultaneous multiple input and output
4. OpenDC should be suitable for Cloud Computing Environments

The remarks 1 and 2 are already mentioned above. About the remark 3, OpenDC can distinguish input devices and output devices completely so that it can accept multiple data simultaneously input from each input device and also can output them simultaneously to each output device. About remark 4, our proposed HID framework is available not only for video games but also for various applications controlling website, robot, IPTV and so on because we provide Server-Client based communication mechanisms using the Websocket Protocol, not Bluetooth Protocol, in our HID framework to exchange data between input devices and output devices. By achieving the above remarkable points, we can provide a HID framework very useful for software developers to make more exciting applications more easily.

The remainder of this paper is organized as follows: in the following sections, we explain the architecture of OpenDC and introduce actual example game that uses OpenDC framework. Before that, we introduce related work in the next section. Finally we conclude the paper in the last section.

RELATED WORK

In entertainment fields, there are various HIDs to make corresponding applications more exciting. Especially, Nintendo Wii Remote Controller is one of the most famous HIDs. Since the device has a 3-axis acceleration sensor, using it we can transmit our arm locomotion data into application software. Kinect, the motion capture device produced by Microsoft Inc., is one of the recent attractive HIDs. The device directly captures human actions using its two cameras and one depth sensor and applies them into the game world without holding any real device [Shotton, J.]. It is also possible to use smart phones as HIDs because at present smart phones like iPhone and Android phone have various sensors to obtain real-world data, e.g., an acceleration sensor, a gyroscope and so on. In this way, it will be possible to rapidly develop new HIDs with various sensors and to make them available in the near future.

OpenDC is a framework for an application using HID. There are some researches to use a specific HID like Phantom Device [Miyahara, K., 2005]. OpenDC can support such specific HID. OpenDC also supports multiple input devices under Cloud Computing Environment. So we can employ OpenDC for development of applications that use various HIDs remotely shared by multiple users [Hutchins, M.] [Miyahara, K., 2009].

We started this research from the result of the previous research [Takasugi, S.] about a serious game for rehabilitation training. As explained in previous section, serious game is a game designed for a primary purpose other than pure entertainment, i.e., education, healthcare and so on. Recently, some game researchers call such approaches that take game mechanics into non-game applications, "Gamification" [Deterding, S.] [McGonigal, J.]. Some research of serious games uses virtual reality or augmented reality technologies besides standard ICT technologies [Broeren, J.] [Burke, J.]. There are some researches about serious games for health care [Thompson, D.]. We are applying our serious game for rehabilitants. We are investigating how the game works effectively for rehabilitants. We have applied the game to 48 rehabilitants for 3 months and obtained several good results although the results will be reported in another paper.

ARCHITECTURE

In this section, we explain the architecture of OpenDC. First of all, we introduce the components of OpenDC. OpenDC consists of mainly three modules, i.e., Input Client Module, Output Client Module and Server Module as shown in Figure 2. Input and Output Client Module work on a client application. Server Module works as a server for exchanging data between input client and output client applications. This module also works with Input Client Module on an input device or with Output Client Module on an output device in cases that there is no server. Exchanges of data between Input Client Module/Output Client Module and Server Module are carried out by IP communications on the Internet. Indeed, we use Websocket Protocol in OpenDC for the communications.

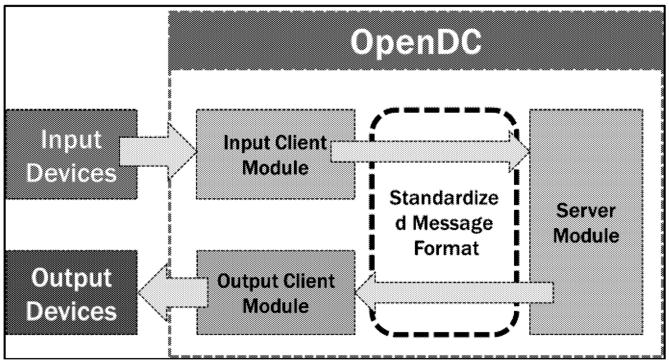

Figure 2: Overview of OpenDC architecture

Websocket Protocol and JSON Message Format

Websocket is one of the communication standards for computer networks, and its API and Protocol are standardized by W3C and IETF respectively. Websocket supports bidirectional, full-duplex TCP communications. This communication standard is originally designed for communications between a web client and a web server. But we can use it for any server-client applications besides web services. The reason why we employ Websocket Protocol for our OpenDC is that it is suitable for Cloud Computing Environments.

We employ JSON (JavaScript Object Notation) format for exchange data between Input and Output Client Module in OpenDC. JSON is compliant with ECMA-262, revision 3 and extensible like XML (Extensible Markup Language). JSON represents data as a set of several pairs of attribute name and its value. The reason why we select JSON format among extensible data formats is that JSON is simple and easy to handle data in its format.

The message format based on JSON has several elements. IP address is used to identify one of the computer devices. Application Name is used to identify one of the applications simultaneously run on a common computer device identified by its IP address. Device ID is used to identify one of the input/output devices simultaneously connected to the common computer device that works in the application identified by its Application Name. OpenDC assigns any unique number to each device as its device ID. Device Name is used to identify one of input/output devices by specific names. This is an optional element. Other elements are input data sent from the corresponding input device.

```
{
  “IP Address” : “133.5.X.XX”,
  “Application Name” : “Example ”,
  “Device ID” : 5,
  “Device Name” : “iPad”,
  “Data” :
    {
      “Acceleration Sensor” :
      {
        “X” : 0.2, “Y” : 0.3, “Z” : 0.01
      }
    }
}
```

Message format used by communications in OpenDC

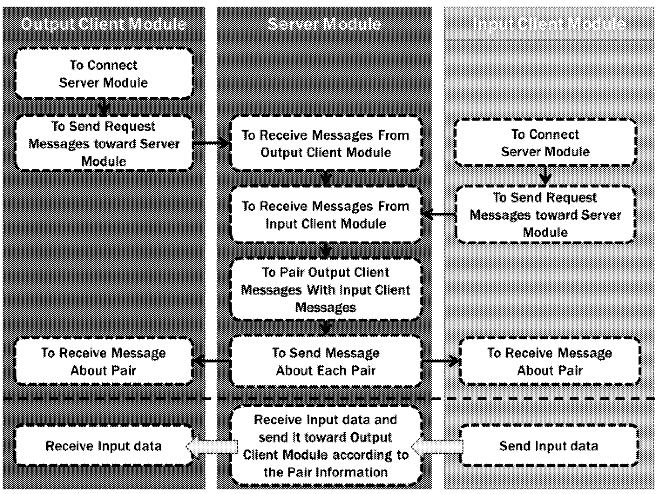

Figure 3: Communications among Input/Output Client Modules and Server Module

Output Client Module

Output Client Module runs on an output device on which generally a certain client application runs. Output Client Module communicates with Server Module according to the several steps shown in Figure 3. When Output Client Module connects to Server Module, the Output Client Module sends a request massage to Server Module. The message includes the information about data types which mean what kinds of input data the Output Client Module can accept. The massage consists of sets of data types with logical add (OR) or logical product (AND), e.g., *“ACCELERATION SENSOR | GYROSCOPE”*. Such a representation of data types is useful in cases that an input device can support various types of input data. In OpenDC, when Output Client Module receives assigned types of data, Output Client Module calls each event handlers corresponding to the data types. If an input device supports two types, Output Client Module calls both event handlers. If an output device needs only one of Acceleration sensor data or Gyroscope data and an input device supports both types of data, the representation for that should be *“ACCELERATION SENSOR | GYROSCOPE”* using a logical add. In this case *“ACCELERATION SENSOR”* has a higher priority rather than “GYROSCOPE”. Then, Output Client Module calls only one corresponding event handler. If an output device accepts two types of data simultaneously in the application run on it, the representation for that should be *“ACCELERATION SENSOR & GYROSCOPE”* using a logical product. Output Client Module sends the message about data types with the information about their minimum and maximum values which Output Client Module can accept. These values are necessary for Server Module to adjust input data sent from Input Client Module and send the appropriate data to Output Client Module. This enables developers to individually develop applications run on an output device from those on an input device. This feature of OpenDC provides transparent development framework among input and output devices.

Input Client Module

Input Client Module runs on an input device on which generally a certain client application runs. Input Client Module communicates with Server Modules according to the several steps shown in Figure 3. When Input Client Module connects to Server Module, the Input Client Module sends a request massage to the Server Module. As well as Output Client Module, the message includes the information about data types which mean what kinds of input data the Input Client Module can send. The massage representation is similar to that of Output Client Module. The request message is significant for making matching between data sent from input devices and data requested by output devices to reduce the overhead about data transmission between Input Client Module and Server Module and between Output Client Module and Server Module.

Server Module

Server Module works as a server for exchanging data between input and output client applications. This module also works with Input Client Module on an input device or with Output Client Module on an output device in cases that there is no server. As shown in Figure 3, Server Module mainly plays a role to make paring requests which Input and Output Client Modules send to. Server Module registers the request pairs into a database. We make Server Module using Node.js. Node.js is JavaScript programming framework dedicated for web servers. The framework runs on Google V8 JavaScript Engine. The reason why we select Node.js for server side programs is that using the framework, it is easy to build servers of client-server applications besides web servers those are very light-weight and those can treat large data simultaneously requested from many clients.

EXAMPLE VIDEO GAME

In this section, we introduce a video game using OpenDC as shown in Figure 4. We have already developed the video game for rehabilitation training as one of the achievements of our serious game project. There are various patients having a different symptom in rehabilitation institution. Their rehabilitation trainings are also different styles. In the trainings, we choose “Standing-up Training”, one of the basic trainings for all rehabilitant as assigned “grade A” training in Japanese Rehabilitation Guideline [Japanese Stroke Therapy Guideline Committee], i.e., repeats of one cycle that are sitting on a chair and standing up slowly. Ideally, a rehabilitant should perform the cycle in 4 seconds according to Japanese Rehabilitation Guideline. One set of the training includes 40 cycles. A rehabilitant has to perform 3 to 5 sets of the training every day. As a result, a rehabilitant takes this simple repetitive action 100-200 times every day. This is a very tedious training. Thus, we have developed the serious game for this training. Although it is very difficult to define roles which effectively motivate rehabilitants about the training because there are various symptom levels of rehabilitants, complex roles are not suitable. We decided to employ a simple role of the game in that a rehabilitant grows a tree represented as CG on a screen by his/her stand-up and sit-down actions. The tree is shown in the third picture in Fig. 4. In this video game, rehabilitants can receive some rewards, i.e., medals or cards according to the height of their tree. This is important to enhance patient’s motivation. The goal of the game is to grow the tree until its height specified by a therapist.

The game system obtains rehabilitant’ motion data of his/her standing-up and sitting-down using Wii Fit which is a balance board produced by Nintendo Inc. We originally designed the game to support only Wii Fit as its input device. Because of the use of OpenDC, we can employ other devices like iPhone and Android mobile phone as an input device of the game without any modifications of the game programs. In this case, we can use an acceleration sensor of iPhone for capturing the standing-up and sitting-down motion data. The output client application, i.e., the serious game just receives input data sent from Server Module that negotiates with Input Client Module on a server. We can also use multiple input devices because OpenDC can distinguish each of these devices and handle their input data individually. For example, we can play the serious game using Wii Fit and iPhone simultaneously as input devices. Similarly, we can also use multiple output devices. In this case, we can check input data on a certain output device during the play of the game runs on another device.

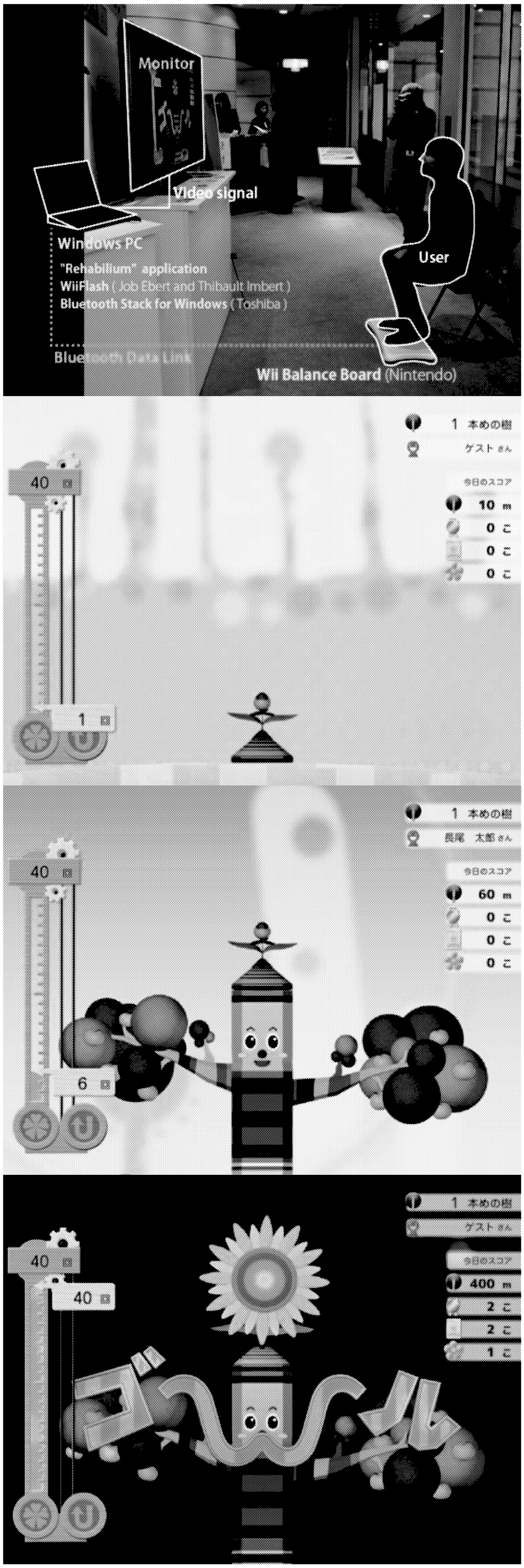

Figure 4: A video game example using OpenDC

This functionality is very significant for the case that therapist wants to check input data of the game using any portable device separately from the game. This is one of the remarkable features of OpenDC.

Results of actual training performed by rehabilitants using the serious game introduced in this paper indicate higher efficiency for the rehabilitation. We will report about it in other conferences of more suitable topics in the near future.

CONCLUSION AND FUTURE WORK

In this paper, we proposed OpenDC and introduced its architecture. Furthermore, we showed one actual rehabilitation game developed using OpenDC. We have been developing OpenDC to provide transparent programming framework that allows developers to develop various types of applications including video games those support various types of human interface devices. The features of OpenDC provide the standardized communication protocol for enabling human interface devices to exchange their data each other derived from the use of Websocket IP communication, JSON message format and Node.js.

OpenDC can be used for various types of applications besides video games, e.g., websites, digital signage applications, IPTV, remote control for robots and so on. We will develop various applications using OpenDC to clarify its usefulness as one of our future works.

ACKNOWLEDGEMENT

In this serious game project, many members of various fields have supported us. Especially, we wish to thank rehabilitation therapists, Jiro Kajiwara and Kenta Hayashida, in Nagao Hospital for their help with our experiments.

REFERENCES

Shotton, J. and Fitzgibbon A., Cook, M., Sharp, T. Finocchio, M., Moore, R., Kipman, A. and Blake, A., 2011. “Real-Time Human Pose Recognition in Parts from Single Depth Images”.

Miyahara, K., Okada, Y. and Nijima, K. 2006, “A Cloth Design System Using Haptic Device and Its Collaborative Environment”, CD-ROM Proc. Of the HAVE’ 2006-IEEE International Workshop on Haptic Audio Visual Environments and their Applications, IEEE ISBN: 1-4244-0761-3, 48-53.

Miyahara, K. and Okada Y. 2009, “RoCoS: room-based communication system and its aspect as development tool for 3D entertainment applications”, ACE ’09 Proceedings of the International conference on Advances in Computer Entertainment Technology, 3-10.

Hutchins, M., Stevenson, D., Gunn, C., Krumpholz, A., Adriaansen, T., Pyman, B. and O’Leary, S., 2005, “Communication in a networked haptic virtual environment for temporal bone surgery training”, Virtual Reality (Springer), Vol9, No.2-3, 97-101.

Takasugi, S., Suzuki, M., Kawamura, Y., Nejime, Y., Kawano, I., Kawashima, T., Matsumoto, K. and Iwamoto, Y., 2004, “12-month Intervention of Playing Arcade Games in the Elderly Woman”, International Symposium on Preventing Falls and Fractures in Older Person (2004).

Japanese Stroke Therapy Guideline Committee, 2009, Stroke Therapy Guideline, 300-304.

Broeren, J.B, Bjorkdahl, A., Claesson, L., Goude, D., Lundgren-Nilsson, A., Samuelsson, H., Blomstrand, C., Sunnerhagen, K. and Rydmark, M., 2008 “Virtual Rehabilitation after stroke”, Studies in Health Technology and Informatics, Vol. 136, 77-82.

Thompson, D., Baranowski, T., Buday, R., Baranowski, J., Thompson, V., Jago, R. and Griffith, M., 2010, “Serious Video Games for Health: How Behavioral Science Guided the Development of a Serious Video Game”, Simulation and Gaming, Vol. 41, Issue 4, 587-606.

Burke, J., McNeill, M., Charles, D., Morrow, P., Crosbie, J. and McDonough, S., 2010, “Augmented Reality Games for Upper-Limb Stroke Rehabilitation”, Proceedings of Second International Conference on Games and Virtual Worlds for Serious Applications (VS-GAMES), 75-78.

Deterding, S., Sicart, M., Nacke, L., O'Hara, K. and Dixon, D., 2011, “Gamification: Using Game Design Elements in Non-Gaming Contexts”, Proceedings of the 2011 Annual Conference Extended Abstracts on Human Factors in Computing Systems (CHI EA'11), Vancouver, Canada, 2425-2428.

McGonigal, J., 2011, “Reality Is Broken: Why Games make Us Better and How They Can Change the World.”, The Penguin Press.

BIOGRAPHIES

Kosuke Kaneko is a doctor course student in Graduate School of Information Science and Electrical Engineering, Kyushu University, Japan. He received his B.A. degree in Science and M.A. degree in Information Science from Kyushu University in 2006 and 2008, respectively. His main research interests include 3D Computer Graphics, Game Engine Architecture and Serious Games.

Yoshihiro Okada is an Associate Professor in Graduate School of Information Science and Electrical Engineering, Kyushu University, Japan. He received his Bachelor, Master degree and Doctorate in Engineering from Hokkaido University, Japan in 1988, 1990 and 1993, respectively. Current his research interests include 3D multimedia, Information Visualization, Human-Computer Interaction and Interfaces, also Virtual Reality applications.

Hiroyuki Matsuguma is a Lecturer of Graduate School of Design, Kyushu University, Japan. He received his B.A. and M.A. degree in Design from Kyushu Institute of Design, Fukuoka, Japan in 1994 and 1996, respectively. His main research interests include Computer Graphics Design, Computer Animation and Serious Games.

AUTHOR LISTING

AUTHOR LISTING