3rd INTERNATIONAL NORTH-AMERICAN CONFERENCE

ON INTELLIGENT GAMES AND SIMULATION

GAMEON-NA 2007

EDITED BY

Paul Fishwick

And

Benjamin Lok

SEPTEMBER 10-12, 2007

UNIVERSITY OF FLORIDA
GAINESVILLE, USA

A Publication of EUROSIS-ETI

Printed in Ghent, Belgium

3RD International North-American Conference

on

Intelligent Games and Simulation

GAINESVILLE, USA

SEPTEMBER 10-12, 2007

Organized by

ETI

Sponsored by

EUROSIS

Co-Sponsored by

Ghent University

GR@M

UBISOFT

Larian Studios

GAME-PIPE

The MOVES Institute

Hosted by

The University of Florida

Gainesville, USA

EXECUTIVE EDITOR

PHILIPPE GERIL
(BELGIUM)

EDITORS

Conference Chairs

Paul Fishwick
University of Florida
Gainesville, USA

Benjamin Lok
University of Florida
Gainesville, USA

PROGRAMME COMMITTEE

Game Development Methodology

Track Chair: Licinio Roque, University of Coimbra, Coimbra, Portugal
Joaquim Ramos de Carvalho, University of Coimbra, Portugal
Óscar Mealha, University of Aveiro, Portugal
Eija Karsten, University of Turku, Finland
Jari Multisilta, University of Tampere, Finland
Esteban Clua, Universidade Federal Fluminense, Brasil

Physics and Simulation

Graphics Simulation and Techniques

Magy Self El-Nasr, Penn State University, University Park, USA
Pieter Jorissen, Universiteit Hasselt, Diepenbeek, Belgium
Ian Marshall, Coventry University, Coventry, United Kingdom
Marco Roccetti, University of Bologna, Bologna, Italy

Facial, Avatar, NPC, 3D in Game Animation

Marco Gillies, University College London, London, United Kingdom
Yoshihiro Okada, Kyushu University, Kasuga, Fukuoka, Japan
Paolo Remagnino, Kingston University, Kingston Upon Thames, United Kingdom
Marcos Rodrigues, Sheffield Hallam University, Sheffield, United Kingdom
Joao Manuel Tavares, FEUP, Porto, Portugal

Rendering Techniques

Sushil Bhakar, Concordia University, Montreal, Canada
Joern Loviscach, Hochschule Bremen, Bremen, Germany
Frank Puig, University of Informatics Sciences, Havana, Cuba

PROGRAMME COMMITTEE

Artificial Intelligence

Artificial Intelligence and Simulation Tools for Game Design

Stephane Assadourian, UBISOFT, Montreal, Canada
Michael Buro, University of Alberta, Edmonton, Canada
Penny de Byl, University of Southern Queensland, Toowoomba, Australia
Abdennour El-Rhalibi, Liverpool John Moores University, Liverpool, United Kingdom
Antonio J. Fernandez, Universidad de Malaga, Malaga, Spain
Tshilidzi Marwala, University of Witwatersrand, Johannesburg, South-Africa
Gregory Paull, The MOVES Institute, Naval Postgraduate School, Monterey, USA
Oryal Tanir, Bell Canada, Montreal, Canada
Christian Thurau, Universitaet Bielefeld, Bielefeld, Germany

Learning & Adaptation

Christian Bauckage, Deutsche Telekom, Berlin, 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
Pascal Estrallier, Universite de La Rochelle, La Rochelle, France
Maja Pivec, FH JOANNEUM, University of Applied Sciences, Graz, Austria
Martina Wilson, The Open University, Milton Keynes, United Kingdom

Intelligent/Knowledgeable Agents

Nick Hawes, University of Birmingham, United Kingdom
Scott Neal Reilly, Charles River Analytics, Cambridge, USA
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
Nicholas Graham, Queen's University, Kingston, Canada

Opponent Modelling

Ingo Steinhauser, Binary Illusions, Braunschweig, Germany

Peripheral

Voice Interaction

Oliver Lemon, Edinburgh University, Edinburgh, United Kingdom
Bill Swartout, USC, Marina del Rey, USA

Artistic input to game and character design

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

Storytelling and Natural Language Processing

Jenny Brusk, Gotland University College, Gotland, Sweden
Terry Harpold, University of Florida, Gainesville, USA
Laurie Taylor, University of Florida, Gainesville, USA
Ruck Thawonmas, Ritsumeikan University, Kusatsu, Shiga, Japan
R. Michael Young, Liquid Narrative Group, North Carolina State University, Raleigh, USA
Clark Verbrugge, McGill University, Montreal, Canada

PROGRAMME COMMITTEE

Online Gaming and Security Issues in Online Gaming

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

MMOG's

Michael J. Katchabaw, The University of Western Ontario, London, Canada
Alice Leung, BBN Technologies, Cambridge, USA
Yusuf Pisan, University of Technology, Sydney, Australia
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 Janiero, Brazil
Richard Ferdig, University of Florida, Gainesville, USA
Erol Gelenbe, Imperial College London, United Kingdom
Henry Lowood, Stanford University Libraries, Stanford, USA
Tony Manninen, University of Oulu, Oulu, Finland
Jaap van den Herik, University of Maastricht, Maastricht, The Netherlands

Games for training

Ahmed BinSubaih, University of Sheffield, Sheffield, United Kingdom
Michael J. Katchabaw, The University of Western Ontario, London, Canada
Jens Müller-Iden, Universität Münster, Münster, 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

Mobile Gaming

Stefano Cacciaguera, University of Bologna, Bologna, Italy
Sebastian Matyas, Otto-Friedrich-Universität Bamberg, Bamberg, Germany

Perceptual User Interfaces for Games

Tony Brooks, Aalborg University Esbjerg, Esbjerg, Norway
Michael Haller, Upper Austria University of Applied Sciences, Hagenberg, Austria
Carsten Magerkurth, AMBIENTE, Darmstadt, Germany
Lachlan M. MacKinnon, University of Abertay, Dundee, United Kingdom

GAME'ON-NA
2007

All articles published in these Proceedings have been peer reviewed

EUROSIS-ETI Publications are ISI-Thomson and INSPEC referenced

For permission to publish a complete paper write EUROSIS, c/o Philippe Geril, ETI Executive Director, Ghent University, Faculty of Engineering, Dept. of Industrial Management, Technologiepark 903, Campus Ardoyen, B-9052 Ghent-Zwijnaarde, 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
Final Cover Design by Grafisch Bedrijf Lammaing, Ostend, Belgium

EUROSIS-ETI Publication

ISBN: 978-90-77381-35-9
EAN: 978-90-77381-35-9

Preface

Welcome to Game-On 'NA 2007, the third North American sister event of the well-established European Game-On conference series on AI and simulation in computer games. The University of Florida in Gainesville is the setting for this year's event and as one of the driving forces behind simulation and research the ideal venue for hosting this topic related event.

Just like previous years game AI and content-generation constitutes the main focus of the event with simulated board games coming in as the second most important factor in game development.

As well as the peer-reviewed papers, Game-On 'NA 2007 features a keynote talk by Jim Brazell, Consulting Analyst, Digital Media Collaboratory, University of Texas, Austin entitled: "The Future is Here: Video Games, Virtual Worlds and Mixed Reality".The second invited presentation is by Roger Smith Ph.D. Chief Technology Officer U.S. Army Program Executive Office for Simulation, Training and Instrumentation entitled: "Taking Game Technology Seriously" Last but not least there is also Tutorial on Mixed Reality by Charles Hughes, University of Central Florida

Game-On 'NA 2007 is of course, also about making contacts in the computer game research community. Several social events are planned, including, a conference dinner and tours of the Digital Worlds Institute - REVE and Artificial Studios. This tour will include, a tour of the office to all who'd like to see it, and a demonstration of the company's various products (which include "Reality Engine", "CellFactor", "Monster Madness" We hope you find your time at this Game-On 'NA productive and enjoyable while also enjoying the Florida hospitality.

Paul Fishwick and Benjamin Lok
Conference Chairs
University of Florida
Gainesville, USA

CONTENTS

CONTENTS

SCIENTIFIC PROGRAMME

KEYNOTE

Games-Nano-Bio-Info-Cogno: How are video games connected to 21st century science and learning?

Jim Brazell,
Consulting Analyst,
Digital Media Collaboratory,
University of Texas at Austin
jim@ventureramp.com

Keywords: Video Games, Virtual Worlds, Serious Games, 21st Century Science, Convergence, Transdisciplinary.

Introduction:

> "… not only is change a constant, but the pace of change is accelerating. It's growing exponentially. And it's about doubling every decade. And so in the next 25 years we'll see 100 years of progress at to-day's rate of progress… It's not just a matter of dealing with one revolution. We have many intersecting revolutions in biology, information science, materials science with nanotechnology and so on. We're going to have to deal with this panoply of intersecting, accelerating trends." *--Kurzweil, Voices of Innovation, American Association of Engineering Societies*

Today, we are witnessing the consilience ("jumping together," Wilson) of natural and physical sciences. This jumping together of scientific domains, evident in the fusion of nanoscience, bioscience, information science and cognitive science is expressed in the literature as "nano-bio-info-cogno" and "convergence." The trend of consilience resurfaces the requirement for transdisciplinarity.

> …transdisciplinarity concerns that which is at once between the dis-ciplines, across the different dis-ciplines, and beyond all discipline. Its goal is the understanding of the present world, of which one of the imperatives is unity of knowledge**.** *--Nicolescu, 2003*

> Transdisciplinarity implies solving real world problems that do not have prescribed answers in such a way as to unite learning, R&D and innovation into one act. *--Brazell, 2004*

Workers with transdisciplinary skills are needed in government, military, industry, and academia (World Technology Evaluation Center; Turpin, 2000; Stanford University, 2002; Arts and Humanities Research Board; Daly, Farley, Thomson, 2001; MST News, 2003; World Technology Evaluation Center; Office of Scientific and Technical Information, 2002; TANSEI, 2002; De Marca, Gelman; Carty, 1998; Nanotechnology Research Institute). To meet the needs and challenges of the 21st century, science, industry and private sector leaders are calling for a qualitative evolution of learning systems:

"Half a millennium ago, Renaissance leaders were masters of several fields simultaneously. Today, however, specialization has splintered the arts and

engineering, and no one can master more than a tiny fragment of human creativity. The sciences have reached a watershed at which they must combine if they are to continue to advance rapidly. Convergence of the sciences can initiate a new renaissance embodying a holistic view of technology based on transformative tools, the mathematics of complex systems, and unified cause-and-effect under-standing of the physical world from the nanoscale to the planetary scale.

> "Educational institutions at all levels should undertake major curricular and organizational reforms to restructure the teaching and research of science and engineering so that previously separate disciplines can converge around common principles to train the technical labor force for the future.
>
> "Manufacturing, biotechnology, information and medical service corporations will need to develop partnerships of unparalleled scope to exploit the tremendous opportunities from technological convergence, investing in production facilities based on entirely new principles and materials, devices and systems, with increased emphasis on human development." --*World Technology Evaluation Center, 2002*

Transdisciplinarity may seem like Ivory Tower language, however, transdisciplinaritry implies beyond the disciplines—a concept that is beyond many specialized academics. Where can we find transdisciplinary actors today?

You may be surprised with the answers—virtual worlds, serious games and network video games. The mod'ers and builders affiliated with these communities of practice are transdisciplinary actors. Mod'ers fuse artistic, scientific and engineering techniques. Mod'ers often straddle learning and commercialization--a key aspect of transdisciplinarity ("Counter Strike" and the origins of "Space War"). To sum the connection between gaming and 21st Century Science: both endeavors require whole brain, adaptive leaders able to cope and even flourish in uncertain, complex situations requiring effective human collaboration in network environments.

Educational programs, economic development initiatives and workforce programs should recognize the relationship between popular network game youth culture, the needs of 21st Century Science and the pressing requirement for new engineering and science-related teachers, executives and R&D professionals. Gamers represent a bridge to the natural learning systems and creativity needed to cope with the increased velocity of new scientific and technical knowledge.

Links between 21st Century Science, gaming and learning are emerging. In medical science Cai, Snel, Bharathi, Klein, and Klein-Seetharaman have developed BIOSIM, a network learning game and Problem Solving Environment (PSE). The game uses data from the human genome, a game engine, and four and five-year old children to teach biology and simultaneously to identify the chromosome responsible for fatal meningitis. The BIOSIM Problem Solving

Environment contains three interaction modes: role-play, voyage, and networked problem solving meningitis (Cai, Snel,

Bharathai, Klein, Klein-Seetharaman, 2003).

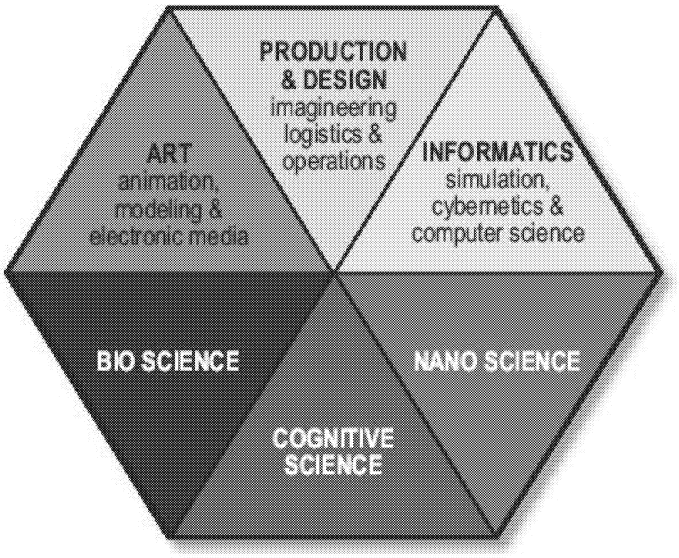
This cross appropriation of innovation from one domain to another through social construction of knowledge is an example of transdisciplinarity. BIOSIM illustrates cross appropriation among K-12 education, network games, network learning and bio-informatics domains.

BioSIM 1.0

Source: Cai, Snel, Bharathai, Klein, Klein-Seetharaman, 2003

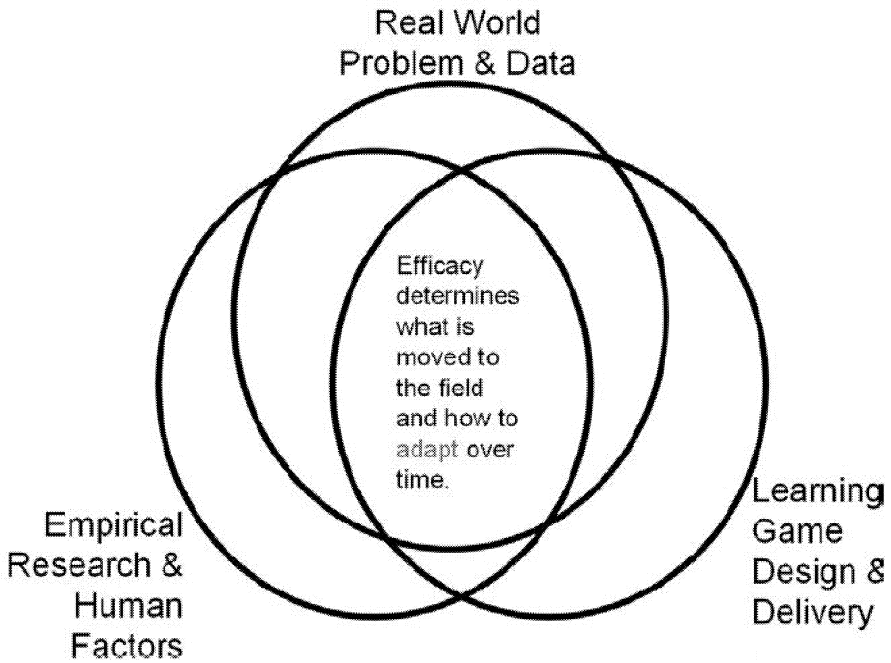
The knowledge domains in the BIOSIM example illustrate unity of systems--transdisciplinarity. The functional domains required to create BIOSIM are nearly identical to network games, with the exception of increased emphasis on cognitive science and subject matter expertise in the domain of inquiry (genetics).

nano-bio-info-cogno-games knowledge domains



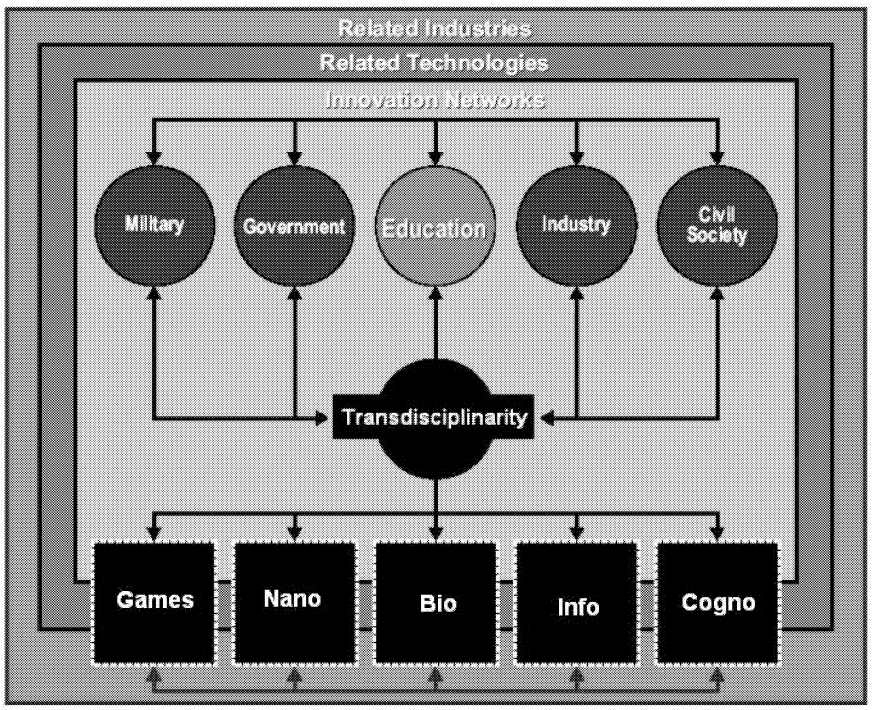
Network learning games such as BIOSIM are an "indicator species" illustrating changes in the educational learning ecology. These learning systems fuse real world data, learning design and empirical research to unify platforms for learning and doing. The major barrier to diffusion of new learning systems is empirical research that demystifies the efficacy of these models.

Unified Learning and Operational Environments and Tools



Games-Nano-Bio-Info-Cogno curri-cula and practicum should mirror self-organizing communities of practice, learning through social construction of technological sys-tems, critical thinking, collaboration in network environments, and distributed cognition. Theoretical underpinnings should be taught through applied learning environ-ments and tools such as telescience, whereby experimentation drives inquiry and innovation. Studies in ethics, social consequences of technology and philosophy should be emphasized. A transdisciplinary path of unified learning, IP creation and commercialization should also be explored with special attention to cultural, political and legal values and systems.

Games-Nano-Bio-Info-Cogno



K-12 schools, colleges and universities should consider forming trans-disciplinary learning models with inter organizational networks supporting specific historical and emerging industry, market and technology clusters relevant in each locality and as a component of a larger holistic network of regional and global economic development activities.

Swarm learning through semi-auto-nomous networks should be used to shift our focus from replacing teachers with technology to human mentoring and peering through online tools and environments; from teaching to learning, from assembly line systems and organizations to self-synchronizing, network structures; from theory or practice to theory and practice; from

proprietary learning systems to open learning systems; from eLearning to the whole learning ecology; from proprietary teaching architectures to open, egalitarian learning systems.

Education systems with a focus on state-of-the-art and emerging technologies and cultural practices should support the fusion of games-nano-bio-info-cogno with the following goals:

- Establish a new, broader learning systems model through transdisciplinarity.
- Network academics with industry and government needs.
- Fuse learning, R&D and commercialization.
- Unite disciplines through situated learning, problem-based inquiry and collaborative discovery.
- Explore the value choices made possible by modern science and technology.
- Pursue local, regional, statewide, national, and global network linkages.
- Facilitate the diffusion of new technologies, new knowledge, and new processes that work based on empirical validation.

To learn more about Games-Nano-Bio-Info-Cogno, please download the full report Gaming: A Technology Forecast – Implications for Texas Community and Technical Colleges on the web at:

http://system.tstc.edu/forecasting/reports/dgames.asp

Bibliography

Adams, Ernest W. "Getting into the Game: Skills and Careers in Game Development." *2002 Game Developer's Game Career Guide.* Fall 2002. Retrieved 17 July 2003 from http://www.designersnotebook.com/Getting_Into_The_Game.pdf.

Arts and Humanities Research Board. "The Case for Arts and Humanities Research in EU Framework Programme VI." Retrieved 3 August 2003 from http://www.ahrb.ac.uk/images/4_91372.pdf.

Bailey, Terryll. "Skill Standards for Electronic Game Content Production." Lake Washington Technical College. 2003. Retrieved January 22, 2004 from http://www.wa-skills.com/pdfs/egame_production/lwtc_standards_combined.pdf.

Bigcharts.com. August 2003. Retrieved 11 August 2003 from http://www.bigcharts.com.

Bloom, Adam, Email interview, Siebel Systems Inc., 13 August 2003. Brazell, Jim, and Doug Monroe. "Innovation Networks for Sustainable Workforce Development, Pattern Language for Living Communication," *Directions in Advanced Computing (DIAC)*. Computer Professionals for Social Responsibility, June 2003. Retrieved 4 August 2003 from http://diac.cpsr.org/cgi-bin/diac02/pattern.cgi/public?pattern_id=355.

Brown, S. "The Social Life of Information in the Digital Age and Kids that Grow Up Digital, Connecting with the Wired Generation:Young People, Digital Technology and the Media." Univ. of California Berkeley, 28 March 2003. Retrieved 4 August 2003 from http://journalism.berkeley.edu/events/conference2003/present/brown.pdf.

Bureau of Labor Statistics, U.S. Department of Labor, *Occupational Outlook Handbook, 2002-03 Edition,* "Systems Analysts, Computer Scientists, and Database Administrators." Retrieved 31 July 2003 from http://www.bls.gov/oco/ocos042.htm.

By the Numbers: U.S. Markets 1-20. *RCR Wireless News*. 18 November 2002. Retrieved 4 August 2003 from http://www.rcrnews.com/files/2002%20us%20markets.pdf.

Cai, Y., I. Snel, B. Bharathai, C. Klein, and J. Klein-Seetharaman. "Towards Biomedical Problem Solving in a Game Environment." 2003., Retrieved 4 August 2003 from http://www.andrew.cmu.edu/~ycai/biogame.pdf.

Career connections level designer: Agoodnet, Inc. 2003. Retrieved July 29, 2003 from http://careers.awn.com/jobdisplay.php3?job_no=6162.

Carty, Dr. A.J. "Future Opportunities for Science and Technology Leadership in Ottawa." National Research Council Canada, 15 April 1998. Retrieved 3 August 2003 from http://www.nrc-cnrc.gc.ca/newsroom/speeches/future98_e.html.

Chatham, M. "Training Superiority: DARPA Universal Persistent On-Demand Training Wars." DARPA Defense Sciences Office, August 2002. Retrieved 4 August 2003 from http://www.darpa.mil/dso/future/darwars2/darwars_1_files/frame.htm.

Chen, Jeanine, Ravi Patel, and Robert Schaefer. "Information Technology Sector Report." Rice University, 6 February 2002. Retrieved 29 July 2003 from http://www.ruf.rice.edu/~admn543/IT/IT_sector_report.pdf.

Church, D., J. Della Rocca, R. Hunicke, W. Spector, and E. Zimmerman. "The Study of Games and Game Development." IGDA Education Committee, 25 February 2003. Retrieved 4 August 2003 from http://www.igda.org/academia/curriculum_framework.php

Cohen, M. "Games Tester/Quality Assurance." *Career Connections*. 2003. Retrieved 17 July 2003 from http://careers.awn.com/jobdisplay.php3?job_no=5477.

Comstock, A. Interviews by phone and email. Electronic Game Developers Society, 17 August 2003.

Consumer Electronics Association. *5 Technologies to Watch*. October 2002. Retrieved August 3, 2003 from http://www.ce.org/publications/books_references/CEA_5TechBk.pdf.

Courtie, A., and M. M. Walker. "Current hiring trends." *2002 Game Developer's Game Career Guide*. Fall 2002.

Cowhey, P. *New Challenges and New Challenges and Opportunities for the Global Telecommunications and Information Industries*. December 2002. Retrieved 11 August 2003 from http://brie.berkeley.edu/~briewww/pubs/pubs/IGE%20Seoul.pdf.

Crosby, O. "Working so others can play: Jobs in video game development." *Occupational Outlook Quarterly,* Summer, 2000.

Csikszentmihalyi, Mihaly. *Flow: The Psychology of Optimal Experience.* New York: Harper & Row, 1990.

Daly, Herman, and Josh Farley. *Ecological Economics: Principles and Applications.* Retrieved 3 August 2003 from http://www.uvm.edu/~jfarley/237/chap1.pdf.

De Marca, J. Roberto B., and Alex Gelman. "Message from the President: Globalization of Industry and Academia." IEEE Communications Society, July 2001. Retrieved 3 August 2003 from http://www.comsoc.org/ci/public/2001/jul/cipresmess.html.

Denning, P. J., and R. M Metcalfe. *Beyond Calculation: The Next Fifty Years of Computing*. New York: Springer-Verlag, 1997.

Duca, John V., and Mine K. Yücel. "Biotech in Texas." Based on a presentation by Federal Reserve Bank of Dallas. July 2002. Retrieved 11 August 2003 from http://www.dallasfed.org/htm/research/pdfs/bd0702.pdf

Entertainment and Media Outlook: 2003-2007. PricewaterhouseCoopers, n.d.. Retrieved 3 August 2003 from http://www.pwc.com/Extweb/industry.nsf/0/8CF0A9E084894A5A85256CE8006E19ED?opendocument&vendor=none.

Entertainment Software Association, formerly Interactive Digital Software Association. "Computer and Video Game Software Sales Grew to a Record-Breaking $6.9 Billion in 2002." 7 January 2003. Retrieved 4 August 2003 from http://www.idsa.com/1_27_2003.html

"ESA Anti-Piracy Information." Entertainment Software Association, n.d. Retrieved 3 August 2003 from http://www.theesa.com/piracy.html

"Essential Facts About the Computer and Video Game Industry: 2003 Sales, Demographics and Usage Data." Entertainment Software Association, formerly Interactive Digital Software Association, 2003. Retrieved 3 August 2003 from http://www.idsa.com/EF2003.pdf.

François, Charles. "Systemics and Cybernetics in a Historical Perspective." *Systems Research and Behavioral Science* 16 (1999): 203-219. Retrieved 11 August 2003 from http://www.uni-klu.ac.at/~gossimit/ifsr/francois/papers/systemics_and_cybernetics_in_a_historical_perspective.pdf.

"Frequently asked questions about the game industry." N.d. Retrieved 26 July 2003 from http://www.diterlizzi.com/support/about_game_industry.shtml

"Fusion." Australian National University. N.d. Retrieved 3 August 2003 from http://www.anu.edu.au/culture/n_activities/fusion/home.htm.

"Games sales and marketing jobs." N.d. Retrieved 27 July 2003 from http://www.datascope.co.uk/jobs_games_sales_marketing.html

Geiger, Kevin. “Across the Great Divide: Making the Leap from 2D to 3D.” June 2001. Retrieved 29 July 2003 from http://www.simplisticpictures.com/learn3d.html.

Ghanbari, R. Email interview, Yahoo, Inc., 11 August 2003.

Godfrey, S. “Motion Capture Animator: Probe Games.” 2003. Retrieved 29 July 2003 from http://careers.awn.com/jobdisplay.php3?job_no=5497.

Hong, Joshua C. “BCD Forum: The State of the Online Game Industry.” 18 September 2002. Retrieved 3 August 2003 from http://www.k2network.net/BCD%20Forum%20%20State%20of%20the%20Online%20Game%20Industry%20(Halifax).ppt.

“Indiana Career and Postsecondary Advancement Center.” N.d. Retrieved 26 July 2003 from http://icpac.indiana.edu/careers/career_profiles/235701.xml.print

International Telecommunication Union (ITU). “Internet for a Mobile Generation: Executive Summary.” Geneva: ITU, 2002. Retrieved 3 August 2003 from http://www.itu.int/osg/spu/publications/sales/mobileinternet/execsumFinal.pdf.

International Telecommunication Union (ITU). “Top 15 economies by 2002 broadband penetration, 2002.” Retrieved 3 August 2003 from http://www.itu.int/ITU-D/ict/statistics/at_glance/top15_broad.html.

International Telecommunication Union (ITU). “Workshop on Promoting Broadband.” 11 April 2003. Retrieved 3 August 2003 from http://www.itu.int/osg/spu/ni/promotebroadband/PB11-ChairmansReport.pdf.

Ito, Mizuko, and Okabe Daisuke. “Mobile Phones, Japanese Youth, and the Re-Placement of Social Contact.” Paper presented at conference, Front Stage – Back Stage: Mobile Communication and the Renegotiation of the Social Sphere, on 23 June 2003 in Grimstad, Norway. Retrieved 3 August 2003 from http://www.itofisher.com/PEOPLE/mito/mobileyouth.pdf

“Job descriptions: Jobs in interactive digital media.” N.d. Retrieved 17 July 2003 from http://www.skillsnet.net/occguide/JDidmanim. cfm

“Job Roles.” N.d. Retrieved 17 July 2003 from http://www.blitzgames.com/gameon/jobroles.htm.

Joslyn, Cliff, Francis Heylighen, and Valentin Turchin. “Metasystem Transition Theory.” Principia Cybernetica Web: Created January 1992, modified 7 July 1997. Retrieved 4 August 2003 from http://pespmc1.vub.ac.be/MSTT.html.

“KoDEC 2002: Korea Digital Entertainment Conference 2002.” April 2002. Retrieved 3 August 2003 from http://www.gdah.org/admin/event/event/KODEC.pdf.

Krahulik, and Holkins. “The Game Tester’s Manifesto.” 2002. Retrieved 28 July 2003 from http://www.penny-arcade.com/porktester.php3.

Kuhn, Thomas S. *The Structure of Scientific Revolutions*. Chicago: University of Chicago Press, 1962.

Kuittinen, Petri. “Background of Video Games.” October-December 1997. Retrieved 4 August 2003 from http://www.hut.fi/~eye/videogames/intro.html.

Küppers, Günter. “Self-organization—The Emergence of Order: From Local Interactions to Global Structures.” University of Bielefeld: n.d. Retrieved 4 August 2003 from http://www.uni-bielefeld.de/iwt/sein/paperno2.pdf.

Kurzweil, Ray. “The Law of Accelerating Returns.” 7 March 2001. Retrieved 4 August 2003 from http://www.kurzweilai.net/articles/art0134.html?printable=1.

“Latest Global, Handset, Base Station, & Regional Cellular Statistics.” N.d. Retrieved 3 August 2003 from http://www.cellular.co.za/stats/stats-main.htm.

“Lead Texture Artist.” Posting on gamesindustry.biz. 16 April 2003. Retrieved 17 July 2003 from http://www.gamesindustry.biz/recruitment_page.php?action=view&job_id=3294.

Leonard, Jean-Pierre. “Audio Engineer.” 2003. Retrieved 30 July 2003 from http://www.jobpostings.ca/jobsearch/companyjobdetail.cfm?JobID=3389.

“Level Designer.” Games-Jobs.com. N.d. Retrieved 17 July 2003 from http://www.games-jobs.com/.

“Level Designer: Job Description.” N.d. Retrieved 29 July 2003 from http://www.lostboysgames.com/pages/job3.htm.

Linuxdevices.com. August 2003. Retrieved 11 August 2003 from http://linuxdevices.com/news/NS3794366397.html.

Macedonia, Michael, and J.C. Herz. “Games and Education.” N.d. Retrieved 4 August 2003 from http://www.educause.edu/ir/library/pdf/ffp0206s

Macedonia, Michael. “Games and Simulation in the Military Education Dilemma.” N.d. Retrieved 4 August 2003 from http://www.educause.edu/ir/library/pdf/ffpiu018.pdf.

Maguire, Flack, Dr. Michael van Lent, Marc Prensky, and Ron W. Tarr. “Defense Combat Sim Olympics—Methodologies Incorporating the ‘Cyber Gaming Culture.’” N.d. Retrieved 4 August 2003 from http://www.marcprensky.com/writing/IITSEC%20Paper%202002%20(536%20V2-Final).pdf.

Maniar, Sumeet D., and Hong, Joshua C. “BCD Forum, Supercomm: US Online Broadband Gaming Opportunity.” June 2003. Retrieved 3 August 2003 from http://www.bcdforum.org/events/SUPERCOMM2003/K2_Network.pdf.

“Merging Micro- and Nanotechnologies.” *MST News: International Newsletter on MICROSYSTEMS and MEMS* 3/03 (June 2003). Retrieved 3 August 2003 from http://www.mstnews.de/pdf_aktuell/news0303_1.pdf.

Monroe, Doug. Interview, 29 July 2003. Information Technology and Security Academy and Alamo-Area Aerospace Academy.

"Motion Capture—what is it?" MetaMotion: n.d. Retrieved 29 July 2003 from http://www.metamotion.com/motion-capture/motioncapture.htm.

Nathan, Rick, and Trevor Fencott. "Emerging Financial Opportunities in the Interactive Media Industry." Goodmans Venture Group: March 2003. Retrieved 3 August 2003 from www.goodmansventuregroup.com/PDFs/Interactive%20Media.pdf.

National Institute of Advanced Industrial Science and Technology. "Nanotechnology Research Institute." N.d. Retrieved 3 August 2003 from http://www.aist.go.jp/aist_e/ressearch_units/research_section/nanotech/nanotech_main.html.

Niemeyer, Greg. "S 240 > Nonlinear Narratives > Games from Sub to Meta." N.d. Retrieved 26 July 2003 from http://art.berkeley.edu/coursework/niemeyer/courses/fs240/wk02_text.htm.

Nobel, C. "New Products Bridge WiFi and Cellular." *EWeek:* 10 March 2003. Retrieved 3 August 2003 from http://wireless.ziffdavis.com/article2/0,3973,923512,00.asp.

Office of Technology Policy. U.S. Department of Commerce. "Understanding Broadband Demand: A Review of Critical Issues." 23 September 2002. Retrieved 3 August 2003 from http://www.ta.doc.gov/reports/TechPolicy/Broadband_020921.htm.

"Online Game Market is Growing but Making Money is Difficult." *DFC Intelligence*: 25 June 2003. Retrieved 3 August 2003 from http://www.dfcint.com/news/prjune252003.html.

Paté-Cornell, M. Elisabeth. "Management of Post-Industrial Systems: Academic Challenges and the Stanford Experience." Stanford University, n.d. Retrieved 3 August 2003 from http://in3.dem.ist.utl.pt/s_issue/pdf/4.PDF.

Pearce, Celia. "Emergent Cultures in Online Games." California Institute for Telecommunications and Information Technology, n.d. Retrieved 4 August 2003 from www.crito.uci.edu/critohours/2003-05-29.pdf.

Personal Computer Milestones. N.d. Retrieved 3 August 2003 from http://www.blinkenlights.com/pc.shtml.

"PlayStation 3 Specs and Rumors." PVC Ask Hank: 30 November 2001. Retrieved 3 August 2003 from http://www.pcvsconsole.com/hank/answer/206.html.

Porter, Michael E. "The Competitiveness of Nations." 1990. Retrieved 4 August 2003 from http://www.cbs.dk/departments/ikl/sproek/bachelor/noter/bachelor2/efteraar-2001/Porters-Diamond.ppt.

"Report: Online Gaming Reaching Critical Point." *GameMarketWatch.com*: 18 September 2002. Retrieved 3 August 2003 from http://www.gamemarketwatch.com/news/item.asp?nid=2562.

Reynolds, Brian. “Facts About the Computer Games Industry by Brian Reynolds.” Retrieved 28 July 2003 from http:// www.bighugegames.com/jobs/industryjobtips.html.

Robertson, B. “2D/3D Artist**:** WMS Gaming, Inc.” 2003. Retrieved 16 July 2003 from http://careers.awn.com/jobdisplay.php3?job_no=5464.

Salamini, Leonardo. “InterLabs: An Interdisciplinary Laboratory Where Students Lead the Academic Computing Revolution.”*Technological Horizons in Education*: 1998. Retrieved 4 August 2003 from http://www.thejournal.com/magazine/vault/A2025.cfm.

“Salary survey reveals thriving game industry.” *Austin Business Journal*: June 17, 2002. Electronic version.
Seun, Moon, and Kevin Geiger. “Across the Great Divide II: What’s Out There for You.” July 2002. Retrieved 29 July 2003 from http:// www.simplisticpictures.com/CGIjobs.html.

“Specialized Tools for Motion Capture.” MetaMotion: n.d. Retrieved 29 July 2003 from http://www.metamotion.com/motion-capture/specialized-motion-capture-tools-1.htm.

Stanford University. “Stanford University Annual Report: Thinking on New Lines.” Stanford Univ.: 2002. Retrieved 3 August 2003 from http://www.stanford.edu/home/administration/ report2002.pdf.

Stone, Sandy. Interview, 1 August 2003. University of Texas-Austin ACTLAB.
“Storyboard artist.” Career Connections: 16 July 2003. Retrieved 17 July 2003 from http://careers.awn.com/obdisplay.php3?job_no=6268.

Synergy Research Group. “WLAN Market Grows 100% in 2002 to $1.8 Billion Vendors Ship 16 Million Devices.” 12 February 2003. Retrieved 3 August 2003 from http://www.srgresearch.com/store/press/2-12-03.html?SID=1&.

TANSEI: The University of Tokyo Magazine. Univ. of Tokyo 2 (March 2002). Retrieved 3 August 2003 from http://www.u-tokyo.ac.jp/eng/tansei/TANSEI02.pdf.

Tapscott, Don. “Growing Up Digital: The Rise of the Net Generation.” Meridian, n.d. Retrieved 4 August 2003 from http:// www.ncsu.edu/meridian/jan98/feat_6/digital.html.

“The Culture of Interaction: The Ten Themes of N-Gen Culture.” Retrieved 4 August 2003 from http://www.growingupdigital.com/FIcult.html.

Growing Up Digital: The Rise of the Net Generation. New York: McGraw-Hill, 1998.

“Technical Level Designer.” Gamesindustry.com: 6 June 2003. Retrieved 17 July 2003 fromhttp://www.gamesindustry.biz/recruitment_page.php?action=view&job_id=3491.

"The Interactive Culture Industry for the Danish Ministry of Culture." *KPMG*: 4 July 2002. Retrieved 3 August 2003 from www.segera.ruc.dk/Christian%20Fonnesbech.pdf.

The Office of Scientific and Technical Information (OSTI). "Basic Science for the Nation's Future." April 2002. Retrieved 3 August 2003 from http://www.sc.doe.gov/Sub/About/basic.pdf.

The Official Counter-Strike Web Site. N.d. Retrieved 4 August 2003 from http://www.counter-strike.net.

Thomson, Kimball. "Cracking the Commercial Code: The SCI Institute Moves Its Collaborative Magic to the Marketplace." *Wasatch Digital iQ*: September 2001. Retrieved 3 August 2003 from http://dced.utah.gov/techdev/CRACKING_THE_CODE.pdf.

"Tools and libraries software engineer." 2003. Retrieved 31 July 2003 from http://www.bctechnology.com/scripts/show_job.cfm.

"Top Ten 3G Ready Nations." 19 February 2003. Retrieved 3 August 2003 from http://www.3g.co.uk/PR/Feb2003/4921.htm.

Turpin, T. "Transdisciplinary Research and the Dissemination of Ideas: A Paradox for Academic Science in the 21st Century." Centre for Research Policy University of Wollongong: June 2002. Retrieved 3 August 2003 from http://www.uow.edu.au/commerce/ibri/opaper6.pdf.

"Usability Analyst/Engineer." 2001. Retrieved 31 July 2003 from http://www.usabilityprofessionals.org/redesign/resources/res_jobbank_detail1.htm.

U.S. Department of Commerce, Bureau of Economic Analysis. "National Income and Product Accounts Tables." 4 August 2003. Retrieved 4 August 2003 from http://www.bea.doc.gov/bea/dn/nipaweb/index.asp.

U.S. Department of Labor, Bureau of Labor Statistics. Occupational Employment Statistics. "2001 National Occupational Employment and Wage Estimates." Retrieved 26 July 2003 from http://www.bls.gov/oes/2001/oes272012.htm.

"Video Game Companies See Increase in Revenue Income According to New DFC Intelligence Report." *DFC Intelligence*: 11 February 2003. Retrieved 3 August 2003 from http://www.dfcint.com/news/prfeb112003.html.

"Video production process." N.d. Retrieved 29 July 2003 from http://www.mediatechpro.com/vidproc.htm

W.R. Hambrecht & Co. "New Wireless Opportunities Home Networking Focus: UWB vs. WLAN—WLAN Has First Mover Advantage."28 March 2003. Retrieved 4 August 2003 from http://www.sims.berkeley.edu/~paulette/courses/Spring03_HW/IS290_WirelessComm/11_hambrecht-UWB_v_WLAN-03-03.pdf.

Wellman, T. "Computer & video game designers." Career of the Month: February 2003. Career & Employment Services, Lansing

Community College: 2003. Retrieved 26 July 2003 from http://216.239.57.104/search?q=cache:LRUTC8S_clgJ:www.lcc.edu/ces/RTFdocuments/careerofthemonth/February%25202003%2520(Computer%2520%26%2520Video%2520Game%2520Designers).doc+video+game+designer+salaries&hl=en&ie=UTF-8.

"What are Cybernetics and Systems Science?" *Principia Cybernetica*: 1999. Retrieved 4 August 2003 from http://pespmc1.vub.ac.be/CYBSWHAT.html.

Wireless Gaming Review, http://www.wgamer.com/gamedir/, July 31, 2003

World Technology Evaluation Center (WTEC). "Converging Technologies for Improving Human Performance" (prepublication online version), n.d. Retrieved 3 August 2003 from http://www.wtec.org/ConvergingTechnologies/Report/NBIC_A_MotivationOutlook.pdf.

"Converging Technologies for Improving Human Performance." June 2002. Retrieved 4 August 2003 from http://www.wtec.org/ConvergingTechnologies/Report/NBIC_frontmatter.pdf.

"Xbox 2 Estimated Specifications." 12 December 2001. Retrieved 3 August 2003 from http://www.pcvsconsole.com/hank/answer/229.html.

Yaniv, Zvi. Phone interview, Applied Nanotech, 11 July 2003.

Zyda, Michael. "Modeling & Simulation: Linking Entertainment & Defense." Naval Postgraduate School: n.d. Retrieved 4 August 2003 from http://www.movesinstitute.org/~zyda/presentations/NRCDMSORevisedTalk.pdf.

"The Naval Postgraduate School Moves Program—Entertainment Research Directions Moves Academic Group." Naval Postgraduate School: n.d. Retrieved 4 August 2003 from http://www.npsnet.org/~zyda/pubs/SCSC2000.pdf.

Tapscott, Don. "Growing Up Digital: The Rise of the Net Generation." Meridian, n.d. Retrieved 4 August 2003 from http://www.ncsu.edu/meridian/jan98/feat_6/digital.html.

"The Culture of Interaction: The Ten Themes of N-Gen Culture." Retrieved 4 August 2003 from http://www.growingupdigital.com/FIcult.html.

Growing Up Digital: The Rise of the Net Generation. New York: McGraw-Hill, 1998.

"Technical Level Designer." Gamesindustry.com: 6 June 2003. Retrieved 17 July 2003 fromhttp://www.gamesindustry.biz/recruitment_page.php?action=view&job_id=3491.

"The Interactive Culture Industry for the Danish Ministry of Culture." *KPMG*: 4 July 2002. Retrieved 3 August 2003 from www.segera.ruc.dk/Christian%20Fonnesbech.pdf.

The Office of Scientific and Technical Information (OSTI). “Basic Science for the Nation’s Future.” April 2002. Retrieved 3 August 2003 from http://www.sc.doe.gov/Sub/About/basic.pdf.

The Official Counter-Strike Web Site. N.d. Retrieved 4 August 2003 from http://www.counter-strike.net.

Thomson, Kimball. “Cracking the Commercial Code: The SCI Institute Moves Its Collaborative Magic to the Marketplace.” *Wasatch Digital iQ*: September 2001. Retrieved 3 August 2003 from http://dced.utah.gov/techdev/CRACKING_THE_CODE.pdf.

“Tools and libraries software engineer.” 2003. Retrieved 31 July 2003 from http://www.bctechnology.com/scripts/show_job.cfm.

“Top Ten 3G Ready Nations.” 19 February 2003. Retrieved 3 August 2003 from http://www.3g.co.uk/PR/Feb2003/4921.htm.

Turpin, T. “Transdisciplinary Research and the Dissemination of Ideas: A Paradox for Academic Science in the 21st Century.” Centre for Research Policy University of Wollongong: June 2002. Retrieved 3 August 2003 from http://www.uow.edu.au/commerce/ibri/opaper6.pdf.

“Usability Analyst/Engineer.” 2001. Retrieved 31 July 2003 from http://www.usabilityprofessionals.org/redesign/resources/res_jobbank_detail1.htm.

U.S. Department of Commerce, Bureau of Economic Analysis. “National Income and Product Accounts Tables.” 4 August 2003. Retrieved 4 August 2003 from http://www.bea.doc.gov/bea/dn/nipaweb/index.asp.

U.S. Department of Labor, Bureau of Labor Statistics. Occupational Employment Statistics. “2001 National Occupational Employmentand Wage Estimates.” Retrieved 26 July 2003 from http://www.bls.gov/oes/2001/oes272012.htm.

“Video Game Companies See Increase in Revenue Income According to New DFC Intelligence Report.” *DFC Intelligence*: 11 February 2003. Retrieved 3 August 2003 from http://www.dfcint.com/news/prfeb112003.html.

“Video production process.” N.d. Retrieved 29 July 2003 from http://www.mediatechpro.com/vidproc.htm.

W.R. Hambrecht & Co. “New Wireless Opportunities Home Networking Focus: UWB vs. WLAN—WLAN Has First Mover Advantage.”28 March 2003. Retrieved 4 August 2003 from http://www.sims.berkeley.edu/~paulette/courses/Spring03_HW/IS290_WirelessComm/11_hambrecht-UWB_v_WLAN-03-03.pdf.

Wellman, T. “Computer & video game designers.” Career of the Month: February 2003. Career & Employment Services, Lansing Community College: 2003. Retrieved 26 July 2003 from http://216.239.57.104/search?q=cache:LRUTC8S_clgJ:www.lcc.edu/ces/

RTFdocuments/careerofthemonth/February%25202003%2520(Computer%2520%26%2520Video%2520Game%2520Designers).doc+video+game+designer+salaries&hl=en&ie=UTF-8.

"What are Cybernetics and Systems Science?" *Principia Cybernetica*: 1999. Retrieved 4 August 2003 from http://pespmc1.vub.ac.be/CYBSWHAT.html.

Wireless Gaming Review, http://www.wgamer.com/gamedir/, July 31, 2003

World Technology Evaluation Center (WTEC). "Converging Technologies for Improving Human Performance" (prepublication online version), n.d. Retrieved 3 August 2003 from http://www.wtec.org/ConvergingTechnologies/Report/NBIC_A_MotivationOutlook.pdf.

"Converging Technologies for Improving Human Performance." June 2002. Retrieved 4 August 2003 from http://www.wtec.org/ConvergingTechnologies/Report/NBIC_frontmatter.pdf.

"Xbox 2 Estimated Specifications." 12 December 2001. Retrieved 3 August 2003 from http://www.pcvsconsole.com/hank/answer/229.html.

Yaniv, Zvi. Phone interview, Applied Nanotech, 11 July 2003.

Zyda, Michael. "Modeling & Simulation: Linking Entertainment & Defense." Naval Postgraduate School: n.d. Retrieved 4 August 2003 from http://www.movesinstitute.org/~zyda/presentations/NRCDMSORevisedTalk.pdf.

"The Naval Postgraduate School Moves Program—Entertainment Research Directions Moves Academic Group." Naval Postgraduate School: n.d. Retrieved 4 August 2003 from http://www.npsnet.org/~zyda/pubs/SCSC2000.pdf.

SIMULATED CARD AND BOARD GAMES

Multi-agent Modeling of Interaction-based Card Games

Evan Hurwitz
School of Electrical and Information Engineering
University of the Witwatersrand
Johannesburg, Gauteng, South Africa
e.hurwitz@ee.wits.ac.za

Tshilidzi Marwala
School of Electrical and Information Engineering
University of the Witwatersrand
Johannesburg, Gauteng, South Africa
t.marwala@ee.wits.ac.za

Abstract – *Many card games have their strategy defined at high levels of play not by statistical probabilities of cards being drawn, but rather by the interactions of the players themselves. These interactions, often based on prediction of the opposing players' likely decisions and/or holdings, become the determining factors in developing a successful strategy. This play has often been seen as psychological, falling under such labels as* bluffing *and* illogical play. *The creation of a system that is truly free enough to develop its own strategy without influence from the limited expertise of the engineer is detailed within, being of crucial importance. Through use of intelligent agents that learn to play a game purely through empirical observation, it is shown that agents can in fact be created that are capable of reproducing such behaviour, directly disputing the notion of such actions being either illogical or psychological in nature, but rather resulting from incorporating the opponents into the modelled system.*

Keywords: reinforcement, learning, temporal, difference, neural, network, Lerpa, Agent, Modelling.

1 Introduction

Traditional A.I. agents use simple statistical methods and rules to govern their decisions in competitive card games. These approaches, unfortunately, tend to make these agents somewhat predictable, and hence no real match for a human opponent, while also becoming useless for any realistic simulation / analysis of the game involved, due to the oversimplification of the interactions within it. In order to create a strategy that is based upon the anticipated reactions of one's opponents, rather than the expected intrinsic value of one's held cards (one's *hand*), a system needs to be defined that allows for intelligent virtual players, or agents, to interact with each other, and refine their own strategies iteratively. The individual agents also need to have enough freedom of strategy creation to be unbound by the preconceptions of the designer, effectively reproducing true learning, rather than learning what the designer presents to them. With these criterion met, the agents will be able to learn not only the general nuances of the game at hand, but also the preferred strategies of their opponents.

2 Lerpa

The card game being modelled is the game of Lerpa. While not a well-known game, it's rules suit the purposes of this research exceptionally well, making it an ideal testbed application for intelligent agent MAM. The rules of the game first need to be elaborated upon, in order to grasp the implications of the results obtained. Thus, the rules for Lerpa now follow.

The game of *Lerpa* is played with a standard deck of cards, with the exception that all of the 8s, 9s and 10s are removed from the deck. The cards are valued from greatest- to least-valued from ace down to 2, with the exception that the 7 is valued higher than a king, but lower than an ace, making it the second most valuable card in a suit. At the end of dealing the hand he then flips the next card in the deck to determine the trump suit. Regardless, once trumps are determined, the players then take it in turns, going clockwise from the dealer's left, to elect whether or not to play the hand (to *knock*), or to drop out of the hand, referred to as *folding*. Once all players have chosen, the players that have elected to play then play the hand, with the player to the dealer's left playing the first card. Once this card has been played, players must then play *in suit* – in other words, if a heart is played, they must play a heart if they have one. If they have none of the required suit, they may play a trump, which will win the trick unless another player plays a higher trump. The highest card played will win the trick (with all trumps valued higher than any other card) and the winner of the trick will lead the first card in the next trick. At any point in a hand, if a player has the Ace of trumps and can legally play it, he is then required to do so. The true risk in the game comes from the betting, which occurs as follows:

At the beginning of the round, the dealer pays the table 3 of whatever the basic betting denomination is (referred to usually as 'chips'). At the end of the hand, the chips are divided up proportionately between the winners, i.e. if you win two tricks, you will receive two thirds of whatever is in the pot. However, if you stayed in, but did not win any tricks, you are said to have been *Lerpa'd*, and are then required to match whatever was in the pot for the next hand, effectively costing you the pot. It is in the evaluation of this risk that most of the true skill in *Lerpa* lies.

The fact that the game's rules are very simple, and that the complexity lies in the player's interactions, is one very compelling reason to use this game. The multi-stage nature of the game is also attractive for use with the TD(λ) update algorithm.

3 Lerpa MAM

As with any optimisation system, very careful consideration needs to be taken with regards to how the system is structured, since the implications of these decisions can often result in unintentional assumptions made by the system created. With this in mind, the Lerpa Multi-Agent System (MAS) has been designed to allow the maximum amount of freedom to the system, and the agents within, while also allowing for generalisation and swift convergence in order to allow the intelligent agents to interact unimpeded by human assumptions, intended or otherwise.

3.1 System overview

The game is, for this model, going to be played by four players. Each of these players will interact with each other indirectly, by interacting directly with the *table*, which is their shared environment, as depicted in Figure 1.

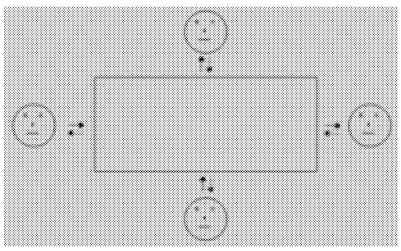


Figure 1. System interactions.

Over the course of a single hand, an agent will be required to make three decisions, once at each interactive stage of the game. These three decision-making stages are:

1. To play the hand, or drop (*knock* or *fold*)
2. Which card to play first
3. Which card to play second

Since there is no decision to be made at the final card, the hand can be said to be effectively finished from the agent's perspective after it has played it's second card (or indeed after the first decision should the agent fold). Following on the TD(λ) algorithm, each agent will update it's own neural network at each stage, using it's own predictions as a reward function, only receiving a true reward after its final decision has been made. This decision making process is illustrated below, in Figure 2.

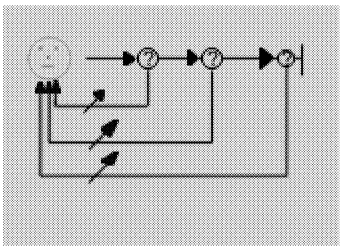


Figure 2. Agent learning scheme

With each agent implemented as described, they can now interact with each other through their shared environment, and will continuously learn upon each interaction and it's consequent result.

3.2 Agent AI design

A number of decisions need to be made in order to implement the agent AI effectively and efficiently. The type of learning to be implemented needs to be chosen, as well as the neural network architecture. Special attention needs to be paid to the design of the inputs to the neural network, as these determine what the agent can 'see' at any given point. This will also determine what assumptions, if any, are implicitly made by the agent, and hence cannot be taken lightly. Lastly, this will determine the dimensionality of the network, which directly affects the learning rate of the network, and hence must obviously be minimised.

3.2.1 Input Parameter Design

In order to design the input stage of the agent's neural network, one must first determine all that the network may need to know at any given decision-making stage. All inputs, in order to optimise stability, are structured as binary-encoded inputs. When making its first decision, the agent needs to know it's own cards, which agents have stayed in or folded, and which agents are still to decide. It is necessary for the agent to be able to determine which specific agents have taken their specific actions, as this will allow for an agent to learn a particular opponent's characteristics, something impossible to do if it can only see a number of players in or out. Similarly, the agent's own cards must be specified fully, allowing the agent to draw its own conclusions about each card's relative value. It is also necessary to tell the agent which suit has been designated the trumps suit, but a more elegant method has been found to handle that information, as will be seen shortly. Figure 3 below illustrates the initial information required by the network.

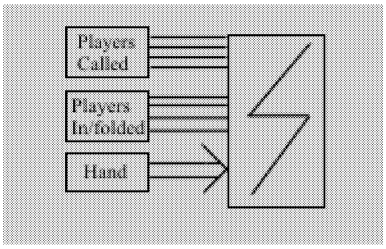


Figure 3. Basic input structure.

The agent's hand needs to be explicitly described, and the obvious solution is to encode the cards exactly, i.e. four suits, and ten numbers in each suit, giving forty possibilities for each card. A quick glimpse at the number of options available shows that a raw encoding style provides a sizeable problem of dimensionality, since an encoded hand can be one of 40^3 possible hands (in actuality, only ${}^{40}P_3$ hands could be selected, since cards cannot be repeated, but the raw encoding scheme would In fact allow for repeated cards, and hence 40^3 options would be available). The first thing to notice is that only a single deck of cards is being used, hence no card can ever be repeated in a hand. Acting on this principle, consistent ordering of the hand means that the base dimensionality of the hand is greatly reduced, since it is now combinations of cards that are represented, instead of permutations. The number of combinations now represented is ${}^{40}C_3$. This seemingly small change from ${}^{n}P_r$ to ${}^{n}C_r$ reduces the dimensionality of the representation by a factor of r!, which in this case is a factor of 6. Furthermore, the representation of cards as belonging to discrete suits is not optimal either, since the game places no particular value on any suit by its own virtue, but rather by virtue of which suit is the trump suit. For this reason, an alternate encoding scheme has been determined, rating the 'suits' based upon the makeup of the agent's hand, rather than four arbitrary suits. The suits are encoded as belonging to one of the following groups, or new "suits":

- Trump suit
- Suit agent has multiple cards in (not trumps)
- Suit in agent's highest singleton
- Suit in agent's second-highest singleton
- Suit in agent's third-highest singleton

This allows for a much more efficient description of the agent's hand, greatly improving the dimensionality of the inputs, and hence the learning rate of the agents.

Next must be considered the information required in order to make decisions two and three. For both of these decisions, the cards that have been already played, if any, are necessary to know in order to make an intelligent decision as to the correct next card to play. For the second decision, it is also plausible that knowledge of who has won a trick would be important. The most cards that can ever be played before a decision must be made is seven, and since the table after a card is played is used to evaluate and update the network, eight played cards are necessary to be represented. The actual values of the cards played are not important, only their values relative to the agent's cards. As such, the values can be represented as one of the following, with respect to the cards in the same suit in the agent's hand:

- Higher than the card/cards in the agent's hand
- Higher than the agent's second-highest card
- Higher than the agent's third-highest card
- Lower than any of the agent's cards
- Member of a void suit (number is immaterial)

Also, another suit is now relevant for representation of the played cards, namely a void suit. Lastly, a number is necessary to handle the special case of the Ace of trumps, since it's unique rules mean that strategies are possible to develop based on whether it has or has not been played. The now six suits available still only require three binary inputs to represent, and the six number groupings now reduce the value representations from four binary inputs to three binary inputs, once again reducing the dimensionality of the input system.

3.2.2 Network Architecture Design

With the inputs now specified, the hidden and output layers need to be designed. For the output neurons, these need to represent the prediction P that the network is making. A single hand has one of five possible outcomes, all of which need to be catered for. These possible outcomes are:

- The agent wins all three tricks, winning 3 chips.
- The agent wins two tricks, winning 2 chips.
- The agent wins one trick, winning 1 chip.
- The agent wins zero tricks, losing 3 chips.
- The agent elects to fold, winning no tricks, but losing no chips.

This can be seen as a set of options, namely [-3 0 1 2 3]. While it may seem tempting to output this as one continuous output, the facts that the results are in fact discrete, and that discrete reqrds are better for stability

purposes are compelling enough to represent the outputs in binary format. Consequently, the agent's predicted return is:

$$P = 3A + 2B + C - 3D \quad (1)$$

where

$$A = P(O = 3) \quad (2)$$

$$B = P(O = 2) \quad (3)$$

$$C = P(O = 1) \quad (4)$$

$$D = P(O = -3) \quad (5)$$

3.2.3 Agent decision making

With it's own predictor specified, the agent is now equipped to make decisions when playing. These decisions are made by predicting the return of the resultant situation arising from each legal choice it can make. An ε-greedy policy is then used to determine whether the agent will choose the most promising option, or whether it will explore the result of the less appealing result. In this way, the agent will be able to trade off exploration versus exploitation.

4 The intelligent model

With each agent implemented as described above, and interacting with each other as specified in section three, we can now perform the desired task, namely that of utilising a multi-agent model to analyse the given game, and develop strategies that may "solve" the game given differing circumstances. Only once agents know how to play a certain hand can they then begin to outplay, and potentially bluff each other.

4.1 Agent learning verification

In order for the model to have any validity, one must establish that the agents do indeed. In order to verify the learning of the agents, a single intelligent agent was created, and placed at a table with three 'stupid' agents. These 'stupid' agents always stay in the game, and choose a random choice whenever called upon to make a decision. The results show quite conclusively that the intelligent agent soon learns to consistently outperform its opponents, as shown in Figure 4. AIden is the A.I. agent, while Randy, Roderick and Ronald are all random decision-makers.

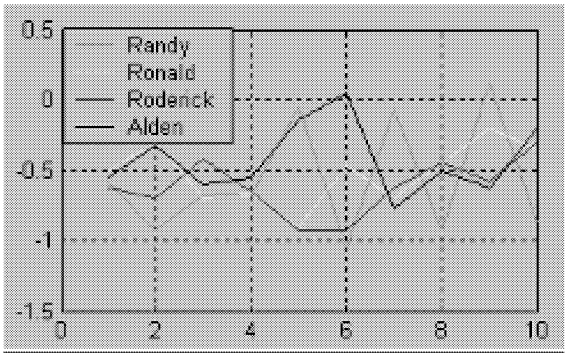


Figure 4. Agent performance, averaged over 40 hands

4.1.1 Cowardice

In the learning phase of the abovementioned intelligent agent, an interesting and somewhat enlightening problem arises. When initially learning, the agent does not in fact continue to learn. Instead, the agent quickly determines that it is losing chips, and decides that it is better off not playing, and keeping its chips!

AIden quickly decides that the risks are too great, and does not play in any hands initially. After forty hands, AIden decides to play a few hands, and when they go badly, gets scared off for good. This is a result of the penalising nature of the game, since bad play can easily mean one loses a full three chips and thus a bad player loses chips regularly. While insightful, a cowardly agent is not of any particular use, and hence the agent must be given enough 'courage' to play, and hence learn the game. The most sensible approach is the 'school-fees' approach, ie forcing the agent to play until comfortable, in effect paying his school fees to learn the game, until such a stage as it seems prepared to play. This was done by forcing AIden to play the first 200 hands it had ever seen, and thereafter leave AIden to his own devices..

4.2 Agent Adaptation

In order to ascertain whether the agents in fact adapt to each other or not, the agents were given pre-dealt hands, and required to play them against each other repeatedly. The results one such experiment, illustrated in Figure 9, shows how an agent learns from its own mistake, and once certain of it changes its play, adapting to better gain a better return from the hand. The mistakes it sees are its low returns, returns of -3 to be precise. At one point, the winning player obviously decides to explore, giving some false hope to the losing agent, but then quickly continues to exploit his advantage. Eventually, at game #25, the losing agent gives up, adapting his play to suit the losing situation in which he finds himself. Figure 5 illustrates the progression of the agents and the adaptation described.

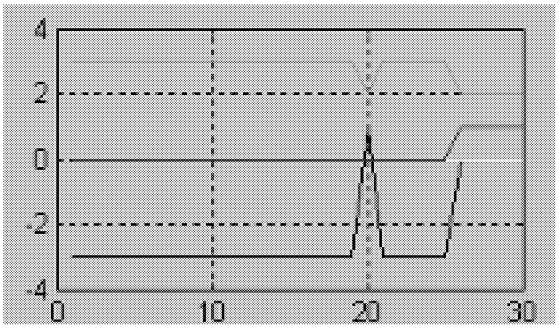


Figure 5. adaptive agent behaviour

4.3 Personality Profiling

Another advantage of the Multi-Agent system is that one need not make the assumption that all players are 'rational', as is assumed in traditional Game-Theory. Many poker players of reasonable skill complain bitterly about playing against beginners, bemoaning the fact that the beginners do not play as they should, as a "rational" player would, and thus throw the better players off their game. A good player, however, should not be limited to assuming rationality from his opponents, but rather should identify his opponents' characteristics and exploit their weaknesses. In order to perform this task, one needs to create "personality" types in agents, this while sounding somewhat daunting, is in fact a rather simple task. All that is required is to modify the reward function for an agent (equation #1) to reflect the personality type to be modeled. Two personality types were created, namely an *aggressive* personality and a *conservative* personality. The aggressive agent uses the reward function:

$$P = 3A + 2B + C - 2D \tag{6}$$

While the conservative agent uses the reward function:

$$P = 3A + 2B + C - 4D \tag{7}$$

Where the symbols have the following meanings:

- P denotes the expected outcome of the hand
- A denotes the probability of winning three tricks.
- B denotes the probability of winning two tricks.
- C denotes the probability of winning one trick.
- D denotes the probability of winning zero tricks.

As can be seen, in this case both agents modify the coefficient of the D term in order to skew their world view. This is certainly not the only manner in which the reward function can be modified in order to reflect a personality, but is the most obvious, since the D term represents the "risk" that the agent sees within a hand. Using the above modifications, the previously detailed strategy analysis techniques can be performed on agents with distinctive personalities. The static analysis of comes to the same results as with rational players, due to the unchanging nature of the problem, while the dynamic analysis yields more interesting results. Dynamic strategy analysis finds different dominant strategies when playing against these profiled personalities. More specifically, the aggressive player is not considered as dangerous when playing in a hand, while the conservative player is treated with the utmost of respect, since he only plays the best of hands. Variations on these reqrd functions are of course possible, each representing different personality 'quirks'.

5 Conclusions and future work

The use of intelligent agents within a broadly-encompassing system has been shown to be capable of reproducing the complex strategies that humans employ when playing games with considerable interaction-based complexities. The agents have been shown to learn the game successfully, and also to adapt to each others' specific play-styles. This enables a researcher to find an optimum play for a specific hand, taking into account the other players, and not only the statistical odds of a hand having a positive likely outcome. Moreover, the ability to profile personalities allows a researcher to find the optimum play not only for a given hand in a given position, but also depending on the observed idiosynchrasies of the particular players that are sitting at the table. Proposed future work would be to optimise the learning speed of the agents, in order to be able to compete with human players.

References

Sutton, R. S, Barto A. G. *Reinforcement learning: An Introduction*, MIT press, 1998.

Sutton, R. S. *Learning to predict by the methods of temporal differences.* Mach. Learning 3*,* (1988), 9-44.

Hurwitz, Marwala. *Optimising Reinforcement Learning for Neural Networks*. Game-On 2005, Eurosis.

Morehead A. H, Mott-Smith G, Morehead P. D. *Hoyle's Rules of Games, Third revised and updated edition*. Signet book. 2001.

Sutton, R. S. *Implementation details of the TD(λ) procedure for the case of vector predictions and*

Backpropogation. GTE laboratories technical note TN87-509.1, 1989.

Sutton, R S. *Frequenlty asked questions about reinforcement learning*, last viewed 24/08/2005, http://www.cs.ualberta.ca/~sutton/RL-FAQ.html

Smart O. *Line Search and Descenets Methods.* http://www.biochemistry.bham.ac.uk/osmart/msc/lect/lect2a.html 02/02/1996. Last Viewed 20/04/2006

Wesley Hines J. *Matlab supplement to Fuzzy and Neural Approaches in Engineering.* John Wiley & Sons Inc. New York. 1997

Miller, Sutton and Werbos. *Neural Networks for Control.* MIT Press. Cambridge, Massecheussettes. 1990

ONLINE POKER SECURITY: PROBLEMS AND SOLUTIONS

Roman V. Yampolskiy
Computer Science and Engineering
University at Buffalo
2145 Monroe Ave. #4
Rochester, NY, 14618
rvy@buffalo.edu

KEYWORDS
Bots, Cheating, Games security, Online games, Poker.

ABSTRACT

Cheating in online computer games is becoming a significant problem as the popularity of such games steadily increases. As a result, it costs thousands of dollars to game designers in lost revenue from disillusioned players who stop participating and in man-hours used for prevention of different forms of cheating. In this paper we review different forms of cheating observed in computer games and in online poker in particular. This is followed by a review of solutions developed to counteract different forms of cheating in computer games.

INTRODUCTION

Multiplayer online computer games are quickly growing in popularity with millions of players logging in every day. While most play in accordance with the rules set up by the game designers, some choose to cheat, to gain an unfair advantage over other players. With the growth in the economic and social importance of the virtual game worlds incidence of cheating are becoming increasingly problematic (Brooke et al. 2004).

Cheating by some players makes the game less interesting for the honest players. As a result, it costs thousands of dollars to game designers in lost revenue from disillusioned players who stop participating and in man-hours used for prevention of different forms of cheating. Consequently, a great deal of current research in computer science is aimed at detecting, preventing and neutralizing cheating in game worlds (Baughman et al. 2006; Chambers et al. 2005; Chen and Maheswaran 2004; Kabus et al. 2005; Li et al. 2004). At the same time cheating is being investigated by researchers in media sciences who analyze social, ethical and moral nature of cheating in games and virtual worlds (Foo 2004; Foo and Koivisto 2004; Hayes, 2006; Kuecklich 2004; Kuo 2006; Sicart 2005; Smith 2004).

SECURITY ISSUES IN GAMES

What is cheating?

Cheating in games can be defined as "any behavior that a player uses to gain an advantage over his peer players or achieve a target in an online game … if, according to the game rules or at the discretion of the game operator(the game service provider, who is not necessarily the developer of the game), the advantage or the target is one that he is not supposed to have achieved" (Yan and Randell 2005; Yan and Choi 2002). Players themselves classify cheating into three often overlapping categories (Consalvo 2005):

- **Anything except unaided play** Using strategy guides, walkthroughs, cheat codes and hacking is all cheating.
- **Code tempering** Any modifications to the source code of the game is considered to be cheating.
- **Cheating other players** The strictest definition which says that cheating takes place only if another player is disadvantaged as a result of the cheaters actions.

A game cheat may have numerous motivations such as: obtaining free play, stealing virtual resources, completing otherwise difficult game quests, getting stuck, enjoying playing God, speeding up action, becoming a famous hacker, sabotaging the game provider out of revenge or jealousy, or acquiring virtual resources for sale in the real world (Consalvo 2005; Lyhyaoui et al. 2005). Rule violations can be classified based on who is being targeted as a victim of the cheating attack. While an attack against multiple targets is feasible, three distinct targets are suggested by Lyhyaoui et al. (Lyhyaoui et al. 2005):

- **Provider** A cheating attack against the game provider includes violation of implicit or explicit contract between the player as a customer and game developer as a service provider. Players may cheat the provider out of subscription fees for the service via the use of stolen credit cards. Colluding players may engage in theft of service via account sharing and so reduce the profitability of developing and running a game. In general, cheating players undermine the confidence of other players in the game's security and so reduce profitability for the game developer because of the reduced enrollment.
- **Players** The most frequent object of an attack is honest players not involved in cheating. Almost all attacks described in later parts of this paper deal with attacks aimed at deceiving this type of target. Their money and virtual assets may be stolen, accounts compromised and overall enjoyment of the game experience ruined.
- **Virtual society** The moral rules which allow the virtual social community to be rather stable may be compromised by unscrupulous users for personal advantage, resulting in

the breakdown of virtual society. A good example of this is known as *camping* which is a technique for remaining in the same advantages location in a virtual world in order to obtain resources and destroy opponents; while not explicitly illegal it makes the game less interesting for other players.

Types of cheating

Taxonomy of cheating methods used in computer games can provide a good starting point for addressing security issues of game design. A number of authors have proposed different classification schemas aimed at categorizing types of attacks seen perpetrated against this form of digital media. Prjtchard (Pritchard 2001) has proposed six different categories of cheating in online games:

1. **Reflex Augmentation** Using an artificially intelligent computer assistant to perform actions faster and with more precision. For example using an aim-bot in a first person shooter game to quickly and precisely target opponents.
2. **Authoritative Clients** Utilizing hacked clients to send altered commands to other players on the network to deceive them about the state of the game.
3. **Information Exposure** Obtaining access to hidden information by compromising client software. For example using a wall-hack to see your opponents through walls.
4. **Compromised Servers** Changing game state at the server level to obtain unfair advantage.
5. **Bugs and Design Loopholes** Taking advantage of the poor design of the game software either via security flaws or logical errors in the game model.
6. **Environmental Weaknesses** Abusing operating conditions or hardware configuration of the system's environment.

Yan et al. (Yan and Choi 2002) proposed a much broader attack taxonomy which incorporated many types of online computer game attacks not explicitly account for by Pritchard. A total of eleven cheating methodologies were presented, but the list was not complete and had to be expanded even further as new ways to cheat were discovered by the dishonest players. In (Yan and Randell 2005) Yan et al. proposed classifying attacks into 15 categories, but it is a safe bet that in the future this list will continue to grow as the computer games and all the related technologies continue to evolve and produce even more clever cheaters.

A. **Misplaced trust** From the client's side it is possible to modify the game client program, configuration data, or both to obtain previously unavailable privileges.
B. **Collusion** Combining of forces by multiple players to help each other, share information and work as a team against players not involved in collusion.
C. **Abuse of procedure or policy** Taking advantage of certain game server policies such as artificially terminating connection to a game server to avoid loosing a game.
D. **Virtual assets** Cheating by acquiring game assets via real money outside of game environment or cheating at such real money transactions.
E. **Machine intelligence** Cheating by using artificially intelligent assistant programs, a.k.a. bots to produce superior play. For example asking a world champion chess program to analyze a list of possible moves and represent the selected one as your own.
F. **Client infrastructure** By changing properties of client infrastructure for example making modification to the graphics driver a cheating player can alter graphics being displayed and so get access to information he is not intended to have like knowing what is on the other side of the brick wall.
G. **Denial of service** Often a server limits a number of login attempts to about three to prevent brute force guessing of passwords. A cheater may purposefully enter incorrect login information for the victim's account to prevent him from logging in to the server. Alternatively a player may be flooded with messages and other interactions preventing him from participating in a game in a timely manner.
H. **Timing** This cheat involves delaying own action in a real time game until actions of other players are known and thus obtaining an advantage over other players.
I. **Passwords** A compromised password allows access to another player's account including all the hidden information, virtual resources and ranking scores.
J. **Lack of secrecy** A cheating player may obtain secret information by observing unencrypted packets as they travel through the network.
K. **Lack of authentication** If there is no proper mechanism for authenticating a game server to clients a cheater can obtain user passwords by setting up a bogus game server.
L. **Design flows** This form of cheating takes advantage of game design mistakes such as exploiting inconsistencies in asset pricing within a virtual environment.
M. **Compromised game servers** A cheater can obtain access to the game host systems and tamper with game server programs.
N. **Internal misuse** An employee of a game server administrator may have privileges for creating virtual assets or super characters which can be sold for real money or generated as a favor to a cheating player.
O. **Social engineering** A method of tricking a player into voluntarily revealing his user name and password for example by impersonating a request from the game server administration.

Similar classification taxonomies have been proposed by others (Banavar 2006; Chen and Chen 2002; Lyhyaoui et al. 2005; Mørch 2003; Webb 2006) but most cite Yan et al. (Yan and Randell 2005) as their initial inspiration. The attack types presented above are considered atomic by definition. Complex attacks are comprised of multiple atomic attack techniques used together to achieve multiple goals or to take advantage of a multistage vulnerability.

SECURITY ISSUES IN POKER

While most security issues outlined above (passwords, denial of service, etc.) are valid concerns for online poker participants, many poker specific cheating methods deserve additional overview and analysis.

Card eavesdropping Observing the cards of opponents by capturing network traffic is only possible if the information is being sent unencrypted which never can happen in a respectable modern online casino. Some information may be gained if a weak or poorly implemented form in encryption is being utilized.

Client hacking As long as the information about opponents' cards is stored only on the game server and is not transmitted to every client, client hacking is a type of cheating which is not likely to present problems either to other players or to casino operators. Reports exist of people modifying images used by the poker client such as the actual representations of cards. Others have succeeded at hex-editing client software to allow registration of forbidden user names such as those used by casino employees or containing foul language (www.tips4poker.com 2006).

Exploiting bad randomness Poor design or implementation of a card shuffling algorithm can result in biased card distribution which can be taken advantage of to predict cards before they are reveled. Modern online casinos use independent security companies to verify correctness and security of their software code but in the past this type of problem has been successfully exploited (Arkin et al. 1999).

Escaping In many online casinos if a player gets disconnected because of a network failure or his computer freezes he is considered to be "all-in" for the amount of money he has bet so far. This feature is supposed to protect honest players from being penalized for hardware problems, but is often used by cheaters to avoid loosing additional funds. A cheating player simulates a loss of connection by disconnecting his computer from the network and by doing so avoids committing any more funds to the pot, but is nonetheless in contention for the winning of the pot.

Profile databases Because players in an online casino have a unique user ID and often play for many months if not years in the same casino it becomes possible to automate the task of player profiling and to do so in bulk, monitoring all players at the same time. Information about the players' aggressiveness, tendency to bluff, strengths, weaknesses, betting patterns and other statistics is automatically collected and made available to anyone interested for a small fee. This allows a subscriber to such a service to make educated decisions about strategy against certain players without investing time and funds necessary to learn their playing style. Web sites such as Poker-edge.com and Pokerprophecy.com are two of the most popular such services (Poker-edge.com, 2006; Pokerprophecy Retrieved 2006).

Collusion (active) A group of players working together attempt to steal pots by raising and re-raising each other to get unsuspecting players with marginal hands to fold. Typically one player has a strong hand while the accomplice is in the pot simply to increase the bets to the point where everyone caught between them folds. Once the contention for the pot is over the weaker of the two hands folds to a raise, and the strong hand gets the pot (Snyder 2006).

Illicit information passing This is the most popular method of online poker cheating. It involves two or more players revealing to each other what cards they are holding. Usually a communications channel independent from the casino such as a phone line or an instant messenger is utilized to accomplish this.

Self collusion Also known as the "Boiler Room" method and "Multi-Accounting", this cheating strategy involves setting up a number of computers in the same location and registering different accounts on each one of them yourself. This allows you to run a poker room where everyone in that room is colluding with you except for the victim who is quickly cheated out of his money (www.tips4poker.com 2006).

Player-poker room collusion In this scenario cheaters can manipulate the deck and trap other players into hands where the poker room partner will eventually win. This is accomplished by dealing a very good hand to the unsuspecting player which as additional community cards become revealed, or even immediately before the flop, is only second best.

Implicit collusion In a tournament settings where, for example, only the top 10 players are paid and there are 11 remaining, implicit collusion involves all players at the table just calling all the way through the river to maximize the likelihood that the short stack be busted by having all hands see the finish. This strategy is never explicitly discussed at the table, but is followed as it gives an advantage to most players.

Chip dumping (tournaments) A group of players who have agreed before the tournament to share profits play against each other very aggressively, with the goal of having one player end up with all the chips, and have the advantage over other players as the chip leader (Snyder 2006).

Excess action hands The poker room deals an excess number of great hands to encourage additional betting action. If the distribution of action hands is not biased it should not hurt individual players in the long run, with the exception of the additional rake being paid to the casino (www.playnoevil.com 2006).

Rake abuse Casino operator can have a very complicated or dynamic rake structure that reduces players' winnings. As the rake system becomes more complicated it becomes easier for the game operator to add additional fees (www.playnoevil.com 2006).

Table 1: Taxonomy of Cheating in Online Poker

Type of Cheating	Cheating Method	Offline	Exploiter			
			Independent		Cooperative	
			Single (account) Player	Casino Operator	Multi (account) Player	Casino-Player
Software Design Flows, Bugs, Abuses	Card eavesdropping	R	P		P	
	Client hacking		P		P	
	Exploiting bad randomness		R		R	
	Escaping		R		R	
	Excess action hands			P		
	Rake abuse			P		
Collusion	Collusion (active)	R			R	P
	Illicit information passing	R			R	P
	Self collusion				R	
	Player-poker room collusion	R				R
	Implicit collusion	R			R	
	Chip dumping (tournaments)	R			R	
Machine Intelligence	Profile databases		R		R	
	Bots		R		R	P
	Bot networks				R	
	Crowd bots			P		

Bots Artificially intelligent programs designed to play a hand automatically are considered to be a form of cheating by most players and online casinos. While they are currently not very good compared to all but the novice players, they are not feared by most, but they will undoubtedly improve in their performance and present real danger to online poker players of all levels in the near future.

Bot networks A number of bots at the same table can be connected to create an information sharking network in which all bots are actively helping each other to win either by simple information sharing or via active betting. Additionally, an even bigger edge can be obtained if such a bot network has access to an external database of players' profiles.

Crowd bots Some online casinos, particularly newly opened ones, in order to make it seem like they have a large base of regular patrons may employ artificial players to fill in any empty seats at the tables. Depending on quality of such bots it may either be to the advantage or disadvantage of real human players. However, if such bots have access to the information which a regular player would not have access to, such as unrevealed community cards, this might give a huge advantage to the casino.

Table 1 shows taxonomy of cheating methods used in online poker. Each cheating methods is grouped according to the type of attack it belongs to, as well as by who is capable of perpetrating such an act. Each attack is marked as either being possible (P) or actually reported (R). A blank means that such cheating method is not possible for a given assailant. "Offline" column is marked if a corresponding cheating method may be used in a regular brick and mortar casino.

EXISTING SOLUTIONS

A number of countermeasures have been proposed to combat game security violations, some of them are outlined below (Davis 2001):

- **Secure Game Contract** Ensures full synchronization of game state information between all parties of the game and reliable non-repudiatable communication of actions and state changes. Allows for full reconstruction of the game to certify that the game was played properly.
- **Trusted Gaming Infrastructure** Includes gaming operator's security infrastructure, interfaces between different parties, and common elements shared by the network gaming community.
- **Transaction Security** Mechanisms for securing e-commerce transactions, for example, digital signatures and

certificates, as well as methods for reliable implementation of distributed transactions.

- **Encryption** Prevents many hacker attacks intended to disrupt or defraud gamers. Protects privacy of players and supports security of e-commerce applications.
- **Regulation, Insurance and Oversight** A third party independent verification of all aspects of game software at the source code level needs to be implemented. Regulation agencies similar to those in charge of overseeing brick and mortar casinos may need to be set up.

Additional general approaches applicable to the broader field of network security not just online game security are suggested by Yan et al. (Yan and Choi 2002):

- **Built-in cheating detection** An intrusion detection system may be seamlessly incorporated into the game software. Recently a system has been proposed based on style of play utilized by the player to capture intruders (Yampolskiy 2006; Yampolskiy and Govindaraju 2006).
- **Make players security-aware** Players need to be educated about potential security threats for example phishing.
- **Good password practice and management** Users should be educated about what constitutes a secure yet easy to remember password (Yampolskiy 2006).
- **Fair trading** Third parties should be involved in exchange and sale of virtual assets.
- **The bug patching approach** Software flows should be addressed by the developer as they are discovered and patched in a timely manner.
- **An active complain-response channel** A way for players to report bugs and suspected cheating behavior should be provided. A quick response from the game administration is essential to insure players' enthusiasm.
- **Logging and audit trail** Accurate records help to combat insider cheating as well as help in investigation of suspected cheating cases.
- **Post-detection mechanisms** Mitigation of damages from cheaters is an important step in maintaining fairness of the game environment. Cheaters should be punished and victims restored to their original state.

Kimppa et al. (Kimppa and Bissett 2005) proposed the countermeasures presented in Table 2 for the acts of cheating encountered in first person shooters, strategy games and role playing games.

Since collusion is probably the most frequently used way to cheat in online games we pay particular attention to techniques used to mitigate this form of cheating. Randomized partnering is probably the best known solution to avoid collusion and is most frequently used in online poker tournaments, however it is not helpful in small single table games (Yan 2003).

Since we are not likely to fully avoid collusion in online poker rooms the next best thing we can do is try to detect it. By evaluating every decision made by the player we can determine if information which is not supposed to be available to the player is being utilized to make a decision. While a single decision may not be sufficient to even suspect collusion a series of optimal actions may provide sufficient proof (Vallve-Guionnet 2005). Additionally, datamining tools can be used to discover players who always tend to play together or who win more frequently in the presence of particular players. Manually analyzing particular hands with private information about player's hands may also allow us to confirm cases of collusion and is frequently done after complains are received from suspicious players at the same table.

Table 2: Countermeasures for computer games

Cheat	Countermeasure
Camping (reserving a spot which is ideal for spotting and killing unsuspecting players)	Kicking the player out after a certain time interval.
Non-Stop jumping to make aiming difficult	Preventing sequential jumping over a certain number of jumps at the server level.
Wallhacks (seeing through walls)	Divide map into chunks to hamper the use of the map information.
Reflex augmentation	Kill the process for the well known cheating software
Aim bots	Kill the process for the well known cheating software
Enhanced damage by compromised client	Use checksums
Raw materials which do not belong to the player	Turn beta testing features off
Map revealing software	Don't send unnecessary information to the player
Fake messages from the sever administration	Forbid account names which resemble administrator accounts
Item duplication or creation	Kill the process for the well known cheating software

CONCLUSIONS

With the steady increase in popularity of games and services offered via the Internet the problem of securing such services from automated attacks became apparent. In this paper we have reviewed cheating methods frequently applied to online computer games alongside approaches used to combat such occurrences. In particular we concentrated on cheating approaches observed at online poker tables.

REFERENCES

Arkin, B., Hill, F., Marks, S., Schmid, M., Walls, T. J., and McGraw, G. "How We Learned to Cheat in Online Poker: A Study in Software Security." *Developer.Com*.

Banavar, H. "Security issues in Multi-player,Distributed Network Games." Available at: ww2.cs.fsu.edu/~banavar/research/NSPaper.htm.

Baughman, N. E., Liberatore, M., and Levine, B. N. "Cheat-Proof Playout for Centralized and Severless Online Games." *IEEE/ACM Transactions on Networking*.

Brooke, P. J., Paige, R. F., Clark, J. A., and Stepney, S. (2004). "Playing the game: cheating, loopholes, and virtual identity." *ACM SIGCAS Computers and Society*, 34(2).

Chambers, C., Feng, W. C., Feng, W. C., and Saha, D. "Mitigating Information Exposure to Cheaters in Real-Time Strategy Games." *Proceedings of NOSSDAV*.

Chen, B. D., and Maheswaran, M. "A cheat controlled protocol for centralized online multiplayer games." *Proceedings of 3rd ACM SIGCOMM workshop on Network and system support for games*, Portland, Oregon, USA, 139 - 143.

Chen, W., and Chen, M. "Internet Game Security." Available at: http://islab.oregonstate.edu/koc/ ece478/03Report /wtchen_mtchen.pdf.pdf.

Consalvo, M. "Gaining Advantage: How videogame players define and negotiate cheating." *Changing Views: Worlds in Play, second annual conference of the Digital Games Research Association*, Vancouver, British Columbia.

Davis, S. B. "Why cheating matters: Cheating, game security, and the future of global on-line game business." *Proceedings of the 2001 Game Developers Conference*.

Foo, C. Y. "Redifining Grief Play." *Other Players Conference*, Copenaghen, Denmark.

Foo, C. Y., and Koivisto, E. "Grief Player Motivations." *Other Players Conference*, Copenaghen, Denmark.

Hayes, E. "Playing it Safe: Avoiding Online Gaming Risks." Available at: http://www.us-cert.gov/reading_room/gaming.pdf.

Kabus, P., Terpstra, W., Cilia, M., and Buchmann, A. "Addressing Cheating in Distributed Massively Multiplayer Online Games." *In Proc. of Intl Workshop on NetGames*.

Kimppa, K. K., and Bissett, A. K. (December 2005). "The Ethical Significance of Cheating in Online Computer Games." *Inter. Review of Information Ethics*, 4.

Kuecklich, J. "Other Playings - Cheating in Computer Games." Available at: http://itu.dk/op/papers/kuecklich.pdf.

Kuo, A. "A (very) brief history of cheating." Available at: http://shl.stanford.edu/Game_archive/StudentPapers/ BySubject/A-I/C/Cheating/Kuo_Andy.pdf.

Li, K., Ding, S., McCreary, D., and Webb, S. "Analysis of State Exposure Control to Prevent Cheating in Online Games." *ACM Nossdav*, Kinsale, Ireland.

Lyhyaoui, Y., Lyhyaoui, A., and Natkin, S. "Online Games: Categorization of Attacks." *The International Conference on Computer as a Tool (EUROCON 2005)*, 1340- 1343.

Mørch, K. "Cheating in online games- threats and solutions." *Publication No: DART/01/03. Norwegian Computing Center/Applied Research and Development*.

Poker-edge.com. "Stats and Analysis." Available at: http://www.poker-edge.com/stats.php.

Pokerprophecy. Available at: http://www.pokerprophecy.com.

Pritchard, M. (February 2001). "How to Hurt the Hackers: The Scoop on Internet Cheating and How You Can Combat It." *Information Security Bulletin*.

Sicart, M. "On the Foundations of Evil in Computer Game Cheating." *roceedings of the Digital Games Research Association's 2nd International Conference - Changing Views: Worlds in Play*, Vancouver, British Columbia, Canada.

Smith, J. H. "Playing dirty - understanding conflicts in multiplayer games." *5th annual conference of The Association of Internet Researchers*, The University of Sussex.

Snyder, A. (August 1, 2006). *The Poker Tournament Formula*, Cardoza.

Vallve-Guionnet, C. "Finding colluders in card games." *International Conference on Information Technology: Coding and Computing (ITCC 2005)*, 774-775.

Webb, S. "A Survey of Cheating Techniques in Online Games." Available at: http://home.cc.gatech.edu/ webb/uploads/10/MiniProject3.pdf.

www.playnoevil.com. "Cheating at Online Poker? A Detailed Analysis." *Play No Evil Game Security*, Available at: http://www.playnoevil.com/serendipity/index.php?/archives/772-Cheating-at-Online-Poker-A-Detailed-Analysis.html.

www.tips4poker.com. "Online Cheating." Available at: tips4poker.com/content/online_poker_cheating.html.

Yampolskiy, R. V. "Behavior Based Identification of Network Intruders." *19th Annual CSE Graduate Conference (Grad-Conf2006)*, Buffalo, NY.

Yampolskiy, R. V. "Analyzing User Password Selection Behavior for Reduction of Password Space." *The IEEE International Carnahan Conference on Security Technology (ICCST06)*, Lexington, Kentucky.

Yampolskiy, R. V., and Govindaraju, V. "Use of Behavioral Biometrics in Intrusion Detection and Online Gaming." *Biometric Technology for Human Identification III. SPIE Defense and Security Symposium*, Orlando, Florida.

Yan, J. "Security design in online games." *Proceedings of the 19th Annual Conference on Computer Security Applications*, Washington, DC, USA, 286- 295.

Yan, J., and Randell, B. "A systematic classification of cheating in online games." *Proceedings of 4th ACM SIGCOMM workshop on Network and system support for games (NetGames '05)*, Hawthorne, NY, 1-9.

Yan, J., and Randell, B. "Security in Computer Games: from Pong to Online Poker." *Techical Report #889, School of Computing Science, Newcastle University*, Available at: http://www.cs.ncl.ac.uk/research/ pubs/trs/papers/889.pdf.

Yan, J. J., and Choi, H. J. "Security issues in online games." *The Electronic Library*, 125-133.

MOVE ORDERING VS HEAVY PLAYOUTS: WHERE SHOULD HEURISTICS BE APPLIED IN MONTE CARLO GO?

Peter Drake
Lewis & Clark College
Department of Mathematical Sciences, MSC 110
0615 SW Palatine Hill Road
Portland, OR 97219
drake@lclark.edu

Steve Uurtamo
SUNY Buffalo State
317 Bishop Hall
1300 Elmwood Ave
Buffalo, NY 14222
uurtamo@gmail.com

ABSTRACT

Writing programs to play the classical Asian game of Go is considered one of the grand challenges of artificial intelligence. Techniques used in a strong Go program would likely be applicable to other games and to non-game domains. Traditional game tree search methods have failed to conquer Go because the search space is so vast and because static evaluation of board positions is extremely difficult. There has been considerable progress recently in using Monte Carlo sampling to evaluate moves. Such programs can be strengthened further by adding domain-specific heuristics. This paper addresses the question of where such heuristics are most effectively applied: to modify the order in which moves are considered or to bias the sampling used to evaluate those moves. Experimental results are presented for three heuristics. We conclude that the sampling "playouts" are the most effective place to apply heuristics.

KEYWORDS

artificial intelligence, Go game, Monte Carlo, UCT, heuristics

INTRODUCTION

Writing programs to play the classical Asian game of Go is widely considered to be a grand challenge of artificial intelligence (Bouzy and Cazenave 2001). While the best Chess programs are on a par with human world champions, the best Go programs still play at the amateur level, at least on the standard 19x19 board.

The traditional approach to computer game play has been minimax search with enhancements such as alpha-beta pruning (Russell and Norvig 2003). Since the number of possible games of Chess or Go is astronomically large (Bouzy and Cazenave 2001), it is not feasible to search the entire game tree. Chess programs therefore rely on domain-specific heuristics for move ordering (considering heuristically "good" moves first and often omitting others) and static evaluation (estimating the value of a board configuration before the game is over). This minimax approach has not been successful for Go. Because the branching factor is much higher in Go, there is a greater risk that a move ordering heuristic might omit good moves. More importantly, because "dead" stones can remain on the board for hundreds of moves before they are actually removed, static evaluation is extremely difficult.

Several computer Go programmers, including the authors, have recently begun exploring statistical sampling approaches called Monte Carlo Go (Brügmann 1993, Coulom 2006, Gelly *et al.* 2006, Drake and Uurtamo 2007). Monte Carlo Go programs respond to a given game situation by randomly completing the game thousands of times, then choosing the move that has led to the best average outcome.

While the Monte Carlo approach has not yet produced 19x19 Go programs at the master level, it has produced startlingly strong 9x9 programs. At the 19x19 level, Monte Carlo programs are now at the level of the strongest traditional programs. At the 12th Computer Olympiad, the top two finishers in the 19x19 competition (MoGo and Crazy Stone) were both Monte Carlo programs; neither lost a single game to any other program.

Because Go is such a rich domain, it is likely that techniques that fare well in Go will be applicable to other domains. Monte Carlo search techniques have been applied to other games (Chung *et al*. 2005) and to non-game domains (Chaslot et al, 2006).

This paper begins with a brief summary of the rules of Go. The Monte Carlo approach to Go is then described. We explain several places where heuristics might be applied within this framework. Three specific heuristics are then presented, followed by our experimental results. The paper closes with conclusions and a discussion of future work.

THE RULES OF GO

While the rules of Go are considerably simpler than those of Chess, space permits only a brief overview here. Interested readers are directed toward any of the many excellent tutorials available in print or on-line, such as (Baker 1986).

Go is a two-player game of perfect information. The board consists of a square grid of intersecting lines; traditionally the grid is 19x19, but smaller boards are sometimes used for teaching new players or for computer Go research. The two players, black and white, each have a supply of stones in their color. Starting with black, the players alternate taking turns. A turn consists of either passing or placing a stone on an unoccupied intersection of two lines. The game ends when both players pass consecutively. Each player scores one point for each intersection that is either occupied or surrounded by his or her stones.

Three additional rules give the game its profound strategic depth. First, a block of contiguous stones is captured (removed from the board) if it is tightly surrounded, i.e., if there are no vacant intersections adjacent to it. Second, a player may not directly cause his or her own stones to be captured. Third, it is illegal to repeat a previous board position.

MONTE CARLO GO

In general, Monte Carlo approaches to game tree search are similar to traditional minimax search, but with statistical sampling used for position evaluation. For example, a program might do a full search to some fixed depth, with the leaves of the search tree evaluated by sampling. To evaluate a leaf position, random moves are played until the end of the game and the score is recorded. The board is then reset to the leaf position and random moves are again played to the end of the game. After many of these playouts, the portion of games won by each side gives a reasonable approximation to the value of the position.

A further refinement of Monte Carlo Go is to focus the sampling on more promising moves. This introduces a tradeoff between *exploration* (sampling the result of each potential move enough times to have an accurate estimate of its value) and *exploitation* (spending the most samples on the moves that have been most successful so far). The strongest extant Monte Carlo programs use the UCT algorithm (Kocsis and Szepesvári 2006, Gelly *et al*. 2006). UCT stands for "Upper Confidence bounds applied to Trees". When choosing a move to sample, UCT chooses the move with the greatest sum of value (portion of won games) and uncertainty. It will therefore always choose either a move that has a good winning record or a move that has not been sufficiently explored. As the sampling proceeds, the uncertainty shrinks and UCT spends most of its time improving its estimates of the best moves. Heavily-sampled tree nodes are expanded, so the search is deepest along the most promising line of play. The algorithm is therefore able to make a confident statement about which move is really best.

APPLYING HEURISTICS

A Monte Carlo Go program can be made even stronger by adding domain-specific knowledge. A number of papers have been published showing the benefits of various heuristics, some of them quite sophisticated. There is some variety in where the heuristics are applied. The options may be divided into two major categories:

- *Move ordering* uses a heuristic to decide the order in which moves are considered within (or at the fringe of) the search tree.

- *Heavy playouts* use a heuristic to bias the "random" moves in the sampling playouts, so that the results will more closely approximate a realistic game.

It is not obvious *a priori* which of these approaches is superior. Move ordering has its effect closer to the root of the tree, which is more relevant to the decision at hand. Furthermore, move ordering only requires computing the heuristics for a very small number of board positions. Heavy playouts require that the heuristics be evaluated for every move of the game. On the other hand, having more accurate playouts may be worth the price of completing fewer playouts in the allotted time. Heavy playouts certainly are better asymptotically, in that if the heuristic were perfect (suggesting only the single best move), only one sample would be needed from each tree node. At the same time, an imperfect heuristic may destroy the benefit of Monte Carlo sampling by exploring an unreasonable subset of the space of possible game completions.

Heuristic Application in Previous Work

Chaslot *et al*. 2007 use move ordering, modifying the UCT formula to take heuristic values into account for sparsely-sampled moves. They also delay exploration of moves with very low heuristic values, which they refer to as "progressive unpruning".

Coulom 2007 uses a very similar technique under the name of "progressive widening". Coulom also uses heavy playouts, producing a probability distribution over legal moves.

Drake and Uurtamo 2007 use heavy playouts. For each move in a playout, a heuristic is chosen with some probability; otherwise a random move is chosen.

Gelly *et al*. 2006 use a different version of heavy playouts, choosing a heuristicially-suggested move if there is one and playing randomly otherwise. The heuristics therefore allow "urgent" moves to be pounced on without slowing down the playouts too much. They also use move ordering and another progressive widening scheme.

Gelly and Silver 2007 use heavy playouts with a probability distribution, but with noise introduced in a variety of ways. They also use move ordering by initializing the value of a new leaf according to heuristics, effectively pretending that the leaf had already been sampled.

Cazenave 2007 uses heavy playouts with a probability distribution.

Our Heuristic Application Techniques

In the experiments in this paper, we apply heuristics in three different ways: move ordering, best-of-n, and first-n.

Our *move ordering* technique treats untried moves at the fringe of the tree as having a UCT value of $1.0 + ch$, where c is a small constant and h is the heuristic value of the move. The untried moves are therefore tried in the order suggested by the heuristic. An untried move is preferred to a move that has been tried once unless the tried move has a very high UCT value, e.g., it has led to a win every time it has been tried. This is consistent with setting the first-play urgency to 1.0 as described in Gelly *et al*. 2006.

Our *best-of-n* technique is a heavy playout variation that attempts both to remedy the high cost of computing heuristics during playouts and to avoid over-biasing the sampling. At each

move during a playout a fixed number (n) of random legal moves are chosen. The move with the highest heuristic rating is chosen.

Our *first-n* technique is another heavy playout variation where, for the first n moves beyond the fringe of the tree, the move rated highest by the heuristic is chosen. After this point the playout is completed randomly.

Three Heuristics

Our experiments involve three different heuristics: the capture heuristic, the fighting heuristic, and the pattern heuristic.

The *capture* heuristic simply attempts to capture enemy stones whenever possible. Moves that capture more stones are rated higher.

The *fighting* heuristic, at an abstract level, plays moves in areas of the board where fights are taking place. Recall that a block of stones is removed if there are no vacant points adjacent to it. These vacant points are called liberties. We define a pseudo-liberty to be a graph edge leading from a stone to an adjacent liberty. The number of pseudo-liberties a block has is at least equal to the number of liberties it has, but may be slightly higher if there are multiple edges leading to the same liberty. Since counting pseudo-liberties is easier than counting liberties, our board-handling code uses pseudo-liberties to determine when a block has been captured. Pseudo-liberty counts are therefore available for use in a heuristic without additional computation. A group with a small number of pseudo-liberties is in some danger of being captured, i.e., is in a fight. The fighting heuristic prefers moves adjacent to blocks (friendly or enemy) that have few pseudo-liberties.

The *pattern* heuristic involves a table of patterns. Each pattern represents a 3x3 region of the board, with each point marked as either black, white, vacant, or off-board (which happens if the center point is at the edge or corner of the board). Associated with each pattern is a weight indicating the value of playing in the center.

Our patterns were automatically extracted from a database of 3199 9x9 games provided by Nici Schraudolph. Since the players of these games were much stronger than random players, we wanted our patterns to suggest playing similar moves. Each game was run through in each of the eight possible rotations and reflections. For each move in each game in the database, the pattern around the move was given a “win”. Each other pattern that appeared elsewhere on the board was given a “loss”. Any pattern with less than 100 wins was discarded, leaving 1639 patterns. The value of each pattern was set to the ratio of wins to (wins + losses) for that pattern.

RESULTS

Experiment 1: Direct Comparison of Heuristics

As a quick check of the quality of our heuristics, we ran a tournament between different heuristics. Every move in these games was decided by direct application of the heuristics, with ties between heuristic values decided randomly. Orego version 5.05 was used.

There were four contestants: random (no heuristic), capture, fighting, and patterns. Each contestant played 1000 9x9 games against each other contestant, 500 as black and 500 as white. Komi was set at 7.5.

Random won 739 games, capture 1224, fighting 1514, and patterns 2523. This suggests that all three heuristics are useful, but patterns is the strongest and capture the weakest. This experiment did not, however, take into account the time required to compute the various heuristics. The next experiment addresses that issue.

Experiment 2: Time Test

In this experiment, we used four instances of Orego, each using as a heuristic one of the four contestants from the previous experiment. The heuristic was applied using best of 4 heavy playouts. This experiment was run on a Mac Pro with two dual-core 3 GHz Intel Xeon processors and 2 GB of RAM.

Orego was given 10 seconds to search for the best move on a 9x9 board. The random heuristic completed 308,112 playouts, the capture heuristic 225,483, the fighting heuristic 269,760, and the pattern heuristic 235,155.

We repeated this experiment with best of 81 heavy playouts. The capture heuristic completed 77,054 playouts, the fighting heuristic 85,360, and the pattern heuristic 59,956. (The random heuristic would not be affected by this change.)

In both cases the fighting heuristic is the fastest, but the order of the other two is not consistent. No heuristic is twice as fast as another, so we do not expect the speed of the heuristics to be a major factor in the next experiment.

Experiment 3: Heuristic Application Within UCT

This is the central experiment of this paper. We pitted Orego against GNU Go (version 3.6), a widely-available Go program with a conventional (non-Monte-Carlo) architecture. Each condition involves a particular heuristic applied in a particular way. In each condition, 320 9x9 games were played against GNU Go (160 as black, 160 as white). Komi was set at 7.5. These experiments were run on Athlon64 3200+ 800FSB RT CPUs running Linux, with Orego allowed 15 seconds to think for each move.

Figure 1 shows the results for the capture heuristic. This heuristic provides its greatest benefit when applied in best of 2 heavy playouts. Best-of-n for large values of n is worse than no heuristic at all, presumably because the increase in the quality of the playouts does not make up for the decrease in the number of playouts.

The fighting heuristic (Figure 2) again performs best in the best-of-n mode. Intriguingly, it performs better with larger values of n. With first-n, we again see a decreasing trend. These two trends are somewhat contradictory, since for sufficiently

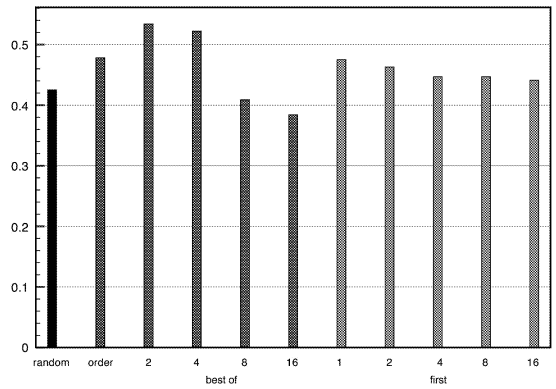


Figure 1: Results for the capture heuristic. The vertical axis shows the portion of games won against GNU Go in each condition. The best-of-2 and best-of-4 performances are significantly better than random (no heuristic) playouts according to a one-tailed T-test, $p < 0.01$ and $p < 0.02$ respectively.

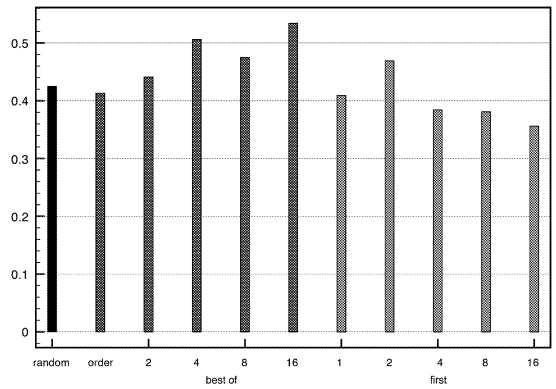


Figure 2: Results for the fighting heuristic. Best-of-4 and best-of-16 are significantly better than random playouts, $p < 0.05$ and $p < 0.01$ respectively.

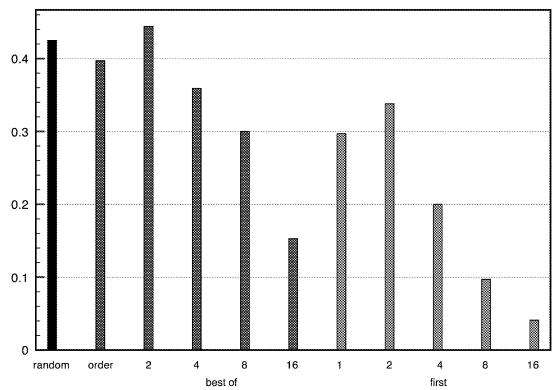


Figure 3: Results for the pattern heuristic. None of the conditions are significantly better than random playouts.

large values of *n* these techniques are identical (on every move in the playout, choose the move with the highest heuristic value).

Given the results of the previous experiments, we were surprised to see that the pattern heuristic weakens Orego in almost every condition (Figure 3). These results do, however, provide further evidence that heavy playouts (specifically the best-of-*n* technique) are the best use of a heuristic.

CONCLUSIONS AND FUTURE WORK

We have reviewed previous research on applying heuristics to UCT Go programs, with an emphasis on where heuristics are applied. Two major categories of heuristic application were delineated: move ordering and heavy playouts. Our experiments, using three different heuristics, showed that heuristics are most effectively applied in heavy playouts. More specifically, a best-of-*n* heavy playout technique seems more effective than a first-*n* technique.

Two of our heuristics (capturing and fighting) proved successful, while the other (patterns) did not. We speculate that the pattern heuristic helps a player *get into* a good position, while the others help a player *take advantage of* a good position. In experiment 1, the capture and fighting heuristics were not at their strongest because they didn't have good positions to take advantage of. In experiment 3, UCT achieved a fairly good position before the playouts began, so capturing and fighting had something to pounce on while the patterns didn't help much. In any case, direct games are not a reliable way of comparing heuristics.

Our results should prove useful in future Monte Carlo programs for Go and for other domains. Knowing where heuristics are best applied should save time in evaluating new heuristics. It may also inform architectural decisions, e.g., encouraging incremental updating of heuristic evaluations so that they may be used throughout the long playouts.

ACKNOWLEDGMENTS

We thank Nici Schraudolph for providing us with recorded game data and the members of the Computer Go Mailing List for their many helpful comments.

REFERENCES

Baker, K. *The Way to Go*. 1986. New York, NY: American Go Association.

Bouzy, B., and Cazenave, T. 2001. Computer Go: an AI Oriented Survey. *Artificial Intelligence* 132(1):39-103.

Brügmann, B. 1993. Monte Carlo Go. Unpublished technical report.

Cazenave, T. 2007. Playing the Right Atari. International Computer Games Association Journal 30(1):35-42.

Chaslot, G., De Jong, S., Saito, J.-T., and Uiterwijk, J. W. H. M. 2006. Monte-Carlo Tree Search in Production Management Problems. In Pierre-Yves Schobbens, Wim Vanhoof, and Gabriel Schwanen, editors, *Proceedings of the 18th BeNeLux Conference on Artificial Intelligence*, Namur, Belgium, pages 91-98.

Chaslot, G.M.J.B., Winands, M.H.M., Uiterwijk ,J.W.H.M., van den Herik, H.J., and Bouzy, B. 2007. Progressive strategies for

Monte-Carlo tree search. Draft, submitted to JCIS workshop 2007.
Chung, M., Buro, M., and Schaeffer, J. 2005. Monte Carlo Planning in RTS Games, *CIG* 2005, Colchester,
Coulom, R. 2006. Efficient selectivity and backup operators in Monte-Carlo tree search. In Proceedings of the 5th International Conference on Computers and Games. Turin, Italy.
Coulom, R. 2007. Computing Elo ratings of move patterns in the game of Go. Draft, submitted to ICGA Computer Games Workshop 2007.
Drake, P., and Uurtamo, S. 2007. Heuristics in Monte Carlo Go. In *Proceedings of the 2007 International Conference on Artificial Intelligence*, CSREA Press.
Gelly, S. and Silver, D. 2007. Combining online and offline knowledge in UCT. In International Conference on Machine Learning, ICML 2007.
Gelly, S., Wang, Y., Munos, R., and Teytaud, O. 2006. Modification of UCT with patterns in Monte-Carlo Go. Technical Report 6062, INRIA, France.
Kocsis, L. and Szepesvári, Cs. 2006. Bandit based Monte-Carlo Planning. In *Proceedings of the 17th European Conference on Machine Learning*. Springer-Verlag.
Russell, S., and Norvig, P. 2003. *Artificial Intelligence: A Modern Approach*, second edition. Upper Saddle River, NJ: Prentice Hall.

BIOGRAPHY

Peter Drake received a BA in English from Willamette University in Salem, Oregon, a MS in Computer Science from Oregon State University, and a PhD in Computer Science and Cognitive Science from Indiana University. He is Assistant Professor of Computer Science at Lewis & Clark College in Portland, Oregon. He is the author of *Data Structures and Algorithms in Java*.

GAME AGENTS

Using artificial neural networks for "common sense" simulation in videogame agents*

A. Barella
J. Fabregat
C. Carrascosa
Dept.de Sistemas Informáticos y Computación
Universidad Politécnica de Valencia
Campus de Vera
46022 Valencia
{tbarella,jfabregat,carrasco}@dsic.upv.es

July 27, 2007

Abstract

Artificial Intelligence plays nowadays a fundamental role in videogames and virtual reality applications. Agents interacting with humans must have a credible visual appearance and must mantain user immersion. They need to know enough about their environment as the user feels it would be "common sense" for an inhabitant of the environment. Deliberation can't be applied for each of these decisions because a videogame agent has time restrictions that prevent so. If the simulation cycle is delayed, the player immersion is lost. A cheap and quick solution has to be applied that offers enough quality for common decisions, sort of a "common sense" for the agent. This paper presents a system integrating sample generation with the training of an artificial neural network for the resolution of a particular case. The prediction of the outcome of a confrontation in order to choose the most auspicious action in each situation.

*This work is partially supported by the Spanish government and FEDER funds under TIN2006-14630-C0301 project.

1 Introduction

Artificial Intelligence plays nowadays a fundamental role in videogames and virtual reality applications (VRA) [Rabin 2002]. Agents interacting with humans must have a credible visual appearance, but also a behaviour that mantains user immersion [Maes 1995]. Agents in both videogames and VRA need to know what the user feels it would be "common sense" for an inhabitant of the environment. Programming all that necessary knowledge is a large amount of work that has to be redone even after minor modifications.

VRAs and computer game development cover many different areas, such as graphics, artificial intelligence and network communications. Nowadays, players demand more sophisticated and credible computer-controlled opponents, but results are sometimes unsatisfactory. This is because of the need of real-time processing constraints to reach a satisfying feeling of immersion for the user. Moreover, to get an illusion of credible behaviour, the artificial intelligence part has to be updated at the same frame rate than graphic part does. Visualization, intelligence and even physics and sound have to be done

in the same simulation cycle. If one part delays, all simulation cycle is delayed, getting a poor result [Garcia 2004].

In this paper, it is proposed to combine an artificial neural network with an intelligent agent. In the bibliography the main cause to combine neural networks and agents in videogames is the ability of agents to adapt [Stanley 2005],[Charles 2004]. Adaptivity is an interesting feature, but it requires a previous development, since you can't deliver a completely untrained agent in a videogame. Even those in which the training is a major feature ([Lionhead 2001], [Nintendo 2005]) need an initial behaviour. Among the most similar aproaches to the one described in this paper it can be cited [Rabin 2002], an application of pattern recognition to the classification of an input vector, that could be used for high-level decision making.

Choosing the right action during execution usually requires the use of time-consuming AI-techniques, unacceptable given the restrictions both in response time and computation resources devoted to the AI. Instead, the use of an artificial neural network previously trained for choosing the best action is proposed. During the execution the time cost of using a trained neural network is cheap and constant.

The use of an artificial neural network, however, needs a large corpus from which to train, and extracting it from real executions of the environment has a very high cost. Since the application isn't set in the real world, a perfect model can be asumed. The solution presented in this paper for obtaining the corpus is, thus, a mix of sample generation and an heuristic computation of the outcome. It is proposed the use of this integrated system of sample generation with the training of an artificial neural network. Both generation and training are automated then for a set of parameters of the problem. This is important because, should the parameters be changed, the regeneration and retraining of the neural network involves no aditional programming further than the changing of these parameters, remaining the time cost during execution constant.

2 Problem Description

The purpose of this paper is to provide an agent situated in a videogame setting with common sense knowledge about its environment. Specifically it must be able to foresee the outcome of a confrontation between itself and several enemies from the videogame, choosing the most auspicious action. A videogame agent has some time restrictions that prevent using a deliberative solution. Being a cheap, quick and experience-based decision, what is needed to solve this problem is some kind of common sense reasoning of the agent about its own environment. The problem chosen to be solved involves high-level decision taking with uncertainty and incomplete knowledge, revising the solution for each new piece of information received, so it is a commonsense reasoning problem[Mueller 2006]. Even being an specific case, it is generalizable to other videogames and has enough complexity to prove the usefullness of the solution.

Two agents engage in combat over a discrete 60×60 squares board, starting in random positions and at least one of them seeing the other. Each one of them chooses, in complementary turns, the orientation, movement (both x and y) and whether to attack. There are several kinds of agents, all with the same set of traits but each one with a given range of values. Many of this traits are real numbers, but even discretizing by intervals there are a high number of cases. The traits of each agent are: *Type*, *Size*, *Direction* and *Weapon* taking enumerated values, and $\{Life, Protection, Experience\} \in \Re$.

During the course of a combat, an agent has access to the full information about his own state, but from his opponent he can only know the kind, size, weapon and the combat skills. This is, so, an imperfect -or partial- information problem, specifically, it is a decision making with uncertainty problem. For solving this family of problems there are many solutions in the bibliography [Russell 1995], like Decision Trees, Finite State Machines and MiniMax Trees.

3 Proposed Solution

Evaluating the different approaches, it's not easy to find a good combination of reusable code and time cost. Solutions are either non reusable and hard to program solutions or heuristic searching in the state space, and this involves an unacceptable execution time cost. Examples of the first approach are decision trees and finite state machines, i.e. solutions that bound the resolution data into the algorithm, so changing the data involves reprogramming. An example of the second one is the MiniMax algorithm. Even with pruning and refining of the algorithms the cost is unacceptable with the time restrictions found in a videogame or a VRA.

A solution that solves the problems of boths approaches would requiere an algorithm with a low time cost and which wouldn't include the data. The workaround proposed is the use of an artificial neural network. A trained neural network has a low and bounded time cost and efficient neural network programming is a well documented field [Nilsson 1996][Ripley 1996]. Changes in data means offline retraining, not code rewriting.

Artificial neural networks use to need a large corpus for training. Extracting it from real executions of the system is usually very time-consuming. The solution proposed in this paper is training a neural network with a set of samples uniformly distributed in the state-space. Samples are generated by means of computing their outcome by a MiniMax algorithm using heuristic knowledge. During sample generation there are not the same time constraints as during the execution. MiniMax tree can be explored to an acceptable depth, thus obtaining a reliable result. However, the complete calculations of each combat for the sample generation would take much more to be practical, and it would be worse had the game a random component in the combats. With this sample corpus heuristically computed the neural network is trained. Being an artificial environment the heuristics can be adjusted since there is an accurate model of the system.

In case it's needed changing some parameters from several elements in the environment (e.g. attack / defense values, or even adding different enemies), the code of the agent remains the same (not code rewriting at decision-making). The corpus is regenerated and used to retrain the neural network. The change would only affect offline computing time. During execution, the agent uses the same neural network, trained with other data. As the neural network does not change, its execution time cost remains the same, something pretty interesting in the videogames and VRAs.

4 Implementation and Results

4.1 Data generation

The first part of the solution is the generation of the samples. A C++ program has been implemented that randomly generates a sample and then, using the MiniMax algorithm with depth 8 it computes the better move for the active agent, both movement and attack decisions. The data is organized in each sample in two groups, the data of the active agent and the visible data of his opponent. Common data of both the agent being trained and the opponent:

- Personal data: *Type* $\in$ {Barbarian, Skeleton, Orc, Giant} and *Size* $\in$ {Small, Medium, Large}.
- Last Action: *Direction* $\in$ {North, South, East, West }, *Movement* in {X, Y} $\in$ [-2 .. 2] and *Attack* $\in$ boolean.
- Combat Data: *Weapon* {Dagger, Sword, Lance}, *Probability of Succesful Attacks* $\in$ [0 .. 1] and *Damage Done* $\in$ [0 .. 1].

Specific data of the agent being trained: {*Life*, *Protection*, *Experience*} $\in \Re$. Output data is an action composed of: *Direction* $\in$ {North, South, East, West }, *Movement* in {X, Y} $\in$ [-2 .. 2] and *Attack* $\in$ boolean.

The corpus has 2236 samples, which are divided in three sets of training (60% of the data), validation (10%) and test (30%).

4.2 Implementation

Since the goal was testing the viability of the use of neural networks for this problem, several topolo-

gies and algorithms were tried, tuning their parameters. The better algorithms were BackPropagation and BackPropagation with momentum (with different values of η and μ). They have been tried out several topologies for designing the neural network: both one and two hidden layers. Moreover, some networks have beeen designed in order of the organization of the input data. The validation method was the hold out method.

Before feeding the neural network with the data, all parameters were normalized and converted to the input format of the neural network tool to be employed, the Sttutgard Neural Network Simulator (SNNS) [Zell 1995], at the end of the experiments the normalization is reverted to evaluate the data.

4.3 Results Analysis

It will be used the MSE (Medium Square Error) of each topology to compare them.

The lesser MSE corresponds to a network with two hidden layers (10-08), with the BackPropagation algorithm, parameter $\eta = 0.2$. As the results need to be denormalized , the three better performers were taken to see how they perform in the real example. The two networks designed following the intuitive structure of the input data have been among the highest MSE.

Topology	Direction	MovX	MovY	Attack	Global
BP1h20	92,70%	96,13%	95,08%	91,80%	79,73%
BP2h10-08	93,59%	95,98%	95,38%	90,91%	80,48%
BPM1h60	93,59%	95,23%	94,93%	92,10%	79,43%

Table 1: Percentage of success in each parameter for the three networks with the lowest MSE

The best performer after denormalizing was the same network as before, as seen in table 1. The global percentage of error is of 19.52%, it means that at least one of the four parameters fails that many times. In the figure 1 it is shown the error broken down to see how many dimensions fail at once: 14,90% of the error is due to error in one dimension, 4.62% are error in two dimensions at once and there is no error due to more than two dimensions at once.

Analysing figures 2 and 3, and taking into account the movement in X and Y, it can be observed that

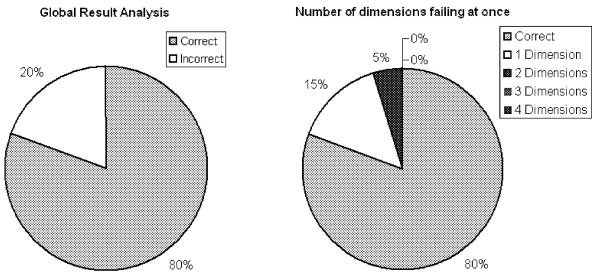


Figure 1: Global error and interference

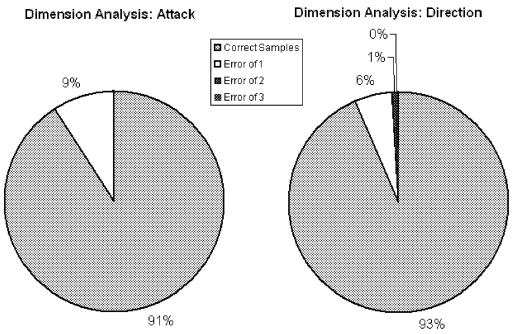


Figure 2: Analyse of the four dimensions: *Attack* and *Direction*

almost all error is spread in one standard deviation, what could be seen in the videogame as moving in the right direction. Looking at the orientation most of the error is in the first standard deviation also, but being orientation discretized in four values, probably it would look worse than the movement; at least the error is of 6,41% and usually the agent won't end up looking the other way. In the attack, which is a binary dimension, with an error of 9.09% , the agent would attack when it is not advisable once in a while, or won't when it should.

5 Conclusions

Applying neural networks to high-decision processes in videogames or VRAs seems a solution with a low and constant time cost, and efficient implementations

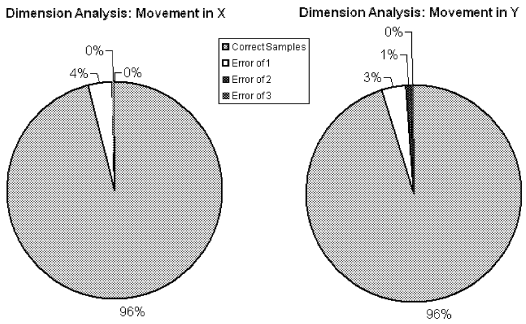


Figure 3: Analyse of the four dimensions: movement in X and Y axis

are well documented [Nilsson 1996]. Automatic generation of the corpus, and the integration with the training offers a cheap and easy way to adapt the agent to modifications in the parameters of the application. The data shows us the viability of combinating an heuristic sample generator and an artificial neural network. Execution time is low and predictable and the error rate is acceptable. The point of view of the authors is that an small error rate would increase the credibility of the agent. An always perfect performance would be frustrating, and in a videogame the artificial intelligence is targeted at the entertainment of the user, not at optimality. As in videogames there is an interest for a scalable difficulty level of the artificial intelligence, several neural networks could be trained, testing the results and asigning the level of difficulty given their performance. Among the future work it is projected to apply the results with JGOMAS [Barella 2006], a framework that integrates a multiagent system in a virtual environment. This framework provides an upper layer where inteligence has to be included, and this is where the neural network described in the present paper will be applied, to test the scalability of the solution to a more complex case.

References

[Barella 2006] Barella, A. ; Carrascosa, C., and Botti, V. 2006. Jgomas: Game-oriented multiagent system based on jade. In *ACM Sigchi International Conference on Advances in Computer Entertainment Technology*, 1–8.

[Charles 2004] Charles, D., and Black, M. 2004. Dynamic player modelling: A framework for player-centric digital games. In *Proceedings of the International Conference on Computer Games: Artificial Intelligence, Design and Education (CGAIDE 2004)*, 29–35.

[Garcia 2004] Garcia, I. ; Molla, R., and Barella, A. 2004. Gdesk: Game discrete events simulation kernel. In *Journal of WSCG, 2004*.

[Lionhead 2001] Lionhead, S. 2001. Black&white.

[Maes 1995] Maes, P. 1995. Artificial life meets entertainment: Lifelike autonomous agents. *Communications of the ACM* 38:108–114.

[Mueller 2006] Mueller, E. T. 2006. *Commonsense Reasoning*. San Francisco: Morgan Kaufmann.

[Nilsson 1996] Nilsson, N. J. 1996. *Introduction to Machine Learning*. http://ai.stanford.edu/people/nilsson/mlbook.html.

[Nintendo 2005] Nintendo. 2005. Nintendogs.

[Rabin 2002] Rabin, S. 2002. *AI Game Programming Wisdom*. Charles River Media.

[Ripley 1996] Ripley, B. D. 1996. *Pattern Recognition and Neural Networks*. Cambridge University Press.

[Russell 1995] Russell, S., and Norvig, P. 1995. *Artificial Intelligence: A Modern Approach*. Prentice Hall International Editions.

[Stanley 2005] Stanley, S. O. ; Bryant, B. D., and Miikkulainen, R. 2005. Evolving neural network agents in the nero video game. In *Proceedings of the IEEE 2005 Symposium on Computational Intelligence and Games (CIG'05)*, 108–114.

[Zell 1995] Zell, A. ; Mamier, G., and Vogt, M. 1995. Snns - stuttgart neural network simulator. user manual, version 4.2.

MODELING AGENTS FOR REAL ENVIRONMENT

Gustavo Henrique Soares de Oliveira Lyrio
Roberto de Beauclair Seixas
Institute of Pure and Applied Mathematics – IMPA
Estrada Dona Castorina 110, Rio de Janeiro, RJ, Brazil 22460-320
email: {glyrio,rbs}@impa.br

Computer Graphics Technology Group – TECGRAF
Catholic University of Rio de Janeiro – PUC-Rio
Rua Marqus de So Vicente 255, Rio de Janeiro, RJ, Brazil 22453-900
e-mail: {glyrio,rbs}@tecgraf.puc-rio.br

KEYWORDS
Agents, Game Development, Simulation, Agents Framework and Lua.

ABSTRACT

This work presents Modeling Agents for Real Environments, a framework to build predefined behavior agents and insert then into real environments to run in real time. Coded in Lua, MARE departs from a formal definition of agents and join games techniques such as A* algorithm, finite state machines, terrain tiles and others to build a fully customizable tool for games or simulation creation.

INTRODUCTION

When developing new games programers, in some fase of the project, will face the necessity to build artificial inteligence models. Most times, those models could be reused (at least as the starting base for new ones) from previous game projects. How nice would be if those models could be just imported? And the modifications made in a script level? Modeling Agents for Real Enviroment - `MARE` intent to provide that. Born to simulate any real environment in real time `MARE` offers generic map and agent classes with methods to realize a sort of actions like walk, talk and interact. The main idea behind `MARE` is join popular techniques used in games, generating a tool that provides a lot of methods to create new terrain types, agents, actions and interactions allowing game creation. `MARE` was coded in `Lua`, a new powerful programming language that has become popular in the game industry as a script language. `Lua` is very easy to comprehend and doesn't demand that the programmer spend a lot of time writing extensible code. Also it's extensible and a multi-platform language and `MARE` has inherited these properties. `Lua` was coded in `ANSI C` and is available to any platform that offers such compiler (Windows, Linux, Mac OS, Solaris, etc.). So, `MARE` is available to all that platforms too.
In the next sessions we will describe how `MARE` was modeled and implemented. It is composed by three distinct parts: Environment, Agents and Rules of Interaction

ENVIRONMENTS

All games have at least one environment. It could be a forest, a mountain, a river, a medieval city, etc. In most cases, games have more then one environment. `MARE` environment differs from game environment because it represents the portion of the game environment that is relevant to the agents.
For example imagine a snow covered plain. Is it really relevant to the agents the agents that walk over that plain to know that it's covered with snow? If the answer is yes, then the snow will be part of `MARE` environment, else it will not.

`MARE` environment is divided in two parts: terrain and objects. The terrain is modeled as a two dimensional array (or a matrix) of tiles. A tile is nothing more than a building block of terrain. Put together several tiles and you can make a terrain. Each tile has a lot of attributes and methods to customize the terrain. Objects are anything that isn't a tile or an agent and can be placed over the terrain (constructions, vegetation, rocks, chairs, swords, etc). Agents also can interact with objects.

Terrain

The reason number one to use tiles to model terrain is that tiles saves memory. Consider a terrain with 100 * 100 tiles in it. That results in a total of 10,000 tiles. Let's use one large bitmap for the terrain instead of tiles. To calculate how much memory the terrain requires, you must multiply the total number of tiles by the size of each tile. Let's consider we are using 64x64 pixels tiles. The following demonstrates this concept:

```
100 tiles wide * 100 tiles high = 10,000 tiles
64 pixels wide * 64 pixels high = 4,096 pixels
per tile
10,000 tiles * 4,096 pixels * 1 byte (8-bit) =
40,960,000 bytes (w/ 256 colors)
10,000 tiles * 4,096 pixels * 4 bytes
(32-bit)=163,840,000 bytes (true color)
```

The simple 100x100 terrain will use 163 megabytes of storage if you use true color. Even if you use 256 colors it will use 41 almost megabytes of memory. Let's calculate the memory storage requirements for the same 100 x 100 terrain, using 100 different tiles.

100 tiles wide * 100 tiles high = 10,000 tiles
64 pixels wide * 64 pixels high = 4,096 pixels per tile
100 different tiles * 4,096 pixels per tile * 4 bytes per pixel = 1,638,400 bytes (to store all tiles)
10,000 tiles * 1 byte per tile = 10,000 bytes (one byte to index each tile)
10,000 bytes + 1,638,400 bytes = 1,648,400 bytes total (indexes + tiles)

The terrain now uses less then 2 megabytes total. You can use a 1000 tile set and still use less than 20 megabytes of storage.

The second reason to use tiles is that they reduce the workload of terrain creation because the same tile pattern is used multiple times within the same image. Also the tile can be rotated to make new tiles that do not need to be reloaded in memory.

MARE tiles have the following attributes available: index, texture, terrain type elevation and offset:

index identify the tile in the terrain;

texture defines the tile image;

terrain type it's an index for the know types of terrain. That allows MARE to identify the tile's characteristics and associate a tile type with agent's interactions;

elevation describes the height of the tile;

offset is the number of pixels that the image corner differs from the tile corner.

Figure 1 shows an example of terrain with four different properties. The terrain scale again can be defined by the user. MARE for default uses 48x48 pixel tiles representing cells of 1 square meter.

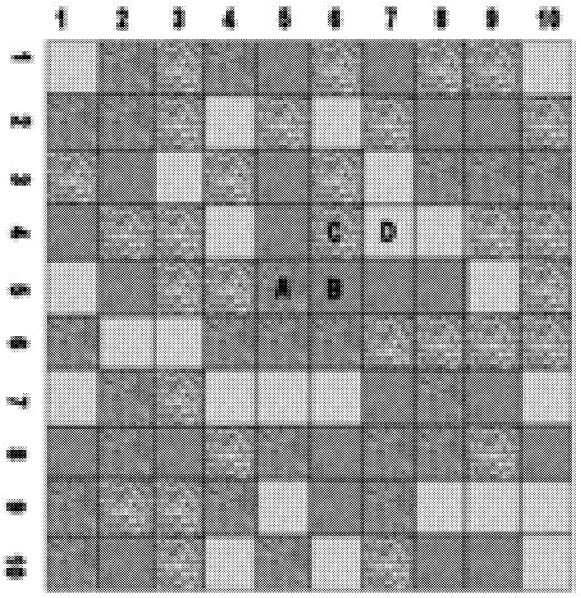


Figure 1: Sample array 10x10 with 4 different tiles modeling a terrain with grass(A), water(B), rock(C) and sand(D) properties. The terrain scale again can be defined by the user. MARE for default uses 48x48 pixel tiles representing cells of 1 square meter.

Objects

Objects are everything that isn't an agent or a tile. It can be rocks, bullets, trees, chairs, clouds, weapons, etc. Objects are placed over the tiles and can interact with the agents. Let's consider a mine for example. It will be placed over a position on the terrain (tile) and if any agent steps on that tile, the mine will run it's interaction function making it explode.

MARE objects have the following attributes:

Time ratio

MARE runs in real time, doing a loop over all the agents that are current inserted on the environment and allowing then to run short time actions. MARE sets the fastest agent movement speed to 1 pixel per time. All the other agents have fractions of that agent's speed. That avoid appear-disappear effect.

For example, let's consider that our tile size is 1 generic unit. Consider also an environment that models tanks and spaceships. The spaceships are the fastest agents, so their speed will be 0.8 tile size. Tanks are lot slower then spaceships. So, the tank speed will be 0.25 tile size.

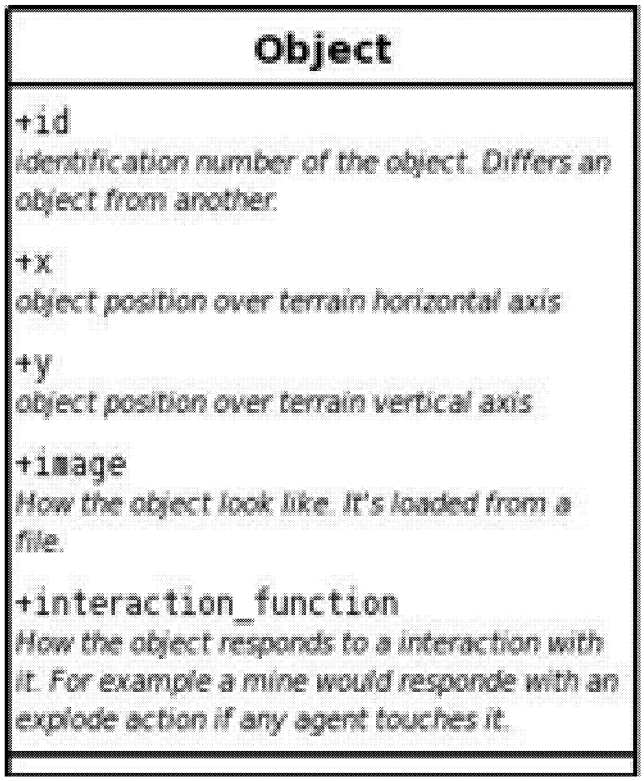


Figure 2: MARE object attributes.

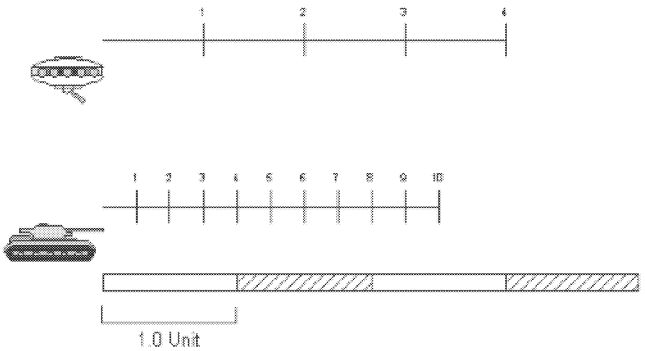


Figure 3: Graphic representation of how agent's speed works in real time. Extracted from (BARRON).

Agents

"An agent is anything that can be viewed as perceiving its environment through sensors and acting upon that environment through effectors. (RUSSEL)"

MARE starts from that formal definition to create its agents. Each agent has sensors and effectors. The sensors were modeled with answers that the agent will have to know how to answer with true or false and each effector will be an action that an agent can take over the environment, over other agent or even over himself.

After modeling the agent's sensors the user will group it in states. That states will describe a condition that an agent can be.

Now that we have seen how MARE models sensors, we need to understand how it models effectors. MARE does that, defining actions that the agent will take when in a state.

We also, identify which events may occur on the environment changing our agent's state. We call these events transitions. We saw till now that MARE agents have states, action and transitions. That will lead us to another common technique used in game programming: Finite State Machines.

Finite State Machines

"A finite state machine – FSM consists in a set of states (including an initial state), a set of inputs, a set of outputs, and a state transition function. The state transition function takes the input and the current state and returns a single new state and a set of outputs. Since there is only one possible new state, FSMs are used to encode deterministic behavior. (FUNGE)"

Our FSM initial state will be the initial agent state. The inputs will be the sensors and the outputs will be these sensors updated. Finally, transitions are conditions that have to be fulfilled to allow state transition function to modify the current state.

Agent Class

The agent class has the following attributes:

Predefined Agent actions

MARE has some predefined actions that should be enough to model any kind of agent that users may need for their simulations. If MARE doesn't provide the action that user judges necessary then it's possible to easily write the new action or even replace a existing action for a new

Figure 4: Description of an agent class.

one. The following list shows and explains the actions already available in MARE:

Move the move action is based on a array of way-points returned by an A* algorithm. The agent turns its direction to the first way-point in the array and move each frame by its speed until the next way-point is reached. When the last way-point is reached the action is ended;

Stop identify the next way-point in the agent way-points array. Removes all the way-points that come after it, making the action of move to stop as son as the agent reach the way-point identified;

Flee when the flee position (object or another agent) enters the line of sight the agent will start a move action to the opposite position;

Pursuit receives a position of a target (object or agent) and moves to that position using the move action;

Affect do some action to another agent, object or itself;

Say sends a message to the target. Maybe another agent or even to the screen;

Think Holds the agent's attention to an event or another agent.

Rules of Interaction

This are a set of rules that will define how MARE components (agents, environment and objects) threat events of interacting with other MARE components. These rules allow that the agents response to then can be set directly by the user.

Agent x Agent interaction

This rules will describe how and agent will respond to an event generated by other agent. With this simple set of rules MARE intend to cover all possible interaction between two distinct agents. Again, if the user finds that any other rule is still necessary, MARE will provide methods for him to code and insert as many rules he thinks he will need.

onOverTile this agent steps over a tile

onObjectAppear an object enters the line of sight.

onObjectDesappear an object leaves the line of sight

onObjectAffect an object does something that directly affects the agent. This can look strange at first time, but let's consider a mine for example: it's a MARE object, as we saw above, and it could explode affecting one or more agents next to it.

Results

This session will present an examples on how to use MARE to build agents for games, presenting creatures that live

on a hole.

Creatures on a Hole Model

Let's consider hungry creatures that live on a hole. They remain randomly moving until another specie of creature falls on the hole. Then the hole creatures will try to eat that creature. Let's see how that kind of agent can be modeled by MARE.

For this example we will build a hole_creature class that will have an agent class as one of its attributes:

Hole Creature attributes:

- MARE agent

Hole Creature methods:

- random move()
- pursuit target()
- attack()
- look for target()
- can_reach_target()

After that, we need to consider which sensors are needed for our agents. Fortunately two sensors will do the job. The questions that our agents will have to be able to answer is: Can i see some other creature that's not a hole creature? Can i reach the creature? Naturally if the answer is yes (`true`) for the first one, the agent will pursuit the creature. Else if the answer is no (`false`) the agent will remain in random move through the hole. If the answer is yes to the second one the hole creature will attack, else it will remain pursuit while it remains in the line of sight.

Now that we have sensor and effectors we can define three different agent states: random move state, pursuit state and attack state. As we saw above states demand state actions and transitions.

While in our first state (random move) the agent will have its sensors with both negative answers. It cannot see a different kind of creature, and, of course, it cannot reach it. The agent will execute random move() function, that consists in a call to MARE predefined action move() with a random environment position as destiny. The state condition function will be look for target() that calls a MARE interaction rule onAgentAppear verifying if the agent type isn't another hole_creature. Finally we will set the state transitions. If conditional function returns yes then a transition to pursuit state will be made. If the answer is no then the agent will remain in random move state.

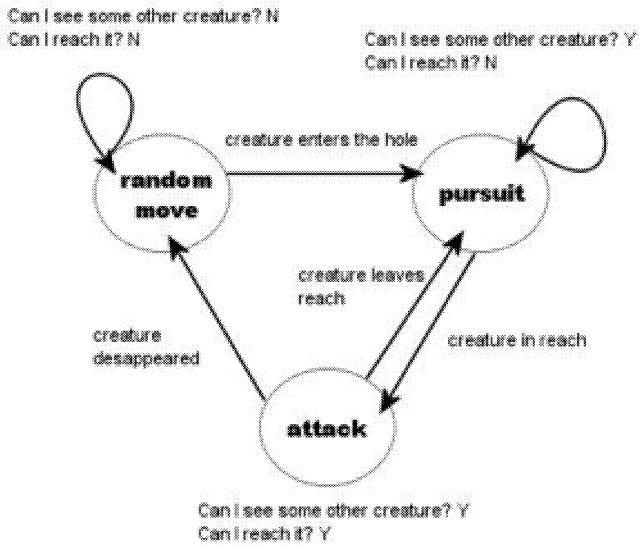


Figure 5: Graphic representation of hole creature agents FSM.

For the second state (pursuit state) the agent will call can reach target() method , which will call another of MARE's predefined rules of interaction: onOverTile; to verify if its positioned next to the target. If yes it will change to attack state, if not it will remain in pursuit.

For the third and last state (attack state) the agent will call method attack() that consists in another call to one more predefined MARE action: affect() with the target creature as parameter. The agent will remain in that state until the creature dye or until it leaves the agent's reach.

Conclusions

The fact that MARE was builded above a formal definition of agents guarantees that it's able to construct any kind agent the user may need. Actually MARE provides tools to generate only pre-defined behavior agents. Although artificial intelligence has been growing a lot on games, these kind of agent still are the most common type of agents, because they are simple to code, understand, maintain, don't demand high computation power and can generate very complex behaviors.

MARE is a agent framework that is not restricted to the game development. It can provide tools to researchers build simulation environments. Currently MARE has been used to simulate enemy behavior (conventional forces and guerrilla) for military training on Brazilian Marines Simulation System.

Lua language is commonly used as a scripting language due to its simplicity and short time learning curve. With

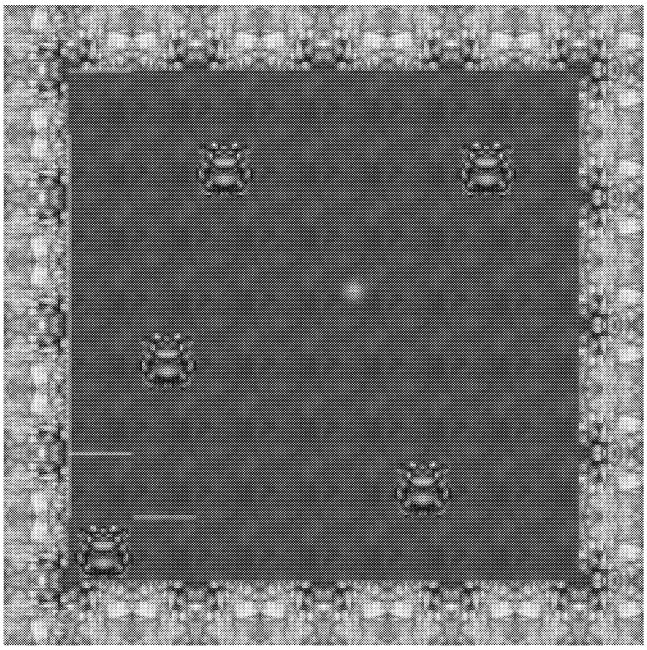


Figure 6: Possible result of implementation of creatures on a hole.

it's core developed in `Lua` language `MARE` not only provides a platform independent tool to create agents behavior but also a simple language to users create their agents. `MARE` users will get quickly familiar with Lua, and that will make then able to improve and even upgrade `MARE`'s core with no effort.

Future Works

Insert a fuzzy logic module will allow the agents decide when they want to take an action improving the framework a lot and insert sockets (by `Lua` sockets) to support network games.

REFERENCES

[RUSSEL] S. Russel P. Norvig, *Artificial Intelligence, A Modern Approach. Second Edition*, Prentice Hall, 1995.

[FUNGE] J. David, *AI for Games and Animation: A Cognitive Modeling Approach*, AK Peters, 1999.

[BARRON] T. Barron, *Strategy Game Programming With DirectX 9.0*, Wordware Publishing Inc., 2003.

[MULHOLLAND] A. Mulholland T. Hakala, *Developers guide to multi-player games*, Wordware Publishing Inc., 2001.

[REYNOLDS] C. Reynolds, *Steering Behaviors for Autonomous Characters*, Proceedings of Game Developers Conference, 1999, 763-782.

[JIANG] B. Jiang, *Agent-based Approach to Modeling Environmental and Urban Systems within GIS*, In Proceedings of 9^{th} International Symposium on Spatial Data Handling, Beijing,2000.

AUTHOR'S BIOGRAPHY

Roberto de Beauclair Seixas works with Research and Development at Institute of Pure and Applied Mathematics - IMPA, as member of the Vision and Computer Graphics Laboratory - Visgraf. He got his Ph.D. degree in Computer Science at Pontifical Catholic University of Rio de Janeiro - PUC-Rio, where he works with the Computer Graphics Technology Group - TeCGraf. His research interests include Scientific Visualization, Computer Graphics, High Performance Computing, GIS, Simulation Systems and Warfare Training Games. Currently he is the advisor of the Warfare Games Center of the Brazilian Navy Marines Corps.

Gustavo Henrique Soares de Oliveria Lyrio works with the Computer Graphics Technology Group - TeCGraf. He got his B.Sc. in Computer Engineering at Pontifical Catholic University of Rio de Janeiro - PUC-Rio. His interests include Computer Graphics and Warfare Training Games. Currently he is developer of the Warfare Games Center of the Brazilian Navy Marines Corps.

GAME ACTORS

USER INTERFACES FOR THE PROVISION OF STRUCTURED INFORMATION AND GUIDANCE FOR ACTORS IN VIRTUAL WORLDS

Alpesh P. Makwana
University of Central Florida
Institute for Simulation and Training
Orlando, Florida, 32826, U.S.A.
E-mail contact: amakwana28@gmail.com

KEYWORDS

Guidance, performance management, user interface, collaboration, role playing games.

ABSTRACT

Researchers at the University of Central Florida are developing a *Cast Member Performance Management (CMPM)* methodology to structure the learning experience of students using a Multiplayer On-line Role Playing Games (MORPG). This paper describes the architecture of the *Cast Performance Management System (CPMS)* which is an implementation of the CMPM methodology. The paper briefly describes the plan for structuring guidance for students and the flow of control while using the CPMS.

INTRODUCTION

Interactive virtual environment has been used for teaching, education, training and gathering knowledge for more then a decade. A large number of researchers and scholars recognize the impact of interactive immersive games and are exploring the potential of video games and computer games to engage learners (Gee, 2003; Prensky, 2001; Squire, 2002). In recent years, using a virtual character in Multiplayer Online Role Playing Games (MORPG) millions of globally distributed gamers interact, collaborate, and form relationships in the network (Yee, 2006; Seay, Jerome, Lee, & Kraut, 2004; Bob, 2004). Research is needed on how to provide training, guided feedback and delivery of structured information in these cooperative activities. Moshell & McDaniel (2006) developed a new paradigm called *Cast Member Performance Management* based on the concept of developing a "*community of learners and organizing their activities."*

A *Cast Member (CM)* denotes a participant in a MORPG who has received special training, and who may receive direct guidance during an exercise, for the purpose of helping to structure the experience of other students called *guest(s).* A guest is a participant in an MORPG who has little or no prior experience with the scenario being presented. A pilot study of CMPM demonstrated the use of this technology in various applications such as learning mathematics, cultural and language learning, and entertainment (Moshell, McDaniel, Makwana, & Wei, 2007). However, little is known about how to provide structured information and guidance to the virtual actor in this shared environment. This paper describes the architecture, flow of control and possible implementation of a Cast Performance Management System (CPMS).

DESIGN OF CAST PERFORMANCE MANAGEMENT SYSTEM

The CPMS user-interface consists of four window panels titled Current Goals & Users, Resources, Story Point Dialogue and Story Map. All the window panels are configurable; they can be minimized, maximized, closed and rearranged or resized as needed.

Current Goals and Users panel

This window panel gives a brief overview of a current story segment. The CM learns the name, objectives, and number of users involved in a current story segment by reading the information listed in the Task-Name, Aim and Users fields. This information also helps the CM to identify the role of other members in a story segment. When a CM moves to a different segment of the story, the information in the fields gets changed and new information is displayed for the next story segment.

Story Point Dialogue panel

The Story Point Dialogue displays the narrative text of the current story segment. Each story segment has a designated objective, and the CMs are supposed to help the guest achieve that objective. The CM can read the narrative text and thereby perform the task related activities as guided in a current story segment. For example, as shown in figure 1, the CM may ask the guest if he/she has the library card. If the guest already has a library card, then the CM will click on "Click Here" and the story point moves forward to the next story segment.
However, if the CM realizes that the guest does not have a library card, then the CM will simply follow the sequence of instructions available in a current Story Point Dialogue. Performance reference and memory assistance are the two principal benefits of displaying the narrative text.

Resources and Collaboration panel

This window panel is a "Chat System" so that CMs can communicate with each other by typing on the keyboard and sending messages to other CMs, or the Show Director.

When a CM clicks "Send" button or hits "Enter", the typed information will be delivered directly to the screen of the selected member. To increase the chances of successful communication, the story segment can also make use of graphical information. The graphical information can be an image, image embedded with text, audio, or video related to particular segment of a story.

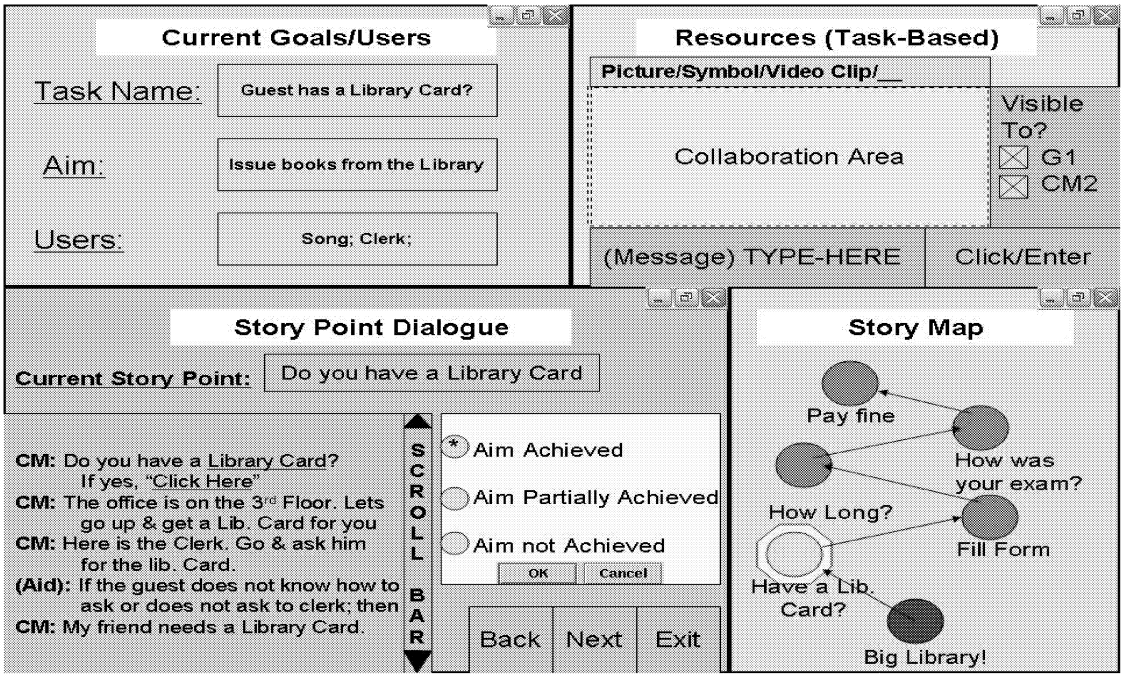


Figure 1: Design and Flow of control of CPMS

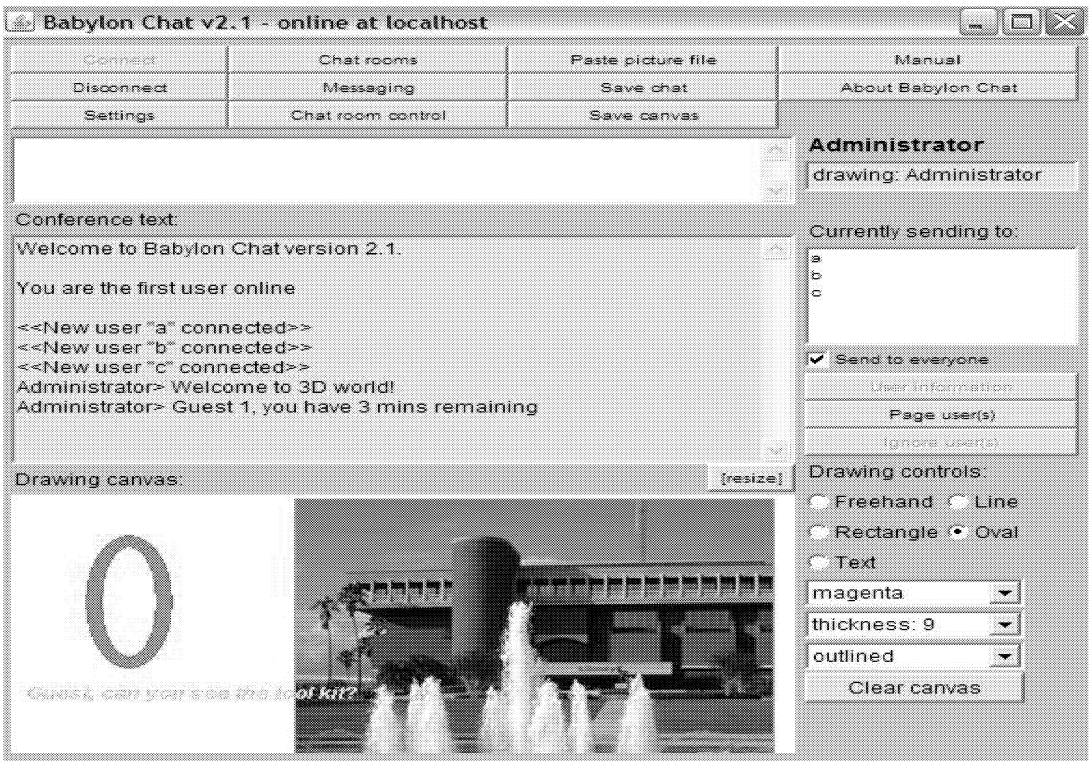


Figure 2: Whiteboard system for Resources and Collaboration panel

Figure 2 shows the implementation of the resources and collaboration panel. It is based on the Java-based whiteboard utility called Babylon Chat. This chat system can create any number of users, create and manage chat rooms, and supports an interaction in private or public. The whiteboard drawing utility can support freehand, image files, squares, ovals, and lines with varying font size and colors. This system is ideal for the CMs to share concepts with Guests.

Story Map panel

The Story Map consists of scattered circular nodes and lines marked with arrows connecting them, and depicts the sequence of story segments. These nodes represent the *"Task Name"* of a story segment. Upon looking at the Story Map, the CM recognizes how the current segment relates to other segments in the story. Initially, before any scenario begins, all the nodes are colored red indicating that objectives for the nodes are not yet achieved. As the CM and guest interact with each other in a segment, the CM will determine the success of the guest in a particular segment by selecting one of the possible options, *Aim Achieved, Aim Partially Achieved, or Aim not Achieved*. These three choices of determining a success appear on a pop-up window when the CM clicks the "*Next*" control button. The CM can go to the preceding segment by clicking the "Back" control button. To exit from the CPMS, the CM will click on the "Exit" control button. The CM confirms the selection by clicking the "Ok" button.

The color of a node gets changed depending on the type of selection being made. The Aim Achieved selection turns the red node into blue, Aim Partially achieved turns red node into Orange, whereas Aim not Achieved does not change the original red color. Changing the color of the node serves as a marker for the CM so as to identify how many tasks has been solved successfully, unsuccessfully or not yet fully successfully.

IMPLEMENTATION

We are currently evaluating the Java Eclipse integrated development environment. Eclipse includes a number of application frameworks that support text editing, graphics and distributed collaboration. We hope to develop a prototype of the CPMS so as to begin testing it during academic 2007/08. CPMS will run in a separate window alongside the role playing game. In some circumstances, CPMS may run on a second computer screen alongside the main game system. This independence from the game engine makes it possible to test CPMS with a variety of online game environments.

CONCLUSION

The architecture of CPMS shows how we plan to deliver structured guided information and allows for a natural representation of a possibly complex "flow of control", depending on actions by cast members and guests. The proposed system will be distributed through an Internet application which has substantial window management capabilities.

BIOGRAPHY

Alpesh P. Makwana is a Modeling and Simulation PhD student at the University of Central Florida. He holds an M.S. in Modeling & Simulation, professional certification in Instructional Design for Simulations, and a B.E. in Computer Science. He is working as a Graduate Research Assistant at the UCF Institute for Simulation and Training. His research interests include collaborative learning in virtual environments, collaborative learning tools design and development, and language learning using Multiplayer Online Role-Playing Games (MORPGs).

REFERENCES

Bob, G. (2004). The Corporation News. MMOG Roundup: Depressing 2004 Edition http://www.corpnews.com/news/fullnews.cgi?newsid1081411764,6286,

Gee, J. P. (2003). What Video Games Have to Teach Us About Learning and Literacy. New York: Palgrave Macmillan.

McLaughlin, A (N.D.) Babylon Java Chat. Retrieved July 30, 2007 from http://visopsys.org/andy/babylon/

Moshell, J. M., & McDaniel, R. (2006). Cast Member Performance Management: Research Report for Fall 2006. Technical Report UCF-SFDM-2006.1. University of Central Florida, School of Film and Digital Media (Orlando, Florida). Available at the following URL: http://www.cs.ucf.edu/~jmmoshell/techreports/2006.1.pdf

Moshell, J.M., McDaniel, R., Makwana, A.P., & Wei,L. (2007). Managing Actors in Serious Games. Proceedings of *GameOn (North America) Conference* for 2007 September 10-12, 2007 – Gainesville, Florida USA.

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

Seay, A.F., Jerome, W. J., Lee, K. S., & Kraut, R. E. (2004). Project massive: a study of online gaming communities. *Conference on Human Factors in Computing Systems CHI '04 extended abstracts on Human factors in computing systems,* 1421-1424.

Squire, K. (2002). *Cultural Framing of Computer/Video Games,2,* Retrieved April 1, 2007 from http://www.gamestudies.org/0102/squire/

Yee, N. (March 2002). Facets: 5 Motivation Factors for Why People Play MMORPG's. Retrieved April 2, 2007 from http://www.nickyee.com/facets/facets.PDF

MANAGING ACTORS IN SERIOUS GAMES

J. Michael Moshell+, Rudy McDaniel+, Alpesh P. Makwana++ and Li Wei*
University of Central Florida
+ Digital Media Department +
++ Institute for Simulation and Training ++
* Modern Languages Department *
Orlando, Florida 32816 USA

E-mail contact: jm.moshell @ cs.ucf.edu

ABSTRACT

Researchers at the University of Central Florida have developed a process for structuring social interaction in serious (i. e. education-oriented) games through the creation and management of a team of "on-line actors" called *cast members.* This paper describes the evolving theory of *Cast Member Performance Management (CMPM).* Pilot studies have concerned applications in math, cultural and language learning and entertainment.

CAST MEMBER PERFORMANCE MANAGEMENT

An imaginary vignette.

Jason logs into the *Turtle Haircut Network* to complete a homework assignment in his eighth grade social studies class. He puts on his headphone/ microphone; this is an audio-interactive game. He finds himself in a role-playing game where he is a British junior clerk in a trading company in Hong Kong in the year 1840. The First Opium War is about to break out. He interacts with other British and Chinese characters – soldiers, taipans, street vendors – that he has read about in his textbook. He has to make decisions, and ultimately has to run for his life when his trading company is burned down. He gets an up-close and personal experience of an important piece of history.

Sounds like a typical adventure game, right? But the game itself was built by college students and high school seniors, and the "British" and "Chinese" characters are all being controlled by other students, in other schools, who are themselves studying various aspects of world history. Jason discovers links that lead to other games in which he can learn to speak Chinese by interacting with people in China, or create comic book-style stories about his experience in Virtual China.

The key intended benefits of Cast Member Performance Management are these:

1) Complex and interesting scenarious can be developed without extensive programming to support the activities of non-player characters or interactive props – because live human beings will be controlling these entities;

2) The cast members themselves will learn at least as much as the guests. They will have to master the subject matter of the world sufficiently to deliver it in an effective and interesting way. And they will learn to think about others' learning and entertainment experience, thus enhancing their own ability to plan and present information and action in electronic media.

Our initial experiments indicate that these goals are achievable.

Making it happen.

Young people are learning many things by playing games (Gee, 2003; Shaffer et al., 2004; Prensky, 2001, 2006; Aldrich, 2005). They develop skills at solving complex problems, exploring intricate universes, and organizing and leading group activities. Our mission is to seek ways of harnessing this self-organized, un-commanded, often addictive behavior to teach essential skills and concepts. To this end, we are developing a methodology for structuring the experience of students engaged in multiplayer online role playing games for learning. We have borrowed terminology from Disney's theme parks, as follows:

> **Guests** are students who enter our role-playing games without prior background, and who are the "customers" in the sense that the game is designed primarily for their benefit. *The character played by the guest is called the Protagonist.*
>
> **Cast Members** are other people who are playing roles within the game, with the primary purpose of providing a high quality, interesting/exciting/creative experience for the guests. The cast members might be other students in the same or different school, university students, or even retirees. *Characters played by the cast members are referred to as Non-Guest Characters (NGC)* – by analogy to the automatically controlled Non-Player Characters (NGC) that often occur in games.

To make the CMPM vision real, we must solve a number of problems including world-building, cast and guest training, performance management and infrastructure. This paper describes our work toward these goals with students at the University of Central Florida during fall 2006 and spring 2007. In both semesters, approximately a dozen Digital Media students participated. In Spring 2007 we also had participation from eleven students in their first year of studying the Chinese language.

World Building is not the principal focus of our current phase, but we had to build worlds for our experiments. Tools for building online worlds are rapidly evolving. We have used three of them. *Neverwinter Nights* (Bioware 2002) has been used by a number of experimenters, and offers simple and reliable multi-player support, so we used it during the Fall of

Figure 1. The Cursed Dark City

2006. Constructing worlds is simple using the built-in tools, if you plan to build medieval-style towns, villages and rural scenes and populate them with knights and monsters.

However it proved quite difficult to add features such as 2d Cartesian coordinate system that we needed for our math lessons. The Torque Game Engine (Garage Games 2007) also came highly recommended, and so we adopted it during Spring 2007. However, to build assets you need expertise in either 3Dmax or Maya, and students with these skills were in short supply. Since Torque is a combat-oriented engine, there is always a weapon in view (e. g. a crossbow.)

By mid-semester, it was clear that world-building was consuming too much energy, so we began to use *Second Life* (Linden 2007). Its built-in tools for construction of scenes, objects and scripts proved sufficient for the short term. We extracted imagery from our Torque world to build our Second Life environment.

In Figure 2, you will see some urban landscape built in 3d in Torque, that was used as a source of texture maps in the Second Life world (below). See (Moshell 2007a) for an extended discussion.

We used the Ventrilo (Flagship 2007) audio communications system, which worked well for our purposes. A number of game engines including Second Life and Olive (Forterra 2007) now incorporate audio chat into their core technologies.

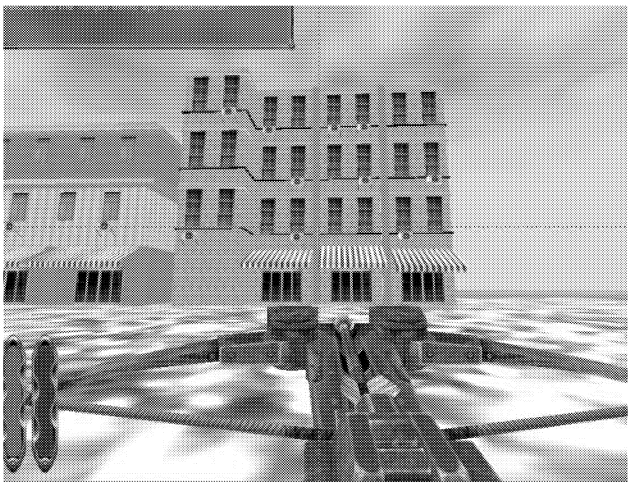


Figure 2. Torque as a Testbed

Figure 3. The Beijing Library in Second Life

Story Lines and Story Guides.

We invited our students to develop stories around themes that we specified. In 2006 the theme was built aroujnd a middle school math requirement to understand and use two-dimensional Cartesian coordinates. To develop a story we had student teams create and "pitch" a variety of protagonists, antagonists and story lines, then select the most promising. *The Cursed Dark City* was a quest game in which the protagonist moves through a village, going to various X,Y coordinates to gather necessary resources from shopkeepers. The protagonist then goes into the cursed dark city to restore it to life and light. (The story is discussed in more detail below.) The plan was for shopkeepers, antagonists and the protagonist's companion/guide to be played by cast members.

The key problem: how do you guide the cast members? If they have too little structure, the entire burden of delivering useful learning falls on the environment. If you give too much structure, there is no scope for creativity by cast members –

and you must anticipate every possible choice and action by the guest. Our response was to design documents called *Character Profiles* and *Segment Guides*.

Character Profiles provide to the cast member (and to the guest) a summary of each character's background, knowledge, capabilities and goals. It is often the case that a cast member will be controlling more than one NGC as the story evolves (though not usually at the same time.)

Segment Guides (SG) are, obviously, based on segments – that is, the discrete parts of a story, or situations that may arise. They are known to (and rehearsed by) the cast members but not to the guest. For instance in the Cursed Dark City, there is one Segment Guide for each shop into which the protagonist may go. A SG specifies the conditions that must be true for this segment to occur. These include the presence of the appropriate characters in the appropriate environment, and prior events that must have happened to trigger this segment. More than one segment can occur in a given location. Sometimes not all of the preconditions for a segment are present; the cast members must then rely on their character profiles to decide what to do.

Segment guides do not dictate actions. Rather they express local goals for the NGC, as well as for the cast member. Goals might look like:

"Merchant: try to sell the protagonist at least one communications device."

"Cast member: try to guide the protagonist to discover the (x,y) coordinate grid that is on the ground."

Segment guides can suggest dialog. The degree of detail varies greatly depending on the level of subject matter. In introductory language work, the segment guide provides a complete skeletal dialog, that should be memorized and used as a starting point for improvised discussion.

Goals for characters are limited to issues that make sense within the story. Characters within the imaginery universe have no knowledge of the educational objectives of the game. Cast members (real humans controlling imaginary characters) on the other hand, have goals at a higher level: they are charged with leading the protagonist's action so that the guest ultimately has the intended experience. There are an infinite number of ways to accomplish any given learning objective, depending on the cast's creativity.

Training

An obvious way to train cast members would be for them to serve first as guests. However, with new worlds, someone must initiate this "feedback process". If the world is still under construction, training presents additional challenges. We have had good success in developing game concepts and in training our initial cast members by building table-top "board games". We use concept art (e. g. photographs from the Internet) and first-draft Character Profiles and Segment Guides. We have the cast members play through the story several times, and then invite an outsider to serve as the Guest. On some occasions (e. g. foreign language training) we use native speakers or faculty members as initial cast members, so as to help organize the cast members' skills.

Show Management

Cast members do not only control avatars. They also control the 'props' – that is, any objects or interactive aspects of the world itself. For instance, if lights are to come on, gates to open, threats to appear, rather than having to program these interactions, we assign a student Prop Manager the responsibility for managing them, improvising as the story evolves. An Show Director provides guidance to the cast members so as to make sure that overall goals are set.

HOW DOES IT WORK IN PRACTICE?

We have conducted four trials, as summarized in Table 1. "DM"=undergraduate Digital Media majors. "Language"=various majors taking Introduction to Chinese. "Chinese" = graduate students from China, serving as cultural consultants.

Table 1: Trials of Cast Member Performance Management

Topic	Scenario	Engine	Participants
Cartesian coordinates	*The Cursed Dark City*	Neverwinter Nights	14 DM
Cultural Immersion	*New Student in China*	Torque	12 DM + 2 Chinese
Chinese Language 1	*Beijing Library*	Second Life	12 DM + 11 language
Entertainment	*Murder Mystery*	Second Life	12 DM

The Cursed Dark City served to develop the concept of the Segment Guide and the role of Show Manager. In the *New Student in China* scenario, we consulted Chinese graduate students and American students who had lived in China, to simulate the experience of living in a Chinese city and shopping in a computer store. In the Beijing Library story we developed an intensive role-playing language practice session, based on one chapter of the introductory language text in use at UCF (Liu, 2002). The *Murder Mystery* explored more elaborate Segment Guides and plot situations.

Cast Members In Action

A typical session involves four to six laptop computers. The cast members (including actors and controllers) sit around a table, with their computers networked together. The guest is in a separate room (if we are using Ventrilo audio) or in the same room if we are simply speaking out loud. The guest's game viewpoint is projected on the wall so that all cast members, managers and observers can see the guest's experience.

Figure 4: Cast Members in Action

A typical scenario lasts for five to ten minutes. Guests are initially told the starting premise: e. g. "you are a new student in China. You do not yet speak Chinese. Go into the city, find your way to a computer store, and buy a new monitor for your computer." After each session, the guest is invited to observe following guests; most accept this invitation. A typical evening will involve three or four sessions, with careful debriefing of cast, guest and observers.

Key observations

1) There is a wide variety of natural talent for cast member performance. Some students instantly "get it", while others struggle with the challenge of remaining in character. It will be necessary to pre-qualify cast members when we move to the delivery of actual instruction.

2) Cast members must actually know more about the subject than guests, for effective learning to occur. Lack of second year language students forced us to use first year students for both guest and cast. When the stronger students served as cast members, useful learning took place- especially after they had several sessions of experience. When weaker students tried to be cast members, they often tried to read from the Segment Guides rather than working from memory and improvising.

3) The role playing experience is highly motivating. Students exerted great effort to build intriguing worlds and story lines, and to enact them in creative ways. We have not formally measured the effectiveness of subject matter learning since we have been focusing on development of the management methodology. However, this model (CMPM) may have a great potential in providing a more interactive and proactive environment in foreign language studies, especially in light of a recent MLA (Modern Language Association) report which signals a paradigm shift from the traditional grammar/literature based model to more cultural oriented, task-based models (Geisler et al., 2007).

4) We still seek the right game engine. Second Life's advantages are its accessibility from anywhere, and the ease of world-building and interactive use. But only one team can use a given 'set' at a given time. A scaleable model must allow hundreds of concurrent sessions in a given world. In Second Life, you would have to purchase and maintain large amounts of real estate to accomplish this goal. We continue to seek an engine that is easy to use, supports interactive audio and does not incur additional cost for concurrent sessions.

Conclusion

Cast Member Performance Management (CMPM) is a new approach to organizing peer interaction in on-line role playing games for learning. We have reported on a pilot study in which we have developed and tested some ideas and techniques, working with middle school mathematics objectives, cross-cultural learning, foreign language learning and entertainment.

We have focused on learning how to structure and guide the interactive dramatic activity of untrained (or lightly trained) student "confederates" called *cast members*, so as to deliver specific learning experiences to other students, called *guests*. At this early stage of development, we have not undertaken to measure the impact of these techniques on learning. We have focused our investigation on the dramatic experience itself.

We are now designing a set of Internet-based collaborative work tools to facilitate the coordination of cast member/guest interaction. We plan to develop and test these tools during 2007 and 2008, and to conduct formal studies of learning effectiveness.

More details about the work can be found in (Moshell 2007a, b).

Acknowledgements

The authors gratefully acknowledge the assistance and support of Dong Mi and Jian Yang who participated in the Cast Member Performance Management Project during the spring of 2007 and provided valuable information about Chinese culture.

The students in UCF Digital Media classes in the fall of 2006 and spring of 2007 were excellent and creative collaborators. The students from First Year Chinese were patient and energetic in applying their new language skills.

This work is based on earlier work by Michael Moshell and Charles Hughes (Hughes 2000, Moshell 2000).

BIOGRAPHIES

J. MICHAEL MOSHELL is a Professor of Digital Media and Computer Science at the University of Central Florida. He founded the Institute for Simulation's Visual Systems Laboratory in 1989, initiated the Digital Media program at UCF in 1995 and was the first Director of the Florida Interactive Entertainment Academy in 2005. He received his Ph.D from Ohio State University in Computer Science. His research concerns the application of computers to learning.

RUDY MCDANIEL is an Assistant Professor of Digital Media at the University of Central Florida. He currently directs the Partnership for Research in Synthetic Environments (PROSE) lab at UCF. His research interests include narrative theory, "serious" video games, knowledge management frameworks, and XML. He has published and presented on the topics of interactive narrative, virtual teams, knowledge management, and the critical analysis of video games. He received his Ph.D. from UCF's Texts and Technology program in 2004.

LI WEI is a Lecturer of Chinese and Asian Studies at Rollins College. As an ethnomusicologist-turned-language teacher, he has a broad interest in music, language and culture. His current research projects include linguistic idiosyncrasy in textbook presentation of culture, appropriation of Buddhism in "world music," and the interplay of music and politics in Post-Mao China. He is currently completing his Ph.D. in Ethnomusicology at Columbia University.References

ALPESH P. MAKWANA is a Modeling and Simulation PhD student at the University of Central Florida. He holds an M.S. in Modeling & Simulation, professional certification in Instructional Design for Simulations, and a B.E. in Computer Science. He is working as a Graduate Research Assistant at the UCF Institute for Simulation and Training. His research interests include collaborative learning in virtual environments, collaborative learning tools design and development, and language learning using Multiplayer Online Role-Playing Games (MORPGs).

References

Aldrich, C. (2005). Learning by Doing: A Comprehensive Guide to Simulations, Computer Games, and Pedagogy in E-Learning and Other Educational Experiences. San Francisco: John Wiley & Sons.

Bioware. (2002). Neverwinter Nights. New York: Infogames.

Flagship Industries (2007). Ventrilo Surround Sound Voice Communication Software. www.ventrilo.com

Forterra Systems (2007). Olive game engine. www.forterrainc.com

Garage Games (2007). Torque Game Engine. www.garagegames.com

Gee, J. P. (2003). What Video Games Have to Teach Us About Learning and Literacy. New York: Palgrave Macmillan.

Geisler, Michael, et al. (2007). Foreign Languages and Higher Education: New Structures for a Changed World. Modern Language Association. Available at the following URL: http://www.mla.org/flreport.

Hughes, C. E.; Moshell, J. M.; Reed, Dean; Chase, Diane; and Chase, Arlen. "The Caracol Time Travel Project," Visual 2000: 3rd International Conference on Visual Computing, Mexico City, 18-22 September 2000. appeared in *Journal of Visualization & Computer Animation* 12(4) 203-214 (2001).

Hughes, C. E.; Moshell, J. M.; Sims, Valerie and Yu, QingYi. "Dynamic Computation of Levels of Detail in Complex Outdoor Environments," Visual 2000: 3rd International Conference on Visual Computing, Mexico City, 18-22 September 2000.

Liu, Xun (2002) New Practical Chinese Reader 2. Beijing: Beijing Language and Culture University Press.

Linden Labs (2007). Second Life. www.secondlife.com

Moshell, J. M.; Hughes, C. E.; Reed, Dean; Chase, Diane; and Chase, Arlen. "Virtual Drama as a Learning Medium: The Caracol Time Travel Project," IEEE VR2000 (Poster Session), New Brunswick, NJ, 18-22 March 2000.

Moshell, J. M. & Mcdaniel, R. (2006). Cast Member Performance Management: Research Report for Fall 2006. Technical Report UCF-SFDM-2006.1. University of Central Florida, School of Film and Digital Media (Orlando, Florida). Available at the followingURL: http://www.cs.ucf.edu/~jmmoshell/techreports/ 2006.1.pdf

Moshell, J. M., Mcdaniel, R., Makwana, A. (2007a). Cast Member Performance Management: Research Report for Spring 2007. Technical Report UCF-SFDM-2007.1. University of Central Florida, School of Film and Digital Media (Orlando, Florida). Available at http://www.cs.ucf.edu/~jmmoshell/ techreports/ 2007.1.pdf

Moshell, J. M., Wei, Li; Makwana, A.; and McDaniel, Rudy (2007b). Role Playing Games and Language Learning. Technical Report UCF-SFDM-2007.2. University of Central Florida, School of Film and Digital Media (Orlando, Florida). Available at: http://www.cs.ucf.edu/~jmmoshell/techreports/ 2007.2.pdf

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

Prensky, M. (2006). Don't bother me mom - I'm learning! St. Paul, MN: Paragon House.

Shaffer, D. W., Squire, K. R., Halverson, R., & Gee, J. P. (2004). *Video Games and the Future of Learning*. Retrieved 3 Jan.2007, from www.academiccolab.org/ resources/gappspaper1.pdf

MMO MODELLING

Dissecting Group Identity in MMOs

Yusuf Pisan
University of Technology, Sydney
http://staff.it.uts.edu.au/~ypisan/
ypisan@it.uts.edu.au

KEYWORDS
Group Identity, Digital Identity, Virtual Environments, Games, MMOs

ABSTRACT

Massively Multiplayer Online Games continue to grow and attract millions of players. While it is generally agreed that the social aspect of MMOs differentiate them from other games, it remains unclear what factors of socialization attract users. We examine group identity in the context of one of the largest MMOs, World of Warcraft, explore how cognitive, behavioural and affective components are exhibited in a guild environment and contribute to a group identity.

INTRODUCTION

Massively Multiplayer Online Games (MMOs) continue to grow and attract millions of players. The impact of MMOs is not just economical. With the average MMO player spending 22 hours/week in the game, MMOs are becoming an important if not the primary form of entertainment and socialization for many people.

MMOs share many similarities with other games including shooting, killing, exploring, winning and losing. What differentiates MMOs is the large number of users and the persistent world the users share with each other. Ducheneaut, Yee, Nickell, and Moore (2006) argue that the role of social activities have been over-estimated and the social side of MMOs are exhibited much more through players having an audience, social presence and spectacle while playing the game mostly by themselves.

Bartle (1996) proposed achievers, socializers, explorers, and killers as the four types of players with distinct motivations. Yee (2006) expanded these categories by examining achievement, manipulation, relationship, immersion and escapism as distinct motivators for players. The social aspect, expressed as socializing or by establishing relationships or through other acts, is what differentiates MMOs from other games. Group identity is defined as members' positive attitudes toward their group (Hinkle, Taylor, Fox-Cardamone & Crook, 1989). The three component of group identity are cognitive, affective, and behavioural components. In the next section we use WoW as an example of a successful MMO and explore its guild structure. In Section 3, we examine the components of group identity and provide examples of how different components are exhibited. Section 7 discusses the preliminary survey we have conducted in WoW.

GUILDS IN WORLD OF WARCRAFT

While forming a guild is relatively easy and only requires getting ten signatures from players, growing and maintaining a guild requires a lot of effort. Users create guilds for a variety of reasons: to keep in contact with friends, to have access to other trusted players they would like to form groups with and most often to have the critical mass to attempt some of the larger end-game instances that require up to 40 people. Guilds will often setup web pages, have forums for discussing strategies and socializing, use signup systems to decide when they will be raiding an instance and manage a guild bank to store items that might be needed later.

Duchenaut et. al. (2006) found that 66% of characters belonged to a guild and 90% of characters over level 43 belonged to a guild. Duchenaut et. al. also found that players in guilds spend more time in game than others and group more often as well. Seay, Jerome, Lee and Kraut (2004) found that on average players were only "somewhat committed" to their guild. A result supported by the large churn rate found by Duchenaut et. al.

Understanding what ties a person to a guild can have serious impact on how MMOs are designed. In the next section we describe components of group identity and demonstrate how different components get expressed in the context of guilds.

COMPONENTS OF GROUP IDENTITY

We adopt Henry, Arrow & Carini (1999)'s model of group identity with cognitive, behavioural and affective components. Group identity as a research area has been explored from various perspectives focusing on different aspects, but attempts to measure it have been limited. Henry et. al. (1999) note:

> Social identity literature emphasizes the cognitive aspect—awareness of a group and self-categorization of oneself as a member (Tajfel, Billig, Bundy, & Flament, 1971). The cohesion literature emphasizes the affective aspect, focusing on interpersonal attraction (e.g., Festinger, Schachter, & Back, 1950; Piper, Marrache, Lacroix, Richardsen, & Jones, 1983; Turner, Hogg, Turner,& Smith, 1984). The common fate literature

emphasizes the behavioral aspect by pointing to the importance of interdependence (Brewer & Kramer, 1986; Chen, 1996).

We expect group identity for WoW players to be exhibited at multiple levels. First, there is the intimate group of friends, possibly friends from real life and other games, who the player groups with. Next, there are the temporary groupings with other players. While a long term relationship may not be established in these temporary groupings, an average user playing 22 hours/week is likely to run into other characters at the same level over time. In fact, school, jobs and other regular commitments can result in regular playing times which make the grouping more likely. The guild the user belongs, and 90% of high level characters would be in a guild, represents the larger formal group that forms players identity. The server community, customs, conventions and habits established in that particular server provides the general background for the guild. At critical times, such as server wide protests or world events, the player interacts with this larger community. Outside the game, official and unofficial forums, web pages and wikis form an even larger community that the user is a part of.

We focus on guild level group identity since 1) joining and remaining in a guild is a voluntary act demonstrating some level of commitment, 2) guilds indicate formal structures and 3) while guild sizes may vary since guild size never exceeds 300, limit imposed by WoW, it is a relatively small group that a player can get to know over time.

COGNITIVE COMPONENT

The cognitive component of group identity is the response to self identity, categorization of oneself as a member of a specific group. The instrument developed and tested by Henry, Arrow & Carini (1999) use the below questions to determine the strength of the cognitive component as a part of a person's group identity. Questions marked with (R) are reverse scored. The questions for each component is spread out in the survey. The question numbers indicate their ordering.

Cognitive
3. I think of this group as part of who I am
6. I see myself as quite different from other members of the group (R)
9. I don't think of this group as part of who I am (R)
12. I see myself as quite similar to other members of the group

In the context of social identity, people might identify themselves as Hispanic, as a mother or through their job title. In the context of MMOs, social categories do not exist. Guild membership is both voluntary and does not allow multiple memberships in the same way social categories allow belonging to multiple groups. Furthermore, the class of the character, such as warrior, mage, priest, etc., determines role in groups and playing style. For a player, important class-specific information on how to best play his character would come from users playing that character whether they are in the guild or not which would be group cross-cutting guild boundaries. However, playing multiple characters would dilute this affinity with one's class as the player can no longer think of himself or herself as belonging to a single class. The race, gender and the profession of a character are rather insignificant to gameplay. Race and gender and looks of the character are set at the very beginning of the game and cannot be changed. It provides visual differences that are often ignored as they do not indicate abilities or talents. When group identity is taken as guild identity, we would expect the cognitive component to be low.

BEHAVIORAL COMPONENT

The common fate of the group as well as interdependence within the group leads to coordinating activities within the group towards common goals. The set of questions developed by Henry, Arrow & Carini (1999) are included below.

Behavioural
2. In this group, members don't have to rely on one another (R)
5. All members need to contribute to achieve the group's goals
8. This group accomplishes things that no single member could achieve
11. In this group, members do not need to cooperate to complete group tasks (R)

Since the primary goal of for forming guilds is to achieve goals that an uncoordinated group cannot achieve, we would expect the behavioural component of the group identity with the guild to be high. In particular, instances that require 25 or 40 people cannot be completed with a pick-up group necessitating guilds. This makes it necessary for players to go back to the same instance week after week.

AFFECTIVE COMPONENT

Cohesion has been defined as the set of forces that act on members to remain in the group. The two forces are 1) the group's attractiveness and 2) the group's ability to help members achieve its goals. The set of questions developed by Henry, Arrow & Carini (1999) are included below.

Affective
1. I would prefer to be in a different group (R)
4. Members of this group like one another
7. I enjoy interacting with the members of this group
10. I don't like many of the other people in this group (R)

While group identification is meaningful at an individual level, cohesion exists at the group level. Henry, Arrow & Carini (1999) argue that interpersonal attraction is a source of group identification and group identification develops as a result of affective bonds among group members. Consequently, development of group cohesion and group identification overlap. Group members that are attracted to each other, enjoy each other's company, would spend more

time together and achieve goals together. The collaboration towards shared goals leads to group interdependence.

We would guess that the initial core group that forms the guild has strong affective component and members have high affinity for each other. As the guild grows and new members join in, we would expect the attraction to new members and attraction among the newer members to not be as strong. Of course, over time the new members could also from affinity groups with each other and with the original group. Ducheneaut et. al (1999) have found that guilds tend to have a core group of people who play longer together and large majority of guilds have a single core group. We would expect results to these questions to vary based on whether the player was a core guild member or not.

PRELIMINARY SURVEY

To test the strength of different component in guild identity, we modified the Henry, Arrow & Carini (1999) survey changing "group" to "guild" in the survey questions. A brief description of the goals and a link to the survey was posted to four official WoW forums, general, off-topic, guild recruitment and welcome forums, during March and April 2007 to contact participants. Participants who chose to click on the survey link received a warning to indicate they were leaving the official WoW pages. Participants were asked to answer the survey questions based on their main character. The first section of the survey asked questions regarding demographics, realm (server) name and approximate amount of time played. The second section of the survey asked social identification questions, shown above, on a 7-point scale. In the third part of the survey participants were given the option to enter freeform comments and include email addresses if they wanted to receive a report on the survey results.

A total of 106 participants completed the survey. We found that the affective component was the highest (6.05) with participants "mostly agree" with the survey statements, followed by behavioral component (5.3) with participants "somewhat agree" and cognitive component (4.7) where participants' average response was between "neither agree nor disagree" and "somewhat agree". While the small sample size makes it difficult to make conclusive statements, affective component seems to be the primary contributor to group identity.

CONCLUSION

MMOs have attracted large number of users making them significant economic, social and cultural artefacts. Players spend an average of 22 hours/week in these environments, indicating a high level of investment in these environments. While it is generally agreed that the social aspect of MMOs makes them special, the factors of social attraction are unclear. In this paper, we have examined the concept of group identity composed of cognitive, behavioural and affective components in the context of World of Warcraft guilds and discussed how these components are exhibited and contribute to the group identity. Our preliminary survey shows that while affective component seems to be the strongest, the results are not conclusive. Understanding factors that strengthen groups is critical for designing MMOs that can maintain and grow their user base.

REFERENCES

Abrams, D., & Hogg, M. A. (1988). *A Social Psychology of Intergroup Relations and Group Processes*: London: Routledge.

Bartle, R. (1996). Hearts, clubs, diamonds, spades: Players who suit MUDs. *Journal of Online Environments, 1*(1).

Brewer, M. B., & Kramer, R. M. (1986). Choice behavior in social dilemmas: Effects of social identity, group size, and decision framing. *Journal of Personality and Social Psychology, 50*, 543-549.

Chen, X. P. (1996). The group-based binding pledge as a solution to public goods problems. *Organizational Behavior and Human Decision Processes, 66*, 192-202.

Ducheneaut, N., & Moore, R. J. (2004). *Let me get my alt: digital identiti(es) in multiplayer games.* In Proceedings of The CSCW2004 Workshop on Representation of Digital Identities, Chicago, IL.

Ducheneaut, N., Yee, N., Nickell, E., & Moore, R. J. (2006). *"Alone together?" Exploring the social dynamics of massively multiplayer online games.* In Proceedings of The ACM Conference on Human Factors in Computing Systems (CHI 2006), Montreal; Canada.

Festinger, L., Schachter, S., & Back, K. (1950). *Social pressures in informal groups*. New York: Harper & Row.

Henry, K. B., Arrow, H., & Carini, B. (1999). A Tripartite Model of Group Identification: Theory and Measuremen. *Small Group Research, 30*(5), 555-581.

Hinkle, S., Taylor, L., D.L., F.-C., & Crook, K. (1989). Intragroup identification and intergroup differentiation: A multi-component approach. *British Journal of Social Psychology, 28*, 305-317.

Piper, W. E., Marrache, M., Lacroix, R., Richardsen, A. M., & Jones, B. D. (1983). Cohesion as a basic bond in groups. *Human Relations, 36*, 93-108.

Seay, A. F., Jerome, W. J., Lee, K. S., & Kraut, R. E. (2004). *Project Massive: A study of online gaming communities.* In Proceedings of CHI 2004, New York.

Tajfel, H., Billig, M., Bundy, R., & Flament, C. (1971). Social categorization and intergroup behavior. *European Journal of Social Psychology, 1*, 149-177.

Turner, J. C., Hogg, M. A., Turner, P. J., & Smith, P. M. (1984). Failure and defeat as determinants of group cohesiveness. *British Journal of Social Psychology, 23*, 97-111.

Woodcock, B. S. An Analysis of MMOG Subscription Growth. Retrieved 1 June 2007, from http://www.mmogchart.com

Yee, N. (2006). The Demographics, Motivations and Derived Experiences of Users of Massively-Multiuser Online Graphical Environments. *PRESENCE: Teleoperators and Virtual Environments, 15*, 309-329.

USING SYNTHETIC PLAYERS TO GENERATE WORKLOADS FOR NETWORKED MULTIPLAYER GAMES

Asif Raja and Michael Katchabaw
Department of Computer Science
The University of Western Ontario
London, Ontario, Canada
N6A 5B7
E-mail: araja4@csd.uwo.ca, katchab@csd.uwo.ca

KEYWORDS

Networked games, multiplayer games, workload generation, synthetic players, bots.

ABSTRACT

The increase in popularity of online games in recent years has motivated research and development efforts into the creation of new algorithms, architectures, and protocols for these games. Generating suitable workloads to permit live empirical testing and experimentation to verify and validate this work, however, is a difficult process, especially if one is working towards massively multiplayer technologies.

To support these efforts, we have developed a framework for synthetic players that generates appropriate workloads for networked multiplayer games. Based on this framework, we have developed a software infrastructure and a prototype synthetic player for the multiplayer adventure game Crossfire. This paper introduces our framework, discusses our implementation efforts, and presents results from initial experiments using our prototype synthetic player. Results to date have been quite promising.

INTRODUCTION

Video games have been projected to continue to enjoy solid growth over the coming years (PricewaterhouseCoopers LLP 2007). Online video games in particular are continuing to grow in popularity, with reports now estimating that 62% of all video game players play games online (NPD Group 2007). As a result, there is a serious need for research and development efforts into creation of new technologies for networked games that provide improved quality of service to players despite the uncertainties and adversities in network performance and reliability that frequently occur over the public Internet (Carlson et al. 2003).

When creating new algorithms, architectures, and protocols for networked multiplayer and massively multiplayer games, a significant challenge comes in the form of verification and validation of research and development results. Traditionally, this has taken the form of simulation, but the increase in availability of and access to sufficient computing infrastructure and network testbeds in recent years (Craven 2006) now makes live empirical testing and experimentation a realistic and attractive option. A key difficulty in doing this, however, is the generation of a suitable workload for the game in question.

Using human players for workload generation is problematic for several reasons. Coordinating a reasonably large number of human players for specific experimentation can be difficult; this is especially problematic when one would like to generate a workload for a massively multiplayer game potentially requiring thousands upon thousands of players. The use of human players also increases the potential for variability and unanticipated influences on experimentation, making experiments difficult to repeat and replicate, and analyses more complicated and susceptible to error. Because of these issues, the use of synthetic players, driven by some form of artificial intelligence, must be considered very seriously.

This paper introduces a new framework for synthetic players, specifically designed with workload generation for empirical testing and experimentation with networked multiplayer games in mind. This framework supports the creation of independent synthetic players that interface with a networked game in the same fashion as its human players would, to realistically and accurately recreate the workload that human players induce on the game. The framework allows synthetic player behaviour to be scripted or generated dynamically, with generated behaviour being either repeatable or different for every gameplay session. The framework is also flexible and robust, allowing it to be used in constructing synthetic players for a wide variety of games in various genres. Furthermore, it provides the monitoring and control features required to support large-scale rigorous experimentation efforts.

Based on this framework, we have developed software and libraries to facilitate the construction of synthetic players for workload generation for networked multiplayer games. As a proof of concept, we have created a synthetic player for the open-source multiplayer adventure game Crossfire (Wedel et al. 2007), and used this to conduct a variety of experiments with this game. This paper discusses these development efforts, as well as initial experiences with using our synthetic Crossfire player to date.

The remainder of this paper is structured as follows. We begin by discussing related work in this area, providing a brief

overview and analysis of this work. We then describe the framework for synthetic players we have developed, and describe how it can be used in workload generation for networked multiplayer games. We then discuss our implementation efforts and our experiences in using our synthetic Crossfire player to date in experimentation. Finally, we conclude this paper with a summary and a discussion of directions for future work.

RELATED WORK

There has been considerable work done towards the application of various principles and techniques from artificial intelligence to creating realistic synthetic players for video games, as discussed in (Rabin 2002) and elsewhere. Many successes have been achieved for a wide variety of types of games, and on-going work in this area is extremely promising.

Unfortunately, the majority of this work is not suitable for workload generation for multiplayer games. To be suitable, these synthetic players would need to be separate and exist outside of the game as a whole, connecting to the game remotely over the network as a human player would, and acting as if it were a human player in this regard. This is not the case, however, in much of the work in this area. In many cases, in fact, the synthetic players rely on abilities and access to information from the game that are generally not available to human players in order to play the game well (Rouse 2005).

There are, of course, many notable exceptions to this. Interesting work has been done towards the use of synthetic players commonly referred to as bots in a variety of games, from commercial, research, and hobbyist perspectives. This includes bots for games such as Quake (Randar 2007), Quake II (Randar 2007, van Lent et al. 1999), Quake III (Randar 2007, van Waveren 2001), Counter-Strike (Booth 2004), Unreal Tournament (Epic Games 2004), Descent 3 (van Lent et al. 1999), and others.

These bots, unfortunately, tend not to be suitable for workload generation in practice either. While they can connect to a game remotely over a network as a human player would, these synthetic players were designed primarily to provide interesting and challenging opponents to the human players of the game, and not as tools to support empirical testing and experimentation. Consequently, they lack many of the features required to properly support workload generation for this purpose, such as monitoring and control capabilities, as examples.

Furthermore, as one can see from the bots listed above, the majority of bots have been created for first person shooter games. These games have relatively set formulae for gameplay, and tend to not support the rich variety of gameplay and interactions available in other genres like role-playing or adventure games. Consequently, there is a need for more robust and flexible approaches to synthetic players to support a wider variety of games. Since most massively multiplayer online games are role-playing games, and these and games from other genres are among the most popular games currently being played (Nielsen Media Research 2007), this is not something that can be ignored.

As a result, in the end, more work is needed to provide synthetic players to support workload generation for networked multiplayer games.

A FRAMEWORK FOR SYNTHETIC PLAYERS

To support the creation of synthetic players for workload generation for networked games, we have developed a general and flexible framework, as shown in Figure 1. This is based on a sense-think-act process that has been used in artificial intelligence, agent development, and game bots, such as the work in (van Lent et al. 1999). The main elements of this framework are described in the sections that follow.

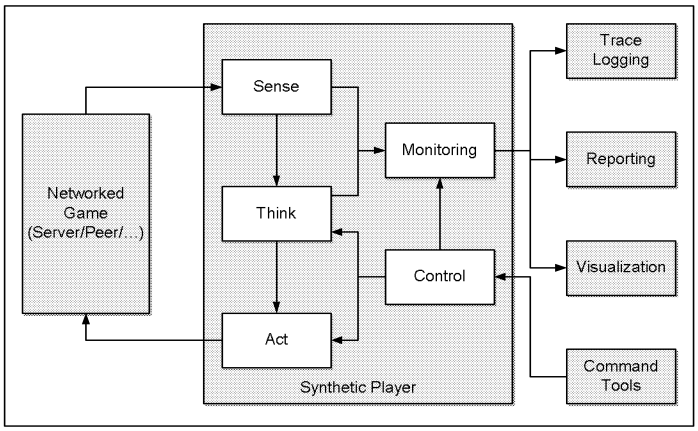


Figure 1: A Framework for Synthetic Players for Workload Generation in Networked Multiplayer Games

Networked Game

The networked game is the game for which workload is being generated by the synthetic player. Ideally, the game should be used without modification from its original form, if at all possible. The networked game can be based on a variety of architectures such as client-server or peer-to-peer; ultimately, it is up to the synthetic player to adapt accordingly to remotely connect and work with the game properly. In the end, the game and other players in the game should be unaware that the synthetic player is not, in fact, a human player after all.

Synthetic Player

The synthetic player is responsible for generating a realistic workload for the networked game in question. It does so by interfacing with the game and playing it in the same manner as a human player would.

The synthetic player consists of two main sets of modules. The first set consists of the sense-think-act logic that enables the synthetic player to properly play the networked game. The second set of modules consists of monitoring and control elements to enable and facilitate experiments using the synthetic player. All of these modules are discussed in the sections below.

Sense

The sense module is responsible for receiving events and messages from the networked game. It uses this information to construct the view of the game and its state that the synthetic player uses in its decision making processes. In theory, this is the same view that a human player would have if it was in the synthetic player's position in the game, although the synthetic player might have access to more information depending on how it interfaces and communicates with the game in question.

Think

The think module is used to encapsulate whatever decision process is used within the synthetic player to guide its actions in playing the networked game and generating its workload. This module relies on the sense module to provide input for the decision process, and has its decisions carried out by passing them onto the act module once they have been formulated.

The decision process used in this module is likely to be some form of rule based system or state machine, but could be virtually any kind of decision making logic. Alternatively, this module could use some kind of scripting system to have the synthetic player generate workload according to a previously scripted sequence of actions. It is also possible to construct a synthetic player with a think module that uses some combination of these approaches as necessary.

Act

The act module is responsible for carrying out the actions decided upon by the think module. It does so by generating appropriate messages and sending them to the networked game according to the formatting and protocol requirements of the game in question. These messages, in the end, are what induce workload on the network game, as the game must receive, process, and respond to these messages appropriately.

Monitoring

The monitoring module is used to observe the synthetic player in playing the networked game and collect and record information about its performance in doing so. Monitoring measures both the workload being generated by the synthetic player and the results of applying the workload to the game. This ensures that the synthetic player is generating the correct workload and provides valuable information that enables qualitative and quantitative evaluation of the experience delivered as a result, as will be discussed further below.

Data being monitored can be collected either from the sense module or the think module. Data elements collected from the sense module are either copies taken from messages received from the networked game, or measurements taken concerning these messages, such as message latency, data volume, and so on. Data collected from the think module, on the other hand, either reports on the synthetic player's decisions or provides the synthetic player's impressions of the game according to its decision logic. The exact data collected, understandably, depends significantly on the game in question.

Data collected through monitoring is directed to modules outside of the synthetic player for a variety of purposes. This includes trace logging, reporting, and visualization, depending on the needs of the experimentation being conducted.

Control

The control module is used to manipulate or tune the behaviour of the synthetic player. This can be done both to influence the sense-think-act logic guiding the decisions made and actions taken by the synthetic player, and to adjust the configuration of the monitoring module, such as what information is collected and how frequently it is collected.

The synthetic player's behaviour in playing the networked game can be directly controlled by manipulating the stream of actions sent to the game by the act module. The synthetic player can also be indirectly controlled by adjusting the decision process used in the think module; for example, by changing internal state, goals, and other elements used in the process. Such control over the synthetic player allows adjustments to generated workloads at run-time, during a test or experiment.

Control is achieved through the use of external command tools that instruct the control module within a synthetic player on what behavioural manipulations are necessary.

Trace Logging

This module is used to create logs of activity from the networked game in real-time, based on data collected by the monitoring module embedded within the synthetic player. The use of the logs depends greatly on the information they contain. For example, a quantitative evaluation of a game can be facilitated by a log consisting of performance measurements over time, such as latency, data volume, anomalies, and so on.

Reporting

The reporting module is used to generate reports of activity from the networked game, typically containing summaries or analyses of data collected by the monitoring module. Again, the use of the reports depends on the information they contain. For example, a report containing summary statistics of various performance measurements, such as those described above, would be quite useful in supporting quantitative evaluations.

Visualization

The visualization module is used to provide a variety of different visual methods of representing data collected by the monitoring module within a synthetic player. These can be as simple as charts or graphs of various data elements, useful for quantitative evaluations, or renderings of the game and its game world, useful for more qualitative evaluations. (In such a case, for example, visualization could provide in essence the same view of the game as would be provided to a human player.)

Command Tools

Command tools are used to exert control over the behaviour of the synthetic player through the use of its embedded control module. Such tools can either be command-line or graphical in nature, depending on requirements for control.

PROTOTYPE IMPLEMENTATION

Based on the framework discussed in the previous section, we have developed software and libraries to ease the construction of synthetic players for workload generation. This includes a layered mapping system as a basis for sense modules to track and record the state of the game world, and a simple and efficient rule based system to be used within think modules to guide decision processes within the synthetic player. Further details of these elements can be found in (Raja 2007).

Using these software elements, we have developed a synthetic player for the multiplayer adventure game Crossfire (Wedel et al. 2007). Crossfire is a fairly mature open-source project that has been developed for several years with an active community of contributors and players. The game has an incredibly rich set of gameplay features allowing its players to interact with each other and the game world in a variety of ways. Crossfire is supported on several computing platforms, including Microsoft Windows, Linux, and many others.

To facilitate our development efforts, we built our synthetic player using the code base for the Linux version of the GTK Crossfire client as a starting point. This allowed us to make use of existing functionality to receive game update messages from a Crossfire server and dispatch actions to the server in our sense and act modules respectively. Since the client is nearly stateless, with almost all game state stored on the server, our sense module made use of our layered mapping system to track the various elements of the game world. (Traditionally, this information would be rendered to the client's graphical interface and not stored at the client.) Human player control was replaced with our own rule based system as a think module.

The original client's graphical interface was kept and preserved to facilitate debugging of the synthetic player, and to act as a rendering system for a visualization module. This allows a human observer to track the game and the quality of the experience being delivered to the synthetic player in real-time. The human player control elements that were replaced by our think module to drive the player were kept for use in a control module to manipulate the synthetic player as necessary during a game. Monitoring, logging, and reporting functions were also added to collect a variety of quantitative performance metrics, including latency and several others.

EXPERIMENTAL RESULTS

To evaluate our approach to workload generation for networked multiplayer games using synthetic players, we conducted a series of tests and experiments using our prototype synthetic player for Crossfire. Initial tests were conducted to verify the synthetic player's ability to play the game and generate a workload, as well as the player's other functionality for monitoring, control, visualization, logging, and so on. After the success of these tests, more thorough performance experimentation was carried out.

Experimentation was conducted in a closed lab environment, making use of 31 workstations connected to the same 100MB switched Ethernet network. Each workstation was powered by a dual core 3.0 GHz Pentium D processor, with 2 GB of physical memory, and Fedora Core 5 Linux as its operating system. One of these workstations was designated to host the Crossfire server for experimentation, while the others hosted synthetic players configured to induce workload on the Crossfire server.

Figure 2 shows performance results from one set of experiments, designed to measure the responsiveness of the Crossfire server as various numbers of synthetic players generated workload. The workload in this case was harsher than one would normally expect, consisting of continuous player motion and conversation with minimal think time between actions. This workload was used as the server proved surprisingly resilient to less intense workloads even with 30 players connected and playing at once. (We are currently investigating how performance holds up with even more synthetic players under lighter workloads, however.)

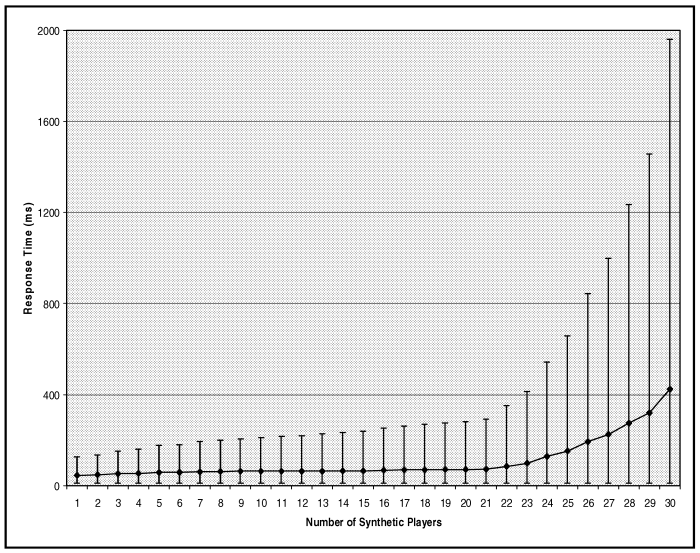


Figure 2: Responsiveness of Crossfire under Increasing Load

The results in Figure 2 were obtained by executing the target number of synthetic players for 10 minutes, during which internal monitoring modules sampled responsiveness every 5 seconds. Responsiveness was calculated as the time required for the Crossfire server to respond to a simple query issued by each synthetic player. Figure 2 presents the mean response time across all samplings from all synthetic players in each 10 minute play interval, along with error bars indicating the minimum and maximum values observed at each load level.

As can be seen from Figure 2, the Crossfire server appeared to do a fairly good job of maintaining a consistent level of performance for up to 21 synthetic players. Past this point, however, response times started to rise dramatically, eventually

reaching an average of more than 400ms at 30 players. This increase in response times is significant and likely more than enough to both impair the ability of players to play the game and affect the ultimate outcome of the game (Armitage 2001). It is also important to note that the variability in response times also increased substantially with more synthetic players in the game, which could further frustrate and annoy human players, had they been playing the game as well.

From a qualitative perspective, we carefully watched the graphical interfaces of the synthetic players as they played the game. As the number of synthetic players in the game increased, we observed a significant increase in the overall delay and jumpiness of the players' displays, especially as the number of players approached 30. This was to be expected considering the increase in response time and response time variation that was measured quantitatively.

In the end, our synthetic players were able to effectively generate workload as instructed. Overall, our synthetic players appear to be very well suited to support testing and experimentation with networked multiplayer games as desired.

CONCLUSIONS AND FUTURE WORK

As networked multiplayer games continue to grow in popularity, it is becoming increasingly important to support research and development efforts in this area to advance the state-of-the-art in both design and technology. Doing so presents many challenges, however, including the generation of suitable workloads to experimentally verify and validate the results of such efforts.

Our current work is aimed at this challenge, providing synthetic players to generate workloads for networked multiplayer games. To this end, we have developed a framework for synthetic players for this purpose, and implemented prototype software based on this framework. In doing so, a synthetic player for the Crossfire multiplayer adventure game was successfully constructed. Experimentation to date using our prototype software has been quite positive and demonstrates great promise for our approach in the future.

There are many possible directions for future work in this area. These include the following:

- Continued experimentation with our synthetic Crossfire player is necessary, particularly with a larger number of players, and under various types of workloads.

- Further development of our synthetic Crossfire player is also possible. This includes adding the option to disable interface visualization elements to permit headless operation, parameterization of personality and behavioural elements to support a wider variety of player types, and extensions to support the incredible variety of gameplay and interactions available within Crossfire.

- Applying our framework to other networked multiplayer games is also important, to help demonstrate its flexibility and suitability in a variety of gameplay situations.

- In particular, applying our framework to a massively multiplayer online game environment is of great interest. This will introduce many challenges in terms of managing and coordinating the large number of synthetic players, as well as in organizing, processing, and analyzing the potentially huge amount of data that can be collected in this kind of scenario.

REFERENCES

Armitage G. 2001. "Sensitivity of Quake3 Players to Network Latency". *Presented at the SIGCOMM Internet Measurement Workshop.* San Francisco, California. (November).

Booth M. 2004. "The Official Counter-Strike Bot". *Presented at the 2004 Game Developers Conference.* San Francisco, California. (March).

Carlson R., Dunigan T., Hobby R., Newman H., Streck J., and Vouk M. 2003. "Strategies & Issues: Measuring End-to-End Internet Performance". *Appeared in Network Magazine.* (April).

Craven D. 2006. Network Testbeds: A Method of Testing Potentially Disruptive Technologies. *MSc Reading Course, Department of Computer Science, The University of Western Ontario.* (February).

Epic Games. 2004. Unreal Tournament 2004. *Published by Atari.* (March).

Nielsen Media Research. 2007. PlayStation 2 Accounted for 42 Percent of Video Game Play in June, Nielsen Reports. *Nielsen News Release* (July).

NPD Group. 2007. Online Gaming 2007: The Virtual Landscape. *NPD Special Report.* (May).

PricewaterhouseCoopers LLP. 2007. Global Entertainment and Media Outlook: 2007-2011. *PWC Report.*

Rabin S. 2002. *AI Game Programming Wisdom.* Charles River Media.

Raja A. 2007. Design and Implementation of an AI Bot for the Adventure Game Crossfire. *MSc Directed Study Report, Department of Computer Science, The University of Western Ontario.* (May).

Randar. 2007. Randar's Bot Page. Available online at: *http://members.cox.net/randar.* (Last accessed August).

Rouse R. 2005. *Game Design Theory and Practice, Second Edition.* Wordware Publishing Inc.

van Waveren J. 2001. The Quake III Arena Bot. *MSc Thesis, Delft University of Technology.* (June).

van Lent M., Laird J., Buckman J., Hartford J., Houchard S., Steinkraus K., Tedrake R. 1999. "Intelligent Agents in Computer Games". *Appeared in the Proceedings of the National Conference on Artificial Intelligence.* Orlando, Florida. (July).

Wedel M. et al. 2007. *Crossfire – The Multiplayer Adventure Game.* (March).

AI TECHNIQUES IN GAMING

The Second Annual Real-Time Strategy Game AI Competition

Michael Buro, Marc Lanctot, and Sterling Orsten
Department of Computing Science
University of Alberta, Edmonton, Alberta, Canada
{mburo|lanctot|sorsten}@cs.ualberta.ca

KEYWORDS
Real-time strategy games, ORTS, real-time AI systems

ABSTRACT

Real-time strategy (RTS) games are complex decision domains which require quick reactions as well as strategic planning and adversarial reasoning. In this paper we describe the second RTS game AI tournament, which was held in June 2007, the competition entries that participated, and plans for next year's tournament.

Introduction

Creating computer-controlled agents for Real-Time Strategy (RTS) games that can play on par with skilled human players is a challenging task. Modern game AI programmers face many obstacles when developing practical algorithms for decision-making in this domain: limited computational resources, real-time constraints, many units and unit types acting simultaneously, and hidden state variables. Furthermore, market realities usually place limits on the time and manpower that can be spent on AI in a commercial game.

Two common ways to circumvent these problems are simply to "cheat": give the AI agent access to more information than the human players are given, and to encode pre-processed human knowledge in the form of scripts. Aside from the obvious issue of unfairness, there are other problems with this solution. The scripts created for the AI hard-codes behaviors, i.e., responses to pre-determined observations, result in predictable decisions. In addition, there is little, if any, long-term planning done during the game.

Today, gamers have higher demands and expectations. They prefer to play online against other humans but not all gamers have access to high-speed Internet connections, and some prefer to play alone. Creating good RTS Game AI is therefore an interesting, challenging, and worthwhile research venture.

There are many scientific motivations for and expectations of research in RTS game AI [2]. Researchers are now developing learning and planning algorithms in this domain [6, 5, 4, 7]. However, different researchers focus on different specific subproblems. The relative quality of the techniques is difficult to assess because they are not exposed to the same empirical evaluations. The spirit of the annual RTS Game AI competition is to encourage the development of these techniques in a common and competitive environment. The quality of various methods can by judged by comparatively measuring the performance of their implementations. Consequently, general conclusions can be drawn from the outcome of the tournament.

Competitions have proved to be excellent way to encourage advancement in AI research, such as when IBM's Deep Blue defeated reigning chess champion Garry Kasparov [3]. Other examples include the RoboCup competition, which has improved techniques in the fields of robotics and multi-agent learning, a computer Go server on which the World's strongest 9x9 Go programs compete, and the AAAI General Game-Playing competition. The purpose of the annual RTS game AI competition is to drive the same progress in real-time AI research.

RTS Games and ORTS

RTS games are tactical simulations involving two or more players, each in control of a growing army of units and bases, with the same goal of conquering the region. Players are faced with a multitude of difficult problems: controlling potentially hundreds of units, limited terrain visibility, resource management, combat tactics, and long-term planning, all of which must be handled simultaneously while considering what the opponent might be doing. Building an AI agent is certainly an ambitious undertaking.

At the very least, a planning agent must control units whose actions lead to potentially varying circumstances. Different unit types can have a variety of different abilities. Coordinating these units both spatially and temporally, in such a way as to maximize their effectiveness, is a nontrivial task. A further complication is that the agent may be subjected to, and have to compensate for, imperfect information (the so-called "fog of war").

A key problem in RTS games is that many decisions necessary for victory (expanding, launching an attack, etc.) carry significant risk. Even smaller decisions such as how to deploy resources and forces can have great consequences. Making these decisions in the wrong circumstances could lead to certain failure.

ORTS

The Open RTS game engine ORTS, available from `www.cs.ualberta.ca/~mburo/orts`, provides a flexible frame-

work for studying AI problems in the context of RTS games. The ORTS engine is scriptable, which allows for game parameters to be easily changed, and new types of games, or subsets of existing games, to be defined.
Units in ORTS are simple geometric primitives located on a fine grid. Map terrain is specified by a grid of tiles of different types and heights. Objects may travel at an arbitrary heading, with collisions handled on the server. Unit vision is tile-based, with different units having a sight range that determines how many tiles away they can see. The vision model also supports "cloaked" units which can only be seen by "detectors".
All ORTS components are free and open-source. Along with the server-client framework, this allows users to create their own AI components capable of acting autonomously or to augment a human player.

The AIIDE RTS Game Competition

The second AIIDE RTS Game competition took place between May 7th and June 1st, 2007. Tournament games were classified into four main categories: cooperative pathfinding, strategic combat, mini RTS game, and tactical combat. Competitors submit different entries for each category. Tournament games are run between the competition entries in the corresponding categories only. In turn, results are classified by game category.

Game 1: Cooperative Pathfinding

In Game 1 the goal is to gather as many resources as possible in a fixed amount of time. The player starts the game with a single base and twenty workers. The workers must travel to mineral patches distributed randomly about the map, mine from them, and return minerals to the main base.
The entire map, which includes the locations of mineral patches, is given to the player as the game starts. The map includes obstacles such as impassible terrain (plateaus) and indestructible roaming "sheep" to complicate the task. No information is hidden from the player, but since simultaneous actions get resolved in random order, there could still be some unpredictable consequences. The challenge then is to efficiently coordinate the paths taken by the agents, which involves avoiding both congestion and planning lag.

Game 2: Strategic Combat

In Game 2, the goal is to destroy as many as of the opponent's bases as possible in a fixed amount of time. Players start with five randomly positioned bases, with ten tanks around each. If all of one player's bases are lost, the game ends and assigns a loss to that player.
As in Game 1, no information is hidden from the player. Plateaus block line-of-sight attacks from tanks, and indestructible sheep roam the map randomly. The challenge in this game is to find attack strategies and formations that maximize the offensive advantage while minimizing the defensive disadvantage. This must be done in a scenario with spatial constraints, so planning when to concentrate or split forces appropriately is key.

Figure 1: Screenshot of Game 1

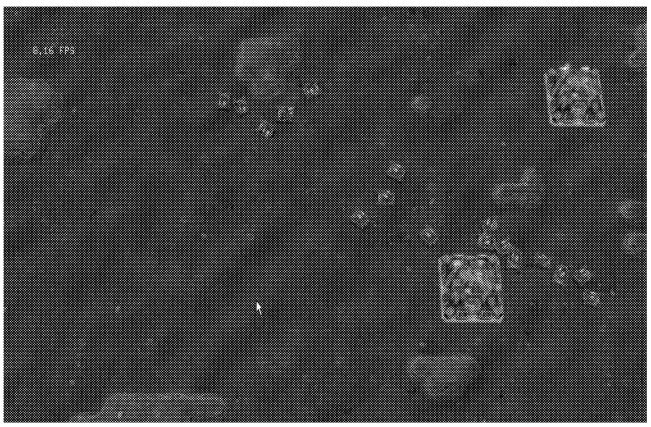


Figure 2: Screenshot of Game 2

Game 3: Mini RTS

Game 3 is a reduced version of a full RTS game. Players start with a single base and a few workers located next to a mineral patch. The only part of the map that the player knows about is what is currently observable by all of the units. The player must use minerals mined by the workers to construct barracks and/or factories, which are used to create marines and tanks. Tanks have greater attack range and power than marines, but also cost more to build.
The goal in this game is to get more points than the opponent. Points are awarded for gathering resources, constructing buildings, creating units, and destroying enemy units and buildings. A player wins automatically if the all of the opponent's buildings are destroyed.

Game 4: Tactical Combat

In Game 4 the goal is simply to eliminate as many of the opponent's units as possible. Players start the game

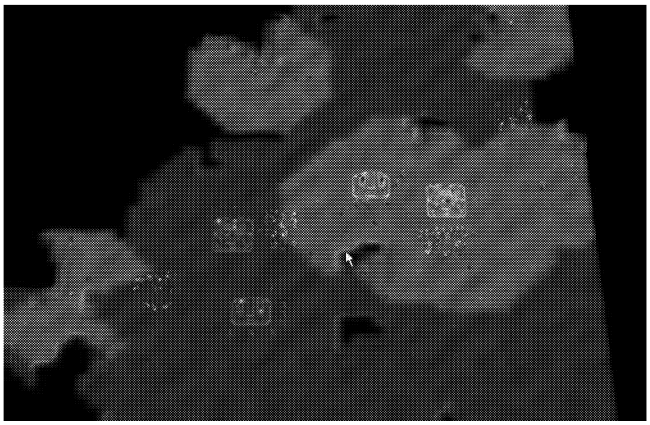


Figure 3: Screenshot of game 3

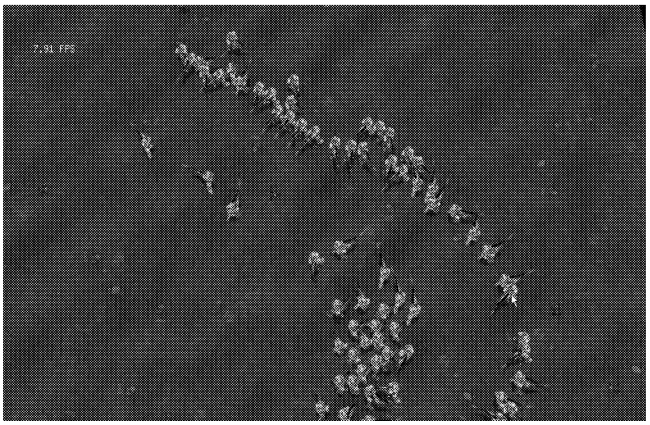


Figure 4: Screenshot of game 4

with 50 marines in random but symmetrically opposed locations. As in games 1 and 2, the entire map and positions of the opponents are known to both players. The map contains no minerals and no terrain obstructions other than roaming sheep. The objective is to find the best combat tactics to defeat the opposing force.

Tournament Setup

On May 7th, competitors were given access to the computers which were to be used during the tournament. The tournament itself was run between May 28th and June 1st. Results were announced June 1st; final results and videos were posted by June 5th.

Each machine was equipped with an Intel Pentium-4 2.4 GHz CPU and 512 MB RAM running 32-bit GNU/Linux with kernel version 2.4.33 and gcc 3.3.6. The tournament management software, which was developed last year, was reused to run a large number of games for each game category.

Authors had access to the tournament computers on which they could upload their programs to test them in individual protected accounts which were frozen just before the tournament commenced. Each participant was asked to send a "random" integer to a member of the independent systems group which also set up the tournament accounts. These numbers were then exclusive-or combined to form the seed of the random number generator used for creating all starting positions. This way, no participant was able to know beforehand what games would be played.

In what follows we present the tournament results and briefly describe the best entries in each category. Videos and more detailed program descriptions are available from the tournament web site [1].

Game 1 Entries

Entries were judged on the average number of minerals gathered after ten minutes over two hundred fifty games. The following table summarizes the results; units are amount of minerals mined.

Entry	Minerals
Warsaw University, Poland (Team B)	6837.5
University of Michigan, USA	6784.9
University of Alberta, Canada	6651.6
Gábor Balázs	5935.6
Naval Postgraduate School, USA	5425.5
Warsaw University, Poland (Team A)	2609.7
Universidad Carlos III de Madrid, Spain	2444.4

Warsaw University (Team B)
Team Leader: Michal Brzozowksi

This entry assigns workers to accessible corners of the closest minerals. Pathfinding is done by searching on a graph based on the terrain. This graph is modified to have one-way edges in key locations. This allows for the formation of lanes between mineral patches and the command center, reducing collisions and allowing this entry to gather more minerals on average than any other entry.

University of Michigan
Team Leader: John Laird

This entry mainly focuses on low-level systems described in [8]. It uses a mining coordinator to assign workers to mineral patches, a pathfinder with heuristics to assist in cooperative pathfinding, and a movement FSM with reactive rules to avoid local collisions with other workers or dynamic obstacles.

University of Alberta
Team Leader: Michael Buro

The entry enumerates routes between the command center and mineral access points, which are locations close enough to at least one mineral to mine. It assigns workers to routes, prioritizing shortest routes. This tends to

cause short round trip times but high congestion which has to be resolved by local obstacle avoidance.

Game 2 Entries

Entries were judged via a round robin tournament with 40 games played per entry pair (320 games played per entry). The following table summarizes the results; the shown percentage is the proportion of wins out of the 320 games played by the entry.

Entry	Wins
National University of Singapore	98%
Warsaw University, Poland (Team B)	78%
University of British Columbia, Canada	75%
University of Alberta, Canada	64%
University of Alberta, Canada [1]	46%
Blekinge Institute of Technology, Sweden	32%
Warsaw University, Poland (Team A)	30%
University of Maastricht, The Netherlands [1]	18%
University of Michigan, USA	6%

National University of Singapore
Team Leader: Lim Yew Jin

This entry makes extensive use of influence maps to represent the strategic state of the map. It intelligently splits its forces into groups based on the situation. These groups attempt to hunt weaker enemy groups, and prioritise taking down units before buildings. Combat efficiency is maximized by lining units up at firing range from the convex hull of enemy groups.

Warsaw University (Team B)
Team Leader: Michal Brzozowski

This entry attempts to fire the most shots and be hit as little as possible by keeping units at the maximum distance from the enemy while still inside their firing range. Units do not advance further until their area is clean, however, they will "rotate" around the enemy position, making room for other allied units to enter firing range. Over time, this can encircle an enemy position and destroy it easily.

University of British Columbia
Team Leader: Zephyr Zhangbo Liu

This entry splits its forces into five squads, and assigns them various targets, such as command centers, enemy groups, or areas. The squads can change targets based on the situation, but not too frequently. Nearby squads can merge if they are not currently occupied. The entry uses clustering to analyse enemy positions, and can rescue its own command centers if they are under attack.

[1] 2006 Entry

Game 3 Entries

Entries were judged by playing 200 games. The results are summarized in the following table. Note that the performance of the University of Michigan's entry suffered from software problems which led to many automatic forfeits.

Entry	Wins
University of Alberta, Canada	89%
University of Michigan, USA	11%

University of Alberta
Team Leader: Michael Buro

This entry uses a hierarchical system of "commanders". Each commander controls multiple units and attempts to complete a specific goal. Commanders can spawn sub-commanders and operate at a specific level of granularity. The entry prioritizes aggressive exploration, expansion and monopolization of the map's resources, so as to inevitably produce marines and tanks faster than the enemy is capable of, and win via sheer numeric strength.

University of Michigan
Team Leader: John Laird

This entry uses a software layer to abstract both the information available and the actions that can be taken, to allow ORTS to be played by a SOAR agent [8]. The agent for this entry followed a plan of building up a force of marines, scouting, and attacking in groups. It is also capable of robustly altering its strategy to compensate for emergencies or unexpected situations.

Game 4 Entries

Entries were judged via a round robin tournament with 100 games played per entry pair (700 games played per entry). The following table summarizes the results.

Entry	Wins
National University of Singapore	99%
University of British Columbia, Canada	75%
Warsaw University, Poland (Team B)	64%
Warsaw University, Poland (Team A)	63%
University of Alberta, Canada	55%
Blekinge Institute of Technology, Sweden	28%
Naval Postgraduate School, USA	15%
University of Michigan, USA	0%

National University of Singapore
Team Leader: Lim Yew Jin

This entry lines up its forces just on the edge of firing range of the convex hull of the set of enemy units. This ensures that a maximum number of units can attack,

while a minimum number of enemies can return fire. A complicated set of rules allows it to efficiently form a tight line formation around an enemy group. This entry is notable for its ability to quickly encircle and destroy enemy squads.

University of British Columbia
Team Leader: Zephyr Zhangbo Liu

This entry uses several small squads to attack the corners of the convex hulls of enemy groups. This lets several units come into range to attack a single enemy, while staying out of the firing range of other enemies. Rules ensure that squads are assigned to attack hull corners in intelligent ways. This entry is also able to quickly break down and destroy many types of enemy formations.

Plans for the 2008 Competition

There are many potential improvements that can be made to the annual RTS game AI competition. This coming year, we plan to address a few in particular:

Simplified Client Software Interface.
Recently, the AI system was restructured as a hierarchy of separate components. The commander interface currently issues commands to each component in a hierarchy, which in turn sends commands to lower level components. Many of the lower level components in the standard ORTS clients need simplification and refactoring. This way, all typical AI functions can be consolidated in one interface and complex behaviors can be compositions of these primary operations.

Opponent Modeling. We are considering adding game categories that allow entries to maintain data on disk across games. Any files created by the entries will be preserved for some proportion of the total number of games in a series against two players. This will allow for learning AI systems to adapt to their opponents, but not to the terrain.

Varying Game Parameters. In the current setting all game parameters such as unit speed and attack range are fixed. To encourage the development of more robust AI solutions we plan to add game parameter randomization, which at game start draws parameter values from specific distributions and therefore forces AI systems to adjust their strategy accordingly.

ORTS Development Roadmap

Several related items within ORTS are also scheduled to be implemented. One addition will be a tweakable graphical user interface. High-level AI behaviors will be attached to graphical components such as buttons and keys, allowing human players to send intelligent commands to a group of units. For example, to execute a spread attack with a group of units, the player will be able to add a customized command which will instruct a group of troops to do so very quickly, without the need to micro-manage their units. Ultimately, we plan to expose human players to the competition entries. Then, we will be able to compare the relative strengths of strategies used by human players versus the strategies employed by the competition entries.
The ORTS project will soon be following a regular release schedule. ORTS will be available in packaged form making it somewhat easier to install and manage. There will be more and better documentation; in particular, a comprehensive, instructional competition guide will be provided to next year's participants. Finally, we hope to provide better support for development and usage of ORTS under Microsoft Windows.

Conclusion

In this paper, we presented the software environment used for the 2007 RTS game AI competition, the results of the tournament, and plans for the future. Many interesting techniques and strategies were implemented and there has been a noticeable improvement in quality of AI techniques in these entries compared to last year's. There has also been more than a two-fold increase in teams and entries than the first competition in 2006. This development is encouraging and we hope the annual RTS game AI competition will continue to attract researchers to this fascinating and complex field.

REFERENCES

[1] The 2007 ORTS RTS Game AI Competition. `http://www.cs.ualberta.ca/~mburo/orts/AIIDE07/`.

[2] M. Buro. Real-time strategy games: A new AI research challenge. In *IJCAI*, pages 1534–1535, 2003.

[3] M. Newborn. *Deep Blue: An Artificial Intelligence Milestone*. Springer, 2003.

[4] S. Ontañón, K. Mishra, N. Sugandh, and A. Ram. Case-based planning and execution for real-time strategy games. In *Proceedings of the Seventh International Conference on Case-Based Reasoning (ICCBR)*, 2007.

[5] M. Ponsen, P. Spronck, H. Munoz-Avila, and D.W. Aha. Automatically generating game tactics through evolutionary learning. *AI Magazine*, 27(3):75–84, 2006.

[6] F. Sailer, M. Buro, and M. Lanctot. Adversarial planning through strategy simulation. In *IEEE Symposium on Computational Intelligence and Games (CIG)*, 2007.

[7] M. Sharma, M. Holmes, J.C. Santamaria, A. Irani, C.L. Isbell Jr., and A. Ram. Transfer learning in real-time strategy games using hybrid CBR/RL. In *IJCAI*, pages 1041–1046, 2007.

[8] S. Wintermute, J. Xu, and J. E. Laird. SORTS: A human-level approach to real-time strategy AI. In *AIIDE*, 2007.

Player Modeling using Knowledge Transfer

Guy Shahine
DigiPen Institute of Technology
5001-150th Ave NE
Redmond, WA 98052
USA
E-mail: gshahine@digipen.edu

Bikramjit Banerjee
School of Computing
The University of Southern Mississippi
118 College Drive
Hattiesburg, MS 39406
E-mail: Bikramjit.Banerjee@usm.edu

KEYWORDS

Transfer learning, Player modeling, Influence diagrams

ABSTRACT

We propose a technique for creating reusable models of other agents (software or human) in a shared environment, with application to video game AI, and AI in general. In particular, we build upon an existing technique for agent modeling using *influence diagrams*, and propose a method to transform it into a reusable model. Such models can be used effectively in reasoning by AI characters (NPCs) for predicting the behavior of human players, for collaboration or competition in *different* tasks. We show experiments in two collaborative tasks (anti-air defense and predator-prey) that clearly demonstrate the reusability and efficiency of our modeling technique across different tasks.

INTRODUCTION

Knowledge transfer (Konidaris and Barto, 2007; Banerjee and Stone, 2007) is a relatively new technique in AI, that is useful in incremental learning of reusable skills and general knowledge. The key idea is to reuse knowledge acquired in previous tasks (called "source") to learn/solve new, but related tasks (called "target") in order to

- offset initial performance in the target task, compared to learning/solving from scratch
- achieve superior performance faster than learning from scratch, when considering learning problems

This new trend in AI research seeks to overcome the long-held tradition of developing specialized systems for given tasks. Instead, we seek to develop life-long learning systems that can map representations and knowledge efficiently from one task to another, and bootstrap performance in the newer tasks by exploiting previously acquired knowledge/skills from a different setting. A motivating example can be as follows: Suppose an agent (NPC) learns to coordinate with a human player in the task of jointly intercepting several incoming missiles; i.e., successful coordination will result in the minimal (perhaps 0) damage to assets from the missiles. Can this agent reuse any of this knowledge in coordinating with the same player in a *different* task, such as coordinated hunting of a prey? Traditional AI and Multi-agent systems literature treats the second task as a new task and embarks on learning a coordination policy from scratch.

The major attraction of knowledge transfer in game AI is the dramatic impact it can have on learning speed, which has so far proved to be a bottleneck for most mainstream machine learning technique in games. No matter what baseline learning technique we choose, knowledge transfer can make it practical in complex game scenarios by exploiting knowledge acquired from previous tasks.

In this paper we focus on the problem of learning models of other agents/players in a domain, and explore the beneficial impact of knowledge transfer on this problem. There may be many motivations for studying the problem of agent modeling. In collaborative domains where communication is expensive, unreliable, and/or prone to interception by adversaries, modeling can be an invaluable tool in predicting the behavior of team-mates, to act in a coordinated fashion (Carmel and Markovitch, 1996). Another major motivation for agent modeling comes from the field of games, especially video games itself. Prohibitive search cost has kept deep searches in the strategy space in game-trees (especially ones with large branching factors, such as chess, or any video game) outside the purview of all but the simplest of games. However, the ability to predict other agents' choices can greatly reduce the branching factor at the search-tree-nodes corresponding to the choices of the other agents. This could enable an agent to only focus on its choices and consequently, search deeper/faster.

We build upon a previous work by Suryadi & Gmytrasiewiez (1999) that outlines a simple technique for learning the decision function of another agent from observations, using an *influence diagram* model (see later section). We argue that the structure of this model does not lend itself to reuse. It is suited only for the problem at hand, and if we were to deal with the same agents/players in another task, we would need to acquire a new model in the new task, all over again. We then show how this model can be transformed into a reusable one, by decomposing the environment-dependent components into two finer parts – one of which is dependent on the agent (its intrinsic character assumed to be unchanged from one task to another; hence transferable across tasks) and the other still depends on the current task. We show that this decomposition allows the second part (task dependent) to be readily deducible from the environment, and may not need to be learned. As a result, our modeling approach enables reuse/transfer of models in many different tasks involving the same set of agents/players.

Essentially, we propose a technique for acquiring reusable models of other agents, through interactions in a variety of

tasks. While the tasks can be widely differing, we assume that the other agents/players (teammates or adversaries) are the same across different tasks. This assumption can be appropriate in many applications. For instance, in video games it is reasonable to assume that a Non-Playing Character (NPC) needs to collaborate or compete with the *same* human player(s) in different tasks, such as collaborative defusing of Improvised Explosive Devices (IEDs), joint patrolling, hunting etc. Consequently, we can

- personalize NPCs to human players, creating the illusion of another "player" rather than a software, and promoting empathy for NPCs,
- allow agents to incrementally model a human player through several tasks, creating an increasingly complete picture of the latter's preferences, styles etc.,
- allow reuse of agent's code and knowledge across different games, thus reducing development time. Human players often do not prefer novice opponents to sophisticated ones, every time he enters a new game. This is especially true when he himself is able to draw on experiences acquired from previous games.

RELATED WORK

Suryadi and Gmytrasiewiez (1999) train an agent to learn a model of another agent in a multi-agents system where the other agent can be either automated or human. The framework makes use of influence diagrams as a modeling representation tool and three strategies are used to create a new model of the other agent, which are based on learning its capabilities, beliefs and preferences, given an initial model and the agents' behavior data. Consequently, through the agents' behavior data, things might change where some capabilities are added or removed, and even some beliefs are altered but the task cannot be switched, i.e. we cannot move to another task while retaining what the agent has learned so far. Although we base our current work on this method, we provide significant extension for reusability of the learned model in other tasks.

Transfer learning has received most attention in the realm of reinforcement learning. The options framework defined by closed loop policies for taking an abstract action over a period of time provides an ideal foundation for knowledge transfer. Examples of options include picking up an object, going to lunch, and traveling to a distant city, that involve specific sequences of primitive actions that can be activated only in certain states. Options enable temporally abstract knowledge and action to be included in the reinforcement learning framework in a natural and general way (Sutton et al. 1999). Consequently, options framework provides methods for reinforcement learning agents to build new high level skills. However, since options are learned in the same state space as the problem that the agent is solving, they cannot be readily used in other problems/tasks that are similar but have different state space. Konidaris and Barto (2007) introduce the notion of learning options in *agent space* – the space generated by a feature set that is present and retains the same semantics across successive problem instances – rather than in *problem space*. Agent-space options can be reused in later tasks that share the same agent-space but have different problem-spaces. Our approach in this paper bears some similarities to this idea of agent-space since we essentially convert a task dependent model language into a player/agent dependent model language. In a recent work (Banerjee and Stone, 2007) we have provided a transfer learning method for reinforcement learning agents in the General Game Playing (GGP) domain. In this paper, we carry forward the lessons learned to impact a new domain, viz., player modeling for other genres of games (RTS, FPS etc.)

While humans effortlessly use experience from previous tasks to improve their performance at novel (but related) tasks, machines must be given precise instructions on how to make such connections. Roy and Kaelbling (2007) describe a Bayesian model based prediction scheme in a meeting-scheduling domain. They provide a hierarchical extension to a Naïve Bayes model that can incorporate data from many different users and make predictions for another user, thus exploiting previous experience. In contrast, we use an influence diagram model and instead of relying on model combination, we seek model decomposition to identify user/player specific components that can then be transferred to a new task. Furthermore, we study our technique in simulated game scenarios with an eye toward future application to video games.

INFLUENCE DIAGRAMS

An Influence diagram (Howard and Matheson, 1984) is a graphical scheme of knowledge representation for a decision problem. It may be viewed as an extension to a Bayesian (Charniak, 1991) or belief network (Pearl, 1988), with additional node types for decisions and utilities. Influence diagrams have three types of nodes: nature node (a.k.a. chance node), decision node, and utility node. Nature nodes correspond to decisions in nature, as in belief networks. They are represented in the diagram as ovals and associated with random variables or features, which represent the agent's possibly uncertain beliefs about the world. Decision nodes holds the choice of actions an agent has, thus represents the agent's capabilities and they are represented as rectangles in the diagram. The agents' utility functions are specified using utility variables, represented as diamonds in the diagram. The links between the nodes summarize their dependence relationships. Evaluation of the influence diagram is done by setting the value of the decision node to a particular choice of action, and treating the node just as a nature node with a known value that can further influence the values of other nodes. Conditional Probability Tables (CPTs) are associated with nature and decision nodes, giving $P(child \mid parents)$. An algorithm for evaluating influence diagrams can be found in Russell and Norvig (1995).

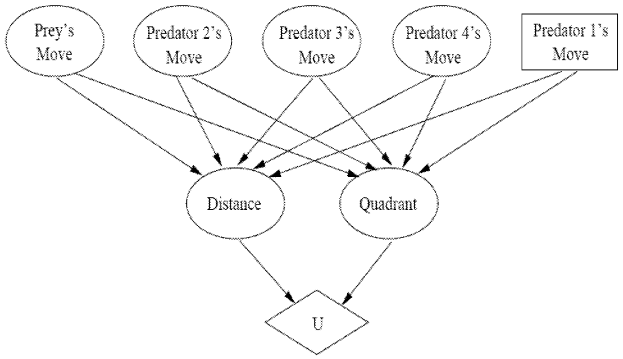


Figure 1: An Influence Diagram model for the Predator-prey task (described later)

Figure 1 above shows an influence diagram used as a basis in the predator/prey task discussed later. Here four predators are trying to jointly catch a prey by closing in on it and surrounding it. In this example, each of the predators 2 thru 4 decides on its next move by taking into consideration two factors: its distance from the prey and the current location of the other predators on the board (whether they are on the same quadrant or spread out, relative to the prey). Based on his observation of these factors, Predator 1 tries to make a prediction for each of the other predators using this influence diagram, and then selects an action that will lead to the best coordination with his predictions.

Below we present another example of a Multi-Agent Influence Diagram (MAID), although this is not a coordination task.

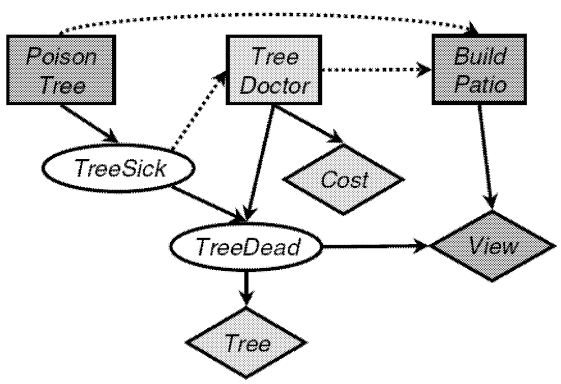


Figure 2: A MAID for the Tree Killer example; Alice's decision and utility variables are in dark gray and Bob's in light gray

In the Tree Killer Example (Koller et al., 2003), Alice is considering building a patio behind her house, and the patio would be more valuable to her if she could get a clear view of the ocean. Unfortunately, there is a tree in her neighbor Bob's yard that blocks her view. Being somewhat unscrupulous, Alice considers poisoning Bob's tree, which might cause it to become sick. Bob cannot tell whether Alice has poisoned his tree, but he can tell if the tree is getting sick, and he has the option of calling in a tree doctor (at some cost). The attention of a tree doctor reduces the chance that the tree will die during the coming winter. Meanwhile, Alice must make a decision about building her patio before the weather gets too cold. When she makes this decision, she knows whether a tree doctor has come, but she cannot observe the health of the tree directly. The Multi-Agent Influence Diagram (MAID) for this scenario is show in Figure 2.

CAPABILITIES

In the following, we use the words "learner", "modeler", and "agent" interchangeably, and refer to the target of modeling as "player", meaning the human player. As in Suryadi and Gmytrasiewicz (1999), a player's capabilities are formed by the set of actions ($a_i, i = 1,..., M$) that it can perform to interact with the world. These actions will be represented in the influence diagram as possible value assignments to the player's decision. Initially, only the known actions are available to form the CPTs, but if a player is observed to employ other (previously unknown) actions, the set can be expanded to include these new actions. Conversely, if the player somehow loses the ability to perform a certain action, that can be determined by its absence in the observation history, especially if the player consistently picks the next most likely action (according to the model that the agent maintains). The missing action can then be deleted from the player's capability.

In Figure 1, each of the predators can move to one of the four cells surrounding its current location. The modeler/learner (Predator 1) can formulate the CPTs for the influence diagram using empirical observations of its teammates' actual moves. Since some available actions may not be known (or pertinent) to the modeler, the latter should ideally account for such contingency using an "*Other*" action that bunches such actions together. For instance, if a certain predator can move to a diagonal neighboring cell unlike what the modeler expects, that capability may become crucial to coordination and needs to be accounted for. See Suryadi and Gmytrasiewicz (1999) for a more detailed discussion of handling agent capabilities in model building.

PREFERENCES

Each player has his own preferences depending on the problem, and takes into consideration a set of features which influence his decisions. Let $X = \{X_1, X_2, \ldots, X_N\}$ be such a set of features. In Figure 1, X is the set of parents of the utility node (U), i.e., the nodes *Distance* and *Quadrant*. As in Suryadi and Gmytrasiewicz (1999), we assume that utility is a linear function of these features

$$U = w_1 x_1 + \ldots + w_N x_N$$

where w_k are unknown weights measuring the influence of feature X_k on the utility, and x_k is a value of X_k. The assumption of linear contribution of x_k is a choice being made for simplicity; a different choice might be necessary for certain domains. E.g., if x_k stands for money, then

$\log(x_k)$ might be more suitable given the sub-linear nature of money's utility (St. Petersburg paradox).

It has been shown in Suryadi and Gmytrasiewiez (1999) that the optimal decision of the player based on the model is given by

$$A^* = \arg\max_{a_i} \sum_{k=1}^{N} w_k \chi_k^i \quad (1)$$

where a_i are the actions available to the player, and χ_k^i denotes the expected value of feature X_k given a_i and background evidence E, i.e.,

$$\chi_k^i = \sum_l x_{k,l} P(X_k = x_{k,l} \mid a_i, E)$$

where $x_{k,l}(l = 1,..)$ denote different possible values of X_k. The probabilities are available from the CPTs of the influence diagram and are assumed to have been already learned, or given. In the predator-prey example in Figure 1, the probability distribution is always pure (e.g., the distance between the next position of the predator after choosing movement direction a_i, and the current position of the prey has a single possible value), whereas it can be a mixed distribution in general.

Using equation (1), the modeler predicts that the player is going to execute action A^*, and verifies if this matches the actual action that the player chooses. If there is a mismatch then it should adjust the weights by applying a well-known gradient descent technique, viz., delta rule (Widrow and Hoff 1960). The idea is to minimize a cost function based on the error signal so that each weight adjustment brings the actual output value closer to the desired value. A commonly used cost function is the mean-square-error criterion:

$$E(t) = \frac{1}{2} \sum_{i=1}^{M} e_i^2(t)$$

where $e_i(t)$ is the error signal generated by comparing the agent's prediction and the player's actual decision. So

$$e_i(t) = d_i(t) - y_i(t)$$

where $d_i(t)$ is the player's decision at t and $y_i(t)$ is the agent's prediction.

Functionally, this model for weight learning can be considered to be a neural network where χ's are the inputs and $y_i(t)$ are the outputs, as shown in Figure 3. In this figure, v_i is the expected utility of the ith action, a_i, and the sigma (not the sigmoid function common in neural nets) module performs the maximization process of equation (1). Now according to the delta rule, the weight adjustment is proportional to the product of the error signal and the input unit. We require normalization of the χ before presenting them to the neural network, so that the resulting weights are normalized as well (Suryadi & Gmytrasiewiez, 1999). The normalization is given by:

$$\aleph_k^i = \frac{\chi_k^i}{\sqrt{\sum_{r=1}^{M} \sum_{s=1}^{N} (\chi_s^r)^2}}$$

Now,

$$\Delta w_k = -\eta \frac{\partial E(t)}{\partial w_k(t)} = -\eta \sum_{i=1}^{M} \frac{\partial E(t)}{\partial e_i(t)} \cdot \frac{\partial e_i(t)}{\partial w_k(t)}$$

$$\Delta w_k(t) = \eta \sum_{i=1}^{M} e_i(t) \aleph_k^i(t)$$

where $k = 1,...,N$ and η is a constant denoting the learning rate.

Neural Network Background

In this subsection, we provide a description of neural networks for the sake of clarity. Readers familiar with this machine learning technique may skip this subsection.

A neural network is made of basic units arranged in layers. The first layer is the input layer and in our case it is formed from several inputs (depending on the problem) represented by the normalized expected values $\aleph_k^i$. The last layer is the output, and in our model there is an output for each possible action, although only one (the modeler's prediction by equation (1)) will be activated at any time. The intermediate layers (if any) are called the hidden layers. The input information is fed to the first layer and then propagated to the neurons of the second layer for further processing. The result is propagated to the next layer and so on until the last layer is reached. The goal of the network is to learn or discover some association between input and output, or to analyze, or to find the structure of the input pattern. The learning process is achieved through the modification of the connection weights between units. These weights, called synaptic weights multiply (i.e. amplify or attenuate) the input information: A positive weight is considered excitatory, a negative weight inhibitory.

Each of these units is a simplified model of a neuron and transforms its input information into an output response. This transformation involves two steps: First, the activation of the neuron is computed as the weighted sum of it inputs, and secondly, this activation is transformed into a response by using a transfer function. Formally, if each input is denoted xi, and each weight wi, then the activation is equal to $a = \sum x_i w_i$ and the output denoted y is obtained as $y = f(a)$.

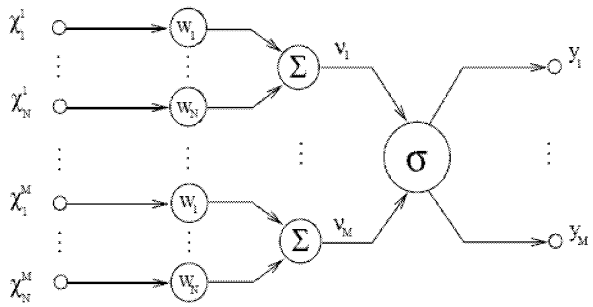


Figure 3: Neural Network Architecture

The architecture of the network, along with the transfer functions used by the neurons and the synaptic weights, completely specify the behavior of the network. Our specific architecture is shown in Figure 3. Notice that the neural units are expanded for clarity and that there is no hidden unit.

Neural networks are adaptive statistical devices. They can change iteratively the values of their parameters (i.e., the synaptic weights) as a function of their performance. These changes are made according to learning rules which can be characterized as supervised (when a desired output is known and used to compute an error signal) or unsupervised (when no such error signal is used).

The Widrow-Hoff rule (1960), a.k.a. gradient descent or Delta rule, is the most widely known supervised learning rule. It uses the difference between the actual output of the network and the desired output as an error signal for units in the output layer. Units in the hidden layers cannot compute directly their error signal but estimate it as a function (e.g., a weighted average) of the error of the units in the following layer. This adaptation of the Widrow-Hoff learning rule is known as error backpropagation. With Widrow-Hoff learning, the correction to the synaptic weights is proportional to the error signal multiplied by the value of the activation given by the derivative of the transfer function. Using the derivative has the effect of making finely tuned corrections when the activation is near its extreme values (minimum or maximum) and larger corrections when the activation is in its middle range. Each correction has the immediate effect of making the error signal smaller if a similar input is applied to the unit. In general, supervised learning rules implement optimization algorithms akin to gradient descent techniques because they search for a set of values for the free parameters (i.e., the synaptic weights) of the system such that some error function computed for the whole network is minimized (Abdi et al. 1999).

OUR CONTRIBUTION

Notice that the factors (viz., distance, quadrant in Figure 1) that affect the player's utility (and hence his decision) change from one problem to another. In the anti-air defense task, the relevant factors are the damage that a missile can cause and the cost of intercepting one. The number of influencing factors (*N*) is also subject to change. As a result the weight vector learned in one task may be completely useless in another, even when modeling the same player. In a video game, this would necessitate learning a new weight vector in every new team-task involving the same set of agents/players. This is surely wasteful and time-consuming, and a major reason behind the impracticality of machine learning in games.

We argue that there are some characteristics that are intrinsic to a player and do not change from one task to another. E.g., his risk-sensitivity, team-spirit and other similar traits can be considered to be a constant. A modeler might consider a fixed, given set of traits $T = \{t_1, t_2, ..., t_S\}$ that reasonably capture any player's disposition. The way to choose the number of traits and their actual meaning is a design issue. The relative importance of these traits (let's say p_j for t_j, where we assume that the t_j's are given but the p_j's are the ones that has to be learned) to a player may also be considered fixed (but unknown) for a given genre of tasks, such as tasks in an RTS game. However, the effect of these traits on the decision factors (distance, quadrant etc) can vary from one task to another. For instance, team spirit can prompt a player to try to increase quadrant value[1] (in predator-prey task), while it may prompt the players to decrease damage value (in anti-air defense task), since damage occurs to shared properties. We model the influence factor between the j^{th} trait (t_j) and the k^{th} decision factor (X_k) as a three-valued (-1, 0 or 1) variable, f_k^j because we are only interested in capturing whether the j^{th} trait increases, decreases or leaves unchanged the decision variable X_k. Combined with the importance of a trait (p_j for t_j), we can directly relate to the weight of a decision variable using the following equation:

$$w_k = \sum_{j=1}^{j=S} p_j f_k^j \qquad (3)$$

Substituting this in Equation (1), we get the action decision of the player in terms of his traits. This reduces the learning task from acquiring the (old) weights w_k, to the (new) weights p_j. The major gain of this decomposition is that the weights to be learned are no longer associated with task specific factors (such as distance, quadrant in predator-prey vs. damage, cost in anti-air defense). Instead, they are now associated with the traits of a player that are fixed over the entire genre of tasks. Note from equation (3) that the right-hand-side does still include the task-specific factor f_k^j, but this can now be easily determined from the task (one of three possible values) and the semantics of the decision factors. For instance, team-spirit should *increase* quadrant value in predator-prey, but *decrease* damage value in anti-air defense. As a result, if we can acquire p_j in one task,

[1] The quadrant value is high if players are spread out in different quadrants, but low if they are concentrated on one or a few, since this makes it harder to catch the prey.

we can reuse this value without learning it again in a new task. Now Equation (3) will let us choose the values of f_k^j in a way that the weight w_k of a certain trait would make sense with the problem that we're trying to solve.

It may seem at this point, that the above formulation of the learning problem makes learning unnecessary after the first task. After all, if we can acquire all the trait weights ($p_j, j = 1,..., S$) in one task, what remains to be learned in the next task? There are at least two reasons why this is not the case. Firstly, all trait weights may not be learnable in any one given task, because some f_k^j may be 0. If f_k^j is 0 for all *k*, then p_j is basically irrelevant to the current task, and hence cannot be learned in the current task. In other words, a task may only invoke a proper subset of the set of S traits designed beforehand; so the rest cannot be learned in that task. The second reason is more subtle and will be discussed in the next subsection, after we have introduced the new weight update rule.

The new learning rule

The new weight learning rule for our formulation can be derived in the same way as in Suryadi and Gmytrasiewicz (1999). Once again, gradient descent on the error function $E(t)$ gives us the necessary change to the new weight p_j by noting that

$$\frac{\partial e_i(t)}{\partial p_j} = \sum_{k=1}^{k=N} f_k^j \aleph_k^i(t) .$$

Plugging this value in Equation (2) in place of $\frac{\partial e_i(t)}{\partial w_k(t)}$, we get

$$\Delta p_j = \eta \sum_{i=1}^{i=M} e_i(t) \sum_{k=1}^{k=N} f_k^j \aleph_k^i(t) \qquad (4)$$

This is the new weight update rule.

Now it is possible that f_k^{j1} and f_k^{j2} are the same for all decision variables X_k, since they have a very limited set of possible values. Then, by Equation (4),

$$\Delta p_{j1} = \Delta p_{j2} .$$

This constrains the joint dynamics of (p_{j1}, p_{j2}) such that they may converge to values that are completely different from their target values. In particular, they will converge to identical values if their initial values are identical, even though their target values may be different. Nevertheless, Equation (3) will be preserved and p_j's will still give the correct target values of w_k. However, since convergence to the target p_j cannot be guaranteed, the values learned in one task may not be accurate for another task. *We claim that these inaccurate values are still a better point for initialization in the next task, than simple default initialization.* We verify this intuition empirically.

EXPERIMENTAL RESULTS

We have used two different tasks for testing the relative efficiency of our approach, compared to Suryadi and Gmytrasiewicz (1999). These are the anti-air defense task, and the predator-prey task, described below. The learning agent works in a team with exactly one other player in both of these tasks. Target weights that simulate the player's decision are chosen, but are not made known to the agent. The agent must learn these weights from repeated joint interaction with the environment, and the resulting observations of the player's action choices. The overall plan

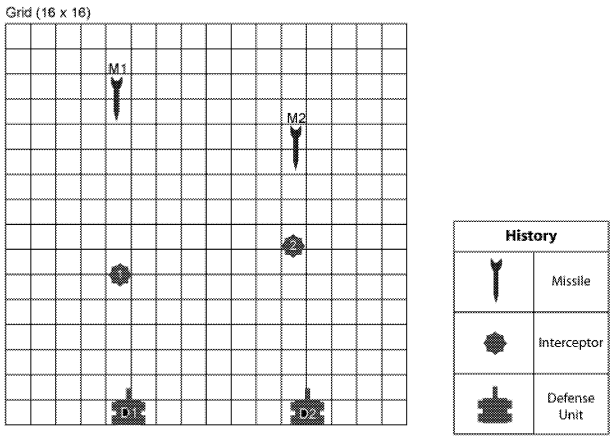


Figure 4: The Anti-air defense task

is to make the agent learn some trait weights in the anti-air defense task, and use these for initialization in the predator-prey task. The trait weights that were not learned in the former, can be initialized to some default values in the latter. We then compare the learning rate with the baseline method (from Suryadi and Gmytrasiewicz (1999)) which must learn from scratch the player model in the latter task. We will consider our knowledge-transfer approach a success if the learning rate is superior to the baseline method. Two particular hallmarks of this superiority are

- a difference in the initial performance; our approach should have a lower initial error by virtue of informed initialization
- a difference in the convergence value; our approach should have lower error on convergence, i.e., it should learn a better model faster than in Suryadi and Gmytrasiewicz (1999).

Anti-air Defense task

This is a 16×16 grid-world adopted from Suryadi and Gmytrasiewicz (1999), with two agents, D1 and D2, the learner and the human player, as shown in Figure 4. In every round of the game, two missiles, M1 and M2 are fired and

the agents must shoot down these missiles with interceptors. If missed, a missile will cause damage proportional to its size. An agent incurs a cost proportional to the accuracy and efficiency with which he intercepts a missile. Without communicating, the agents must choose different missiles to intercept, lest one of the missiles makes it through. Hence, there are two actions available to each agent (M1 or M2) and two decision variables, X_1 = Damage, and X_2 = Cost.

Predator-prey task

This is an 8×8 grid-world with two predators, the learner and the human player, shown in Figure 5. There is a prey that executes a random walk and the predators must catch the prey by either reaching grid-cells neighboring the preys that are on opposite quadrants relative to the prey, or by cornering it. Each agent can select one of 5 actions, viz., go north, west, south, east or stay put. Again for simplicity, there are just 2 decision variables, X_1 = Distance, and X_2 = Quadrant. Distance is the distance between the prey and player if the player executes a chosen action. Quadrant is a score with 3 possible values, viz., 0 (if both predators are on the same quadrant relative to the prey if the player executes the chosen action), 1 (if they are on different quadrants but same side/half of the prey) and 2 (if they are on diametrically opposite quadrants relative to the prey). Therefore, higher values of Quadrant and lower values of Distance are more conducive to catching the prey. In order to surround the prey the learner needs to correctly predict the player's next move.

Experiments and analysis

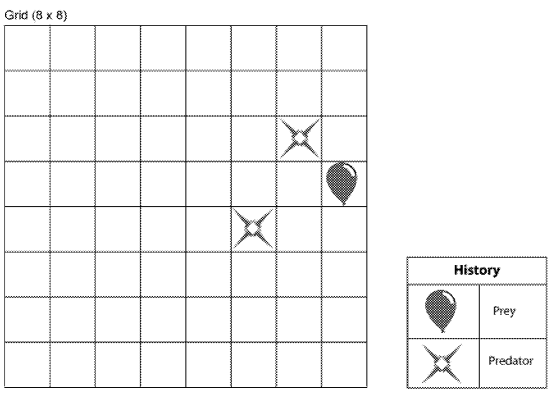


Figure 5: The Predator-prey task

For simplicity, we assume a small set of only 3 traits. To simulate the player's decisions without the involvement of a real human player, we used the values $[p_1, p_2, p_3] = [0.95, 0.272, 0.13]$ which is a normalized vector. Since we assume that the *p*-values are available (albeit not to the modeler), the semantics of the traits are immaterial in this paper; so we leave the traits unnamed. We choose these (and *f*) values such that the resulting weights (from equation (3)) make sense with respect to the features X_k. Considering a specific set of named traits will be necessary in actual human trials where the *p*-values will not be available for experimental validation.

In the anti-air defense task, we used $[f_1^1, f_1^2, f_1^3] = [-1,0,0]$ for X_1 (i.e., Damage) and $[f_2^1, f_2^2, f_2^3] = [0,0,1]$ for X_2 (i.e., Cost). This effectively excludes t_2 from this task, so that p_2 is impossible to learn here. Our simulations show that the agent is able to learn p_1 and p_3 to a reasonable accuracy (0.98 and 0.18 respectively, after 1000 iterations with $\eta = 0.5$), but p_2 stays at the default initial value[2]. This gives the learner a partial picture of the player's disposition that it then leverages in the next task.

In the predator-prey task, the learner initializes p_1 and p_3 to the values it had learned in the previous task, but initializes p_2 to 0 (the default value, since it was not learned). The influence factors in this task were chosen as $[f_1^1, f_1^2, f_1^3] = [-1,1,1]$ for X_1 (i.e., Distance) and $[f_2^1, f_2^2, f_2^3] = [0,1,1]$ for X_2 (i.e., Quadrant). Thus all three traits are needed in this task. The agent then learns all 3 trait-weights (using Equation 4) over 1000 iterations, making a prediction in each iteration based on the current weights and tallying it with the observed choice of the player. A count of the errors in prediction is kept and a cumulative average of these errors is plotted in Figure 6 (titled "with transfer"), averaging over 3 runs.

Additionally, we let the learner use the baseline approach from Suryadi and Gmytrasiewicz (1999) (Equation (2)) to learn w_1 and w_2 from repeated observations of the player's action choices in predator-prey, starting from default weights of 0, since this method does not allow for knowledge transfer. The cumulative average of the resulting number of errors is also averaged over 3 runs and plotted in Figure 6 (titled "without transfer"). A comparison of the two plots in Figure 4 shows that

- the initial error is lower with our approach
- the convergence value of the error is lower with our approach.

Thus we have met both criteria of successful transfer with the proposed technique.

[2] Since the *p*-values are limited in the range [-1,1], we use the default initial value of 0.

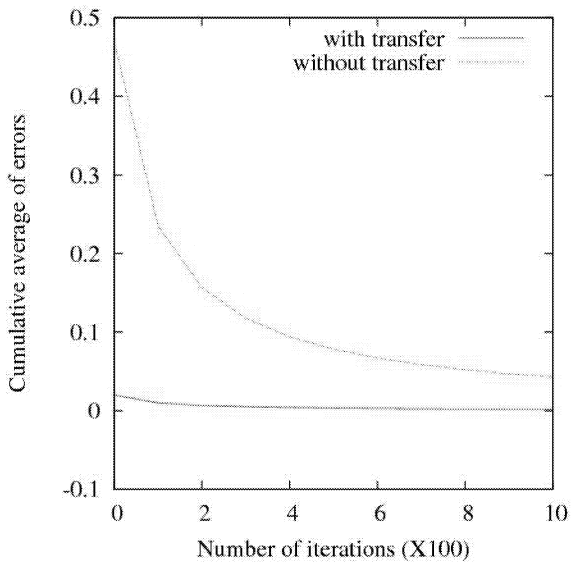


Figure 6: Difference between Agent Modeling with knowledge transfer and without transfer.

CONCLUSIONS

We have presented a reusable player modeling scheme that exploits knowledge (partial model) acquired in previous tasks to bootstrap learning in subsequent tasks. While tasks can be markedly different, the similarity that knowledge transfer exploits is in the fact that the model is of the *same* player that the agent repeatedly meets in a series of tasks. We have shown experiments in two simple tasks that demonstrate the advantage of our approach compared to previous work.

We treat the results in this paper as a proof of concept, and plan a more elaborate study of our approach in more complex RTS games. Additionally, we plan to incorporate actual human decisions instead of simulated decisions.

REFERENCES

Abdi, H., Valentin, D., & Edelman, B. (1999). Neural networks. Thousand Oaks (CA): Sage

Banerjee B., and Stone P. 2007. General Game Learning using Knowledge Transfer. *Proceedings of the Twentieth International Joint Conference on Artificial Intelligence (IJCAI-07)*, Hyderabad, India, January 6-12.

Carmel D. and Markovitch S. 1996. Learning models of intelligent agents. *AAAI/IAAI*, 1:62–67.

Charniak E. 1991. Bayesian networks without tears: making Bayesian networks more accessible to the probabilistically unsophisticated. *AI Mag. Volume 12 issue 4, p50-63. AAAI*, Menlo Park, CA, USA

Howard R. A. and Matheson J. E. 1984. Influence diagrams (article dated 1981). Howard, R.A. and Matheson, J.E. (Eds.), Readings on the principles and applications of decision analysis, 2:719–762.

Koller, Daphne & Milch, Brian, 2003. "Multi-agent influence diagrams for representing and solving games," Games and Economic Behavior, Elsevier, vol. 45(1), pages 181-221, October.

Konidaris G. and Barto A. 2007. Building Portable Options: Skill Transfer in Reinforcement Learning. *Proceedings of the Twentieth International Joint Conference on Artificial Intelligence (IJCAI-07)*, Hyderabad, India, January 6-12.

Roy D. M. and Kaelbling L. P. 2007. Efficient Bayesian Task-Level Transfer Learning. *Proceedings of the Twentieth International Joint Conference on Artificial Intelligence*, Hyderabad, India.

Russell S. J. and Norvig P. 1995. *Artificial Intelligence: A Modern Approach*. Prentice-Hall, Englewood Cliffs, NJ.

Pearl J. 1988. Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference. Morgan Kauffman, San Mateo, CA.

Suryadi D. and Gmytrasiewiez P. J. 1999. Learning models of other agents using influence diagrams. *UM '99: Proceedings of the seventh international conference on User modeling*. Pages: 223-232.

Sutton, R. S., Precup, D., & Singh, S. 1999. *Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning*. Artificial Intelligence, 112, 181--211

Widrow, B., and Hoff Jr., M. E. 1960. Adaptive switching circuits. IRE WESCON Convention Record 96–104

COMPARING OPTIMIZATION METHODS FOR WARGAME AI STRATEGIES

John Rushing, Steve Tanner
Information Technology & Systems Center
University of Alabama in Huntsville
Huntsville, AL, USA
E-mail: jrushing@itsc.uah.edu

John Tiller
John Tiller Software
Madison, AL, USA

KEYWORDS
Optimization, Artificial Intelligence, Strategic Game, Genetic Algorithms

ABSTRACT
This paper describes the use of Genetic Algorithms for developing new and interesting tactics for use by an AI planning system. It also describes how GAs are used to find flaws and loopholes in the game that players my use to game the system. The game used as an example here is Squad Battles, a commercial game developed by John Tiller software with AI developed in part by John Rushing at UAH.

INTRODUCTION
This paper describes experiments with wargame Artificial Intelligence (AI) optimization using genetic algorithms (GA) (Goldberg 1989). The experiments involve automated derivation of high level plans for the *Squad Battles* game. *Squad Battles* is a turn based squad level ground combat game series played on a hex-based map representing terrain, roads, rivers, objectives and other spatial information. The series was developed by John Tiller Software and published by HPS Simulations. It currently includes eight commercial titles, with more in the works. Titles span the period from World War II to the present day.

There are several motivations for this work. Currently in these games, all high level AI orders are manually scripted by the scenario designers. This greatly increases the amount of time that it takes to design and test a scenario. The team has been experimenting with how the GA can both speed up this process, and improve the result. Since commercial success is a primary criterion for the team's work, the research is keenly focused on measurable improvement. In addition, the GA can be used to gain insight into which sorts of plans are optimal for a particular set of circumstances. If the game engine and scenarios are realistic enough, this could even provide some indication about the relative merits of different strategies in real life situations. Often human players will identify exploits that take advantage of loopholes within a game system. A GA set up to optimize only on victory conditions could potentially identify similar loopholes, in part because it would be able to try many possible strategies, even those that on the surface seem counter intuitive, do not make logical sense, or otherwise seem flawed to a human player. Once such an odd but winning strategy is identified by the GA, the loopholes could be closed.

Other researchers have used automated methods to derive AI plans. Madeira et. al (Madeira et. al 2004) used reinforcement learning to derive army level AI plans for Napoleonic wargames. Revello and McCartney (Revello and McCartney 2002) used genetic algorithms to derive strategies for a very simple naval combat game. Miles et. al (Miles et. al. 2004) used genetic algorithms combined with human-derived strategies in an airstrike planning wargame. Vaccaro and Guest (Vaccaro and Guest 2005) compared seven search methods for finding solutions to endgames for the board game RISK, and found evolutionary strategies (genetic algorithms) to be the best. While this list of work is far from exhaustive, it does provide evidence that the use of GAs in game environments has been shown to be of interest. The team's use here is focused on developing new approaches to defining AI plans, and as a means to test the game for potential loopholes.

SQUAD BATTLES ARTIFICIAL INTELLIGENCE
Like many games, during the development of a *Squad Battles* title, the scenario designers have a significant degree of control over the behavior of units. This is accomplished through the use of scripting. Scripting is used primarily to control the attacking side in a game and to specify the overall plan of attack. It can also be used effectively for the defending side as well to do such things as direct reinforcements and undertake counterattacks. In both cases, scripting is used to control where and when units should try to move, and the objectives they should try to achieve. However, the actual behavior of the units in question depends to a large extent on the current circumstances which can be quite complicated. The scripted orders constitute a plan, and it is the job of the AI to execute that plan as well as possible under these current circumstances (Rushing et. Al. 2005).

Each unit can be assigned one or more sequential objectives, with a turn associated with each objective that defines when it should be reached by the unit. This feature can be used to create coordinated attacks involving many units over a series of turns. Orders can be given at any level, from individual squads all the way up to the largest organization in the scenario. If multiple orders exist, the most specific order takes precedence. In addition to movement orders, it is possible to script indirect fire orders. This is useful to soften up defenses along a known axis of advance or to knock out a key target.

Since orders can be given at multiple levels, may span several turns, and deal with changing situations, static scripts can sometimes be too brittle to provide meaningful and entertaining game play. Because of this, there are several cases where the *Squad Battles* AI will modify or improvise on a script. Based on an analysis of the current situation the AI may alter the scripts, either moving up the timetable or moving it back. The AI may also add orders for defending units that do not have assignments and are not being threatened by enemy units.

In some cases, it is possible to move up the time table for an attack. If the way is clear for the units to advance to their objectives, the script time table will be moved up to the current turn so that the units will advance as soon as possible. The AI checks for enemy forces along the axis of advance and for enemy units that have a line of sight and could fire on the advancing units.

In other cases it is desirable to hold back an advance for one or more turns. This is particularly true in cases where vehicles are supporting an infantry advance. If the vehicles advance with no infantry support, they will be very vulnerable to man portable anti-tank weapons. Since the vehicles inherently move much faster than dismounted infantry, it is quite possible for them to outstrip their support. The AI will attempt to detect such cases, and hold back the vehicles to allow the infantry to catch up.

Another example of modification of predefined scripts is when defending units are not holding an objective that is threatened, and they are not near any enemy units. Usually, in such cases they will move to a position where they can help the other defenders. However, units with radios, leaders stacked with radios, and units with indirect fire weapons like mortars etc. will not respond in this way. The response is delayed so the defenders do not over-react. It is also determined by an element of randomness, and where the units are more likely to move as the scenario progresses. The units will generally attempt to reach friendly held objectives or fortifications if possible, and will try to go to a hex that has a good line of sight on a spotted enemy unit.

GENETIC ALGORITHMS FOR OPTIMIZATION OF AI ORDERS

Genetic algorithms are general purpose search algorithms that are used to solve difficult optimization problems. They work by generating potential solutions within a search space specific to the problem of interest. Typically, solutions are described by bit strings in which each bit represents one facet or component of the solution (e.g. the current objective for one unit in a game). The genetic algorithm randomly generates a population of bit strings and evaluates their efficacy using some objective function. It then generates a new population of bit strings by recombining the segments of bits from its existing population, with the higher rated strings contributing more segments to the new population.

In the case of *Squad Battles*, Genetic algorithms are used to search for good AI orders for the scenario of interest. The bit strings produced by the GA are translated to AI orders for the units of one side. The game is then run 50 times using an offline driver program for *Squad Battles*, which runs the game in an AI vs. AI mode - with no user interface activated, which speeds up computation considerably. The average score over the runs is used as the objective function value for the GA. A large number of runs is required to get a stable estimate of the quality of the plan, due to the element of randomness in the game.

Three different approaches are used to translate bitstrings into AI orders. The approaches differ in the range of hexes available for the AI orders. In the first approach, any hex on the map may be a destination for an AI order. In the second approach, only one of every four hexes is allowed. In the final approach only a manually selected subset of the hexes are allowed. In all cases, the GA is run with the following parameters:

Crossover Percentage:	75%
Mutation Percentage:	25%
Bit Mutation Rate:	2%
Population:	500

EXPERIMENTAL RESULTS

The genetic algorithm was run on the Hue-1 scenario from *Squad Battles – Vietnam*. In this scenario about the Tet Offensive and the Battle of Hue, US forces attempting to enter the city of Hue must negotiate a route that includes multiple Viet Cong ambushes. The US is the attacker in this case, and receives points both for destroying enemy units and for exiting their forces

from the map on the opposite side from their entrance. The results of the GA runs are summarized in Table 1, which has the top five scores for each approach. For reference, the average score for the default AI orders included in the scenario (a straightforward advance down the main road) is 44.97, meaning the GA was able to produce substantially better results.

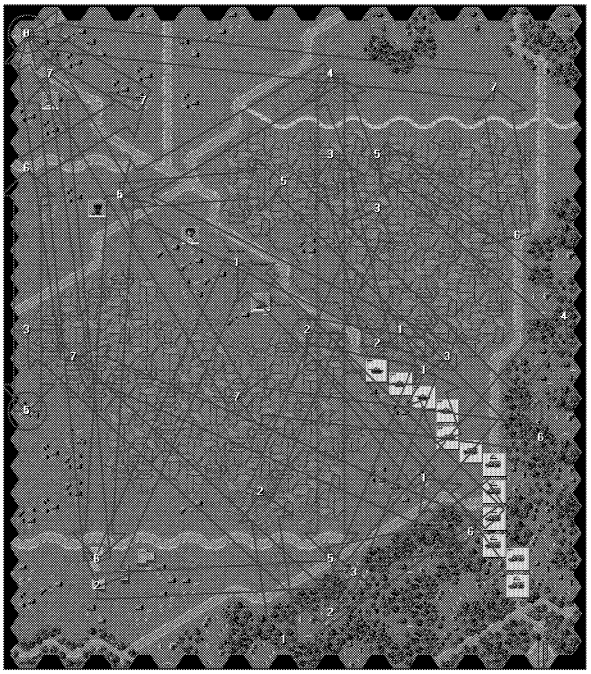
All Hexes	Restricted Hexes	Selected Hexes
189.16	158.46	186.02
188.52	158.44	183.74
188.14	158.00	183.36
187.96	157.10	182.90
187.82	157.00	182.20

Table 1:Top five average scores achieved using each encoding scheme

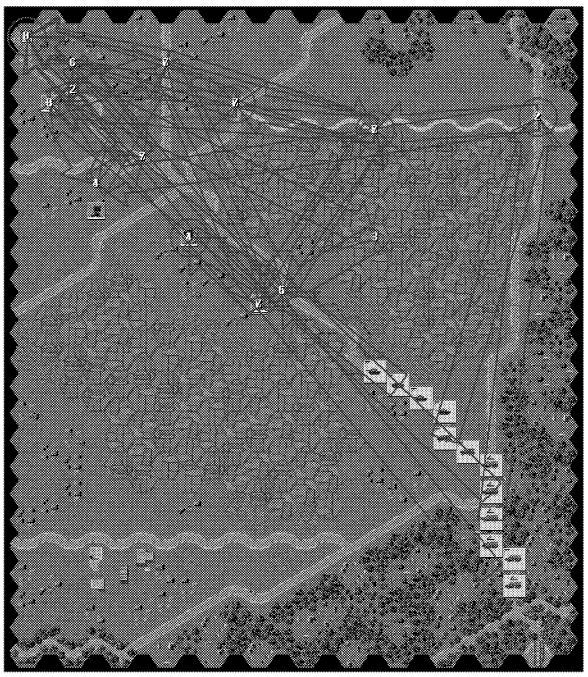
This indicates that placing arbitrary restrictions on the AI orders (restricted hexes column) results in significantly worse scores. Limiting the AI orders to key selected hexes results in slightly lower scores than allowing the GA to optimize over all hexes. While the GA was always able to achieve good numeric scores in terms of victory conditions, the resulting AI orders in some cases were not particularly coherent. This is illustrated in Figure 1, which shows the top scoring plans for the unrestricted search and the search using selected hexes. It is difficult to tell why the seeming chaos of first plan works (a), while the second seems quite sensible (b). In the figure, the blue arrows represent orders given to units over a series of several turns. The units are attempting to travel from the lower right to obtain the objective in the upper left. The one thing the two plans have in common is that they spread the attacking forces off of the main road, preventing bottlenecks and ambushes. In general, the second plan is preferable because it is easier to describe a commander's intent (e.g. "advance along main road with flanking action"). However, the first plan is also very interesting from a game designer's point of view because it illustrates that a rather chaotic plan can actually be very effective within the game

CONCLUSIONS

Genetic algorithms were used to derive sets of AI orders for *Squad Battles*, a squad level turn based ground combat game. The AI orders serve as a top level plan of attack for the scenario, while the dynamic AI in the game engine interprets and executes the orders. Three different coding schemes were used to select hexes for the AI orders. Regardless of coding scheme, the GA was able to find a plan of attack that achieved higher average victory points than the default plan included in the scenario. However, the plans generated using unrestricted search were difficult to understand or explain. The plans generated using a pre-selected group of hexes produced good results and relatively sensible plans. These experiments



(a)



(b)

Figure 1: Best result achieved using (a) all hexes and (b) selected hexes.

demonstrate that it is in fact possible to generate good high level plans using automated search methods with little human intervention.

The team is currently making additional runs, and analyzing the results. It appears that the use of GAs may provide some insight into improvements to the AI

itself. These may be incorporated into the commercial versions of the game to provide better playability and smarter adversaries. The team is also looking into some of the more interested plans generated to see if they identify loopholes that would allow players to game the system. In the example presented here, it may be that the seemly chaotic plan may thwart the defending AI through its shear unpredictably. In real battles, it is unlikely that such an approach would be used. The question then becomes one of how much realism should be represented in the game.

ACKNOWLEDGEMENTS

Funding from the Air Force Office of Scientific Research, Dr. Robert Barker, Program Manager, for project "Optimal Tactics for Asymmetric Warfare" is gratefully acknowledged

REFERENCES

Goldberg, D. 1989. Genetic Algorithms in Search, Optimization and Machine Learning. Addison Wesley. New York.

Madeira, C., Corruble, V., Ramalho, G. and Ratitch, B. 2004. Bootstrapping the Learning Process for the Semi-Automated Design of a Challenging Game AI, *AAAI Workshop on Challenges in Game Artificial Intelligence*, pp. 72-76.

Miles, C., Louis, S., Cole. N. and McDonnell, J. 2004. Learning to Play Like a Human: Case Injected Genetic Algorithms for Strategic Computer Gaming. *Proceedings of the 2004 Congress on Evolutionary Computation*, vol. 2, pp. 1441-1448, June, 2004

Revello, T. and McCartney, R. 2002. Generating War Game Strategies Using a Genetic Algorithm. *Proceedings of the 2002 Congress on Evolutionary Computation*, vol. 2, pp. 1086-1091, May, 2002.

Rushing, J., Tanner, S., Farnell, B., Morgan, G., Tiller, J., *The Observe-Orient-Decide-Act (ODDA) Model for Strategic Game AI*, CGAIMS, Louisville, KY, Jul 27-30, 2005.

Vaccaro, J. and Guest, C. 2005. Planning an Endgame Move Set for the Game RISK: A Comparison of Search Algorithms. *Proc. IEEE Transactions on Evolutionary Computation*, vol. 9, no. 6. pp. 641-652, December 2005.

BIOGRAPHY

John A. Rushing received the B.S. degree in computer and systems engineering from Rensselaer Polytechnic Institute, Troy, NY, in 1988 and the M.S. and Ph.D. degrees in computer science from the University of Alabama, Huntsville, in 1995 and 1999, respectively. He is currently a Senior Research Scientist with the Information Technology and Systems Center, University of Alabama in Huntsville. He is the lead architect and designer of the ADaM data mining toolkit. His research interests include artificial intelligence, pattern recognition, image processing, and data mining.

SIMULATION IN GAME DESIGN

INTERMIPMAPS: AN EXTENDED APPROACH TO GEOMIPMAPPING

Alan Horne, Xin Li
DigiPen Institute of Technology
5001 150th Ave NE
Redmond, WA 98052 USA
ahorne2@digipen.edu, xli@digipen.edu

ABSTRACT

Terrain level-of-detail algorithms have undergone vast changes throughout the past ten years as consumer 3D hardware has become more powerful and flexible; newer algorithms delegate more processing load onto the GPU, enhancing performance at the expense of hardware dependence and algorithmic complexity.

The Geomipmapping technique (de Boer 00) was one of the earlier algorithms designed with 3D hardware in mind; instead of spending resources attempting to find a "perfect set" of triangles, it operates on batches of triangles to assemble a terrain that lies within a specified error bound. Despite the fact that there are many algorithms superior to the Geomipmapping technique in terms of run-time efficiency, the technique is still in widespread use due to its conceptual simplicity and ease of implementation.

The Intermipmapping algorithm presented in this paper aims to maintain the concepts and simplicity of the Geomipmapping technique while eliminating its shortcomings; to achieve this, we extend the geomipmap concept to introduce Intermipmaps, and utilize a bintree-based spatial partitioning scheme to eliminate terrain cracking.

INTRODUCTION

Geomipmapping

The Geomipmapping algorithm is designed to operate on a square height-mapped terrain mesh of vertex side length 2^n+1. A height-mapped terrain mesh is a regular X-Z grid of vertices with varying Y-values; the X-Z axes correspond to the X-Y pixel coordinates of a grayscale image (the height map), and the Y axis corresponds to the grayscale value of the pixel at those particular X-Z coordinates.

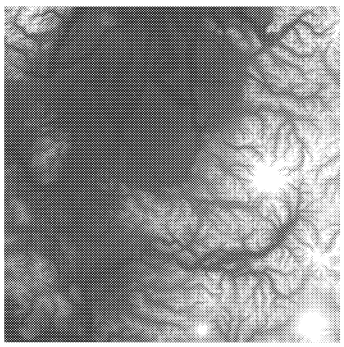


Fig. 1: Example Height Map

The basic Geomipmap concept draws from a texture mapping technique called mipmapping, a process in which a texture is successively filtered into lower detail levels. When a bitmapped texture is applied to a distant object, a lower-detail mipmapping can be displayed in order to maintain visual quality.

Similarly, in the Geomipmapping technique terrain vertex grids are filtered into lower-detail geomipmaps by repeatedly taking every other vertex from the next-highest detail geomipmap (the first geomipmap uses the base mesh to filter from).

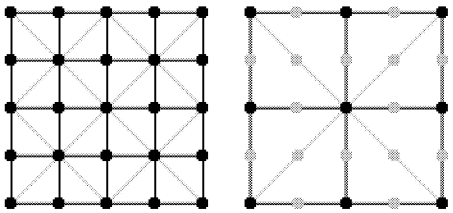


Fig. 2: Geomipmap Vertices – Higher Detail (left) and Lower Detail (right)

The Geomipmapping algorithm begins by partitioning the base height map into a regular grid wherein each grid element is itself a 2^n+1 square mesh; Geomipmap levels are then generated for each grid square. At runtime, the appropriate geomipmap level to display for each visible grid square is selected based upon a screen-space error metric.

Choosing different geomipmap levels for two neighboring tiles causes what is known as "cracking" – places where triangle edges don't match up, leaving cracks in the terrain through which the background color can be seen. This type of artifact is in fact a common problem with most terrain algorithms. While it is not hard to cover up terrain cracks, techniques to do so come at the expense of either visual quality or runtime efficiency.

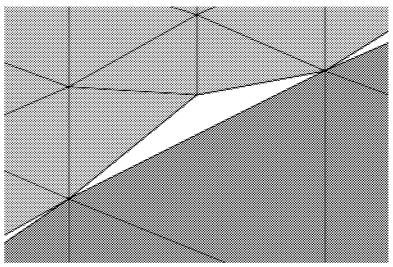


Fig. 3: Terrain Cracking

Intermipmaps

Intermipmaps are best described as an extension of the geomipmap concept; while both are increasingly coarse grid structures, intermipmaps operate on the vertices addressable by following grid lines oriented at 45 degrees from the geomipmap

grid lines.

These intermipmaps have several useful properties; the most obvious is that they introduce additional detail levels for the algorithm to choose from at run-time, potentially allowing a given error tolerance to be met with fewer triangles.

Another useful property is that the intermipmaps are edge-matched to neighboring geomipmaps within one detail level - horizontal / vertical edges of an intermipmap match the next-lowest detail geomipmap, and diagonal edges match the next-highest. This means that geomipmaps and intermipmaps are able to tile together without introducing cracks – with an appropriate selection scheme, geomipmaps and intermipmaps together can seamlessly cover an entire terrain in continuously varying detail with no cracking artifacts.

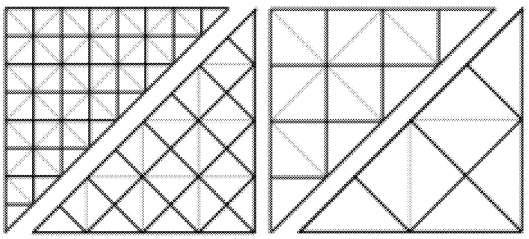


Fig. 4: Geomipmap-Intermipmap Edge-Matching

A method of generating intermipmaps may not be immediately clear, but in application it is relatively easy to implement a generalized method of generating both geomipmaps and intermipmaps. This method will be discussed in a later section.

DATA STRUCTURES

Bintree

Binary triangle trees (bintrees) are binary tree structures that spatially partition a two-dimensional right-triangular area; at each successive tree level a bintree node is partitioned into two smaller child bintree nodes. In addition to its left and right children, each node also maintains a record of its left, right, and base neighbor nodes.

The intermipmap terrain is represented as a pair of bintrees as in ROAM and other algorithms (Duchaineau et al. 97, Cignoni et al. 03); each bintree node represents a triangular chunk of terrain at a geomipmap / intermipmap level determined by the level of the node in the tree hierarchy.

The bintree spatial partitioning scheme is used because when ROAM-like node split rules (Duchaineau et al. 97) are enforced then any two neighboring nodes differ at most by one hierarchical level. When geomipmaps and intermipmaps are placed appropriately in the nodes according to hierarchical level, the nodes are edge-matched and no cracking occurs.

Bintree Nodes

Each bintree node represents a right-triangular patch on the terrain. The detail level of a node is determined by its level in the bintree hierarchy; leaf nodes are level **0**, and the two tree root nodes are level **h – 1**, where **h** is the height of the tree calculated as follows:

$$h = 2\log_2\left(\frac{width_{HM} - 1}{width_{node} - 1}\right) + 1$$

h	bintree height
$width_{HM}$	side length in vertices of the height map
$width_{node}$	side length in vertices of a leaf (specified)

Geomipmaps and intermipmaps are numbered as well to correspond to the tree hierarchy levels. Level 0 corresponds to the highest detail available (the base mesh); intermipmaps and geomipmaps then alternate in successively lower detail levels until the top of the tree is reached. As a result, geomipmaps have even level numbers and intermipmaps have odd level numbers.

Node Generation

An interesting property appears as nodes of differing levels are compared. As the detail level increases, the number of triangles in a given terrain area doubles; the terrain area covered by the node is cut in half, however, so the net effect is that the number of triangles in a node is constant, regardless of detail level. This means that all of the nodes are geometrically similar (that is, they are identical in all aspects aside from scale and orientation); this property is exploited to create generalized techniques for both generating and rendering the terrain nodes.

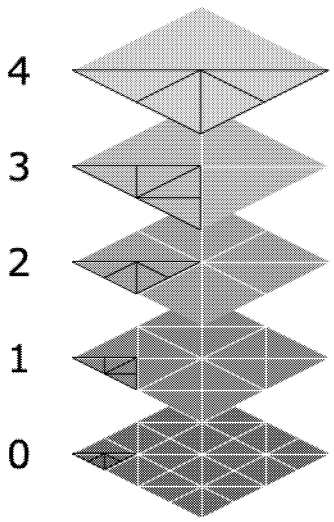


Fig. 5: Node Similarity

Each bintree node has one of eight orientations; each orientation corresponds to one of the eight directions in which a bintree node's apex can point. It is easy to determine a node's orientation when the tree is created – the roots are set to arbitrary orientations of 6 and 2, and each successive child node's orientation is determined by two rules:

```
leftChildOrientation  = (nodeOrientation + 3) % 8
rightChildOrientation = (nodeOrientation + 5) % 8
```

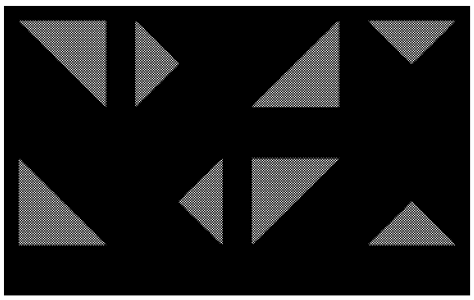


Fig. 6: Node Orientations

These orientations are necessary for the generalization of node creation; each node can be created using a single function that takes orientation, level, and terrain region as parameters. Two nested **for()** loops are used along with an orientation mapping function to generate the node vertex buffers:

```
int iModulus = pow(2, pNode->level / 2)
for(int r = 0; r < NODE_SIZE_THRESHOLD; r++)
{
    for(int c = r; c < NODE_SIZE_THRESHOLD; c++)
    {
            int iR, iC;
            pNode->MapRowCol(iR, iC,
                    r * iModulus, c * iModulus);
            pVB[VBIndex++] = GetVertex(iR, iC);
    }
}
```

Here, `iModulus` is the factor used to "skip" vertices for the creation of geomipmaps and intermipmaps. The orientation mapping function `MapRowCol()` is:

```
void BintreeNode::MapRowCol(int &iRowOut, int &iColOut,
int iRowIn, int iColIn)
{
    switch(orientation){
    case 0:
            iRowOut = iRowIn;
            iColOut = iColIn;
            break;
    case 1:
            iRowOut = iRowIn + iColIn;
            iColOut = iColIn - iRowIn;
            break;
    case 2:
            iRowOut = iColIn;
            iColOut = m_iWidth - iRowIn - 1;
            break;
    case 3:
            iRowOut = iColIn - iRowIn;
            iColOut = m_iWidth - (iRowIn + iColIn) - 1;
            break;
    case 4:
            iRowOut = m_iWidth - iRowIn - 1;
            iColOut = m_iWidth - iColIn - 1;
            break;
    case 5:
            iRowOut = m_iWidth - (iRowIn + iColIn) - 1;
            iColOut = m_iWidth - (iColIn - iRowIn) - 1;
            break;
    case 6:
            iRowOut = m_iWidth - iColIn - 1;
            iColOut = iRowIn;
            break;
    case 7:
            iRowOut = m_iWidth - (iColIn - iRowIn) - 1;
            iColOut = iRowIn + iColIn;
            break;
    default:
            break;
    }
}
```

This function takes a (row, column) pair in 0-orientation space and maps it into another orientation's space. The coordinates must then be mapped into height map space, but this involves only adding the node's row and column offsets to the node-space coordinates.

Once the node's vertices are stored in its vertex buffer, it is possible to stitch together the vertices in whatever indexed manner is deemed appropriate (triangle lists, triangle strips, etc.).

Determining Whether to Render a Node

With the Intermipmapping technique, each bintree node only has one level of detail. If a node's detail level is deemed insufficient for display, then its higher-detail children are rendered instead.

The method of determining whether a node's detail level is sufficient depends on the needs of the application; for the purposes of our implementation we have adapted a screen-space error metric similar to that used by the Geomipmapping technique (de Boer 00). With this metric, the minimum distance at which a node's error is tolerable (that is, below a threshold in pixels) is stored with the node and compared with the view distance at runtime.

At load time, the maximum z-error $\boldsymbol{\delta}$ for each bintree node is calculated - this $\boldsymbol{\delta}$ is the maximum of the differences between the highest-resolution grid vertices and their corresponding interpolated vertices in the node. $\boldsymbol{\delta}$ is then used along with the camera's field-of-view and the vertical screen resolution to find the minimum distance at which $\boldsymbol{\delta}$ projects into $\boldsymbol{\tau}$ pixels:

$$D_n = \frac{v_{res}|\delta|}{2\tau \tan(\frac{fov}{2})}$$

$\mathbf{v_{res}}$	vertical screen resolution in pixels
$\boldsymbol{\tau}$	specified error tolerance in pixels
fov	field-of-view angle

In practice, $\mathbf{D_n^2}$ is stored instead of $\mathbf{D_n}$; this eliminates a square-root operation per node at run-time when comparing the distance from the camera to the node.

ALGORITHM

Initialization

During the initialization process, the two bintrees are traversed and each node is allocated and initialized.

Each node is initialized with the coordinates of the portion of the original height map that it corresponds to, and the area is analyzed to find $\boldsymbol{\delta}$ (and thusly the value of $\mathbf{D_n^2}$). This process consists of iterating through each vertex within the original height map that falls in the node's triangular area and comparing the height value of the original vertex with the interpolated height value at that same point in the simplified node mesh. Figure 7 is a simplified two-dimensional example of this interpolation; in three dimensions it is necessary to interpolate

three times in order to measure the value of **δ**.

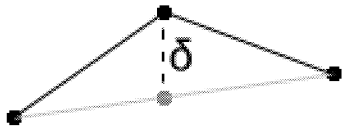


Fig. 7: Interpolating to Find **δ**

The coordinate mapping function used for vertex buffer creation can also be used for this, with one caveat – the highest-detail intermipmap addressing does not map to all of the original vertices of the height map, so a small hack is required to address the left-out vertices. Basically - for each vertex iterated through that isn't in the last column, a value of ½ is added to both the row and column before the coordinate is mapped into height map space.

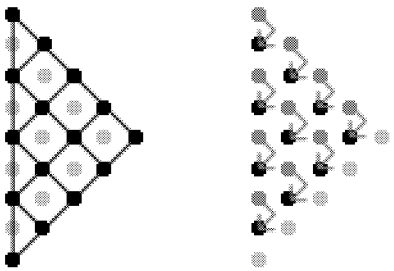


Fig. 8: Addressing Left-Out Vertices in Intermipmaps

Run-time

First, the topmost two nodes are marked as active. Then the update loop runs, recursively checking to see if each node's error is tolerable. If it is not, the node is split / marked inactive (potentially incurring more splits) and the node's neighbors are informed; if it is tolerable, the node is left active and the tree is not descended any further.

Pseudocode:

```
Mark root nodes as active
Process bintree. For each node…
    • If the node is a leaf, add it to the draw list and
continue
    • If the node is already split, proceed immediately
to its children
    • If the squared distance from the camera to the
node is smaller than the node's Dn², split the node
    • Otherwise, add it to the draw list
Draw all nodes in the draw list from front to back
```

IMPROVEMENTS

Frustum culling

The bintree spatial partitioning scheme can be used to quickly cull large portions of the terrain from consideration for rendering, greatly reducing the amount of triangles that are sent to the graphics hardware At load time, a bounding volume is constructed for each bintree node from the limits of the vertices within it; at run-time, if any node's volume is entirely out of the view frustum, it can be ignored. This precludes it and its descendants from distance checks and rendering, dramatically improving frame rate.

Precalculating D_n^2

The value D_n^2 used to determine the appropriate detail level is not view-dependent and so can be pre-calculated at load time, or even calculated in a separate offline process and stored with the height data. At run-time, this saves several mathematical operations – the only computation required per node is a squared distance from the viewpoint to the node center.

Single index buffer

The methods of constructing and rendering bintree nodes are identical for all nodes regardless of detail level and orientation, so it is possible to use a single index buffer to render all the visible nodes. This reduces the amount of data that must be sent to the graphics hardware each frame, increasing performance.

On-Demand Paging

It is prohibitively expensive to store in memory the vertex buffers for all the nodes of a large terrain. The recommended approach is to implement a caching system that will load needed terrain nodes on demand and retain them in memory until it is determined that they are not needed anymore. This ensures that there will not be unnecessary disk access in instances where the camera may be weaving along a detail threshold.

Geomorphing

Another potential improvement is the addition of geomorphing, a process in which terrain patch vertices are smoothly interpolated when moving from one detail level to another. This prevents jarring detail level "popping" at the cost of additional computation. However, in practice an error metric of sufficient accuracy will prevent most popping artifacts without the CPU or GPU cost normally incurred by geomorphing.

RESULTS

Fewer draw calls

Geomipmapping operates on a fixed regular spatial partitioning scheme. This means that as the amount of viewable terrain increases, the amount of nodes that need to be drawn increases dramatically. Intermipmapping is different in that it replaces many smaller high-detail chunks with fewer larger low-detail chunks as the detail level in an area decreases. The result of this is a substantial decrease in the number of draw calls per frame, which translates into less CPU overhead; in addition, the number of draw calls does not vary as widely as with geomipmapping.

Fewer extraneous triangles

Geomipmapping requires some method of crack-filling – without it, it is possible to see gaps between tiles of differing detail. Methods of hiding cracks that add triangles simply for the sake of covering up cracks introduce triangles that do not otherwise contribute to the visual quality of the terrain, using up triangle budget to cover up algorithmic deficiencies. Intermipmapping is designed from the ground up so that crack-filling is automatic, so there are no visual defects at node borders; in addition, there are no triangles in the budget that aren't actively contributing to the terrain rendering.

Better detail level resolution

The addition of intermipmaps helps the granularity of the detail levels – the intermipmaps fall neatly between the geomipmaps in terms of the detail level. This cuts down somewhat on the unnecessary detail introduced with intermipmaps – an area that might be given a detail level of 0 (the highest level) with geomipmapping might only need to have a level of 1 (the first intermipmap) with intermipmapping.

Smaller footprint

Geomipmapping requires the storage of several levels of detail for each patch of a regularly-partitioned grid, a storage requirement that can be lessened with several creative buffer reuse methods; however, the storage requirement is ultimately higher than what can be achieved with intermipmapping. Because each node is similar (in the geometric sense) and the vertex layouts within each node are identical, a single index buffer can be used to render all visible nodes.

Unnecessary Detail

In geomipmapping it is possible for two adjacent nodes to have widely disparate detail levels, something that might be desired if a high-frequency detail area lies close to a low-frequency detail area. This is not possible with intermipmapping, however; the bintree splitting rules ensure that any two nodes that share an edge differ by a hierarchy level of only one. This means that a high-frequency detail area will force surrounding areas to use higher detail levels, causing more triangles to be drawn than necessary to meet the specified error bound.

Algorithmic complexity

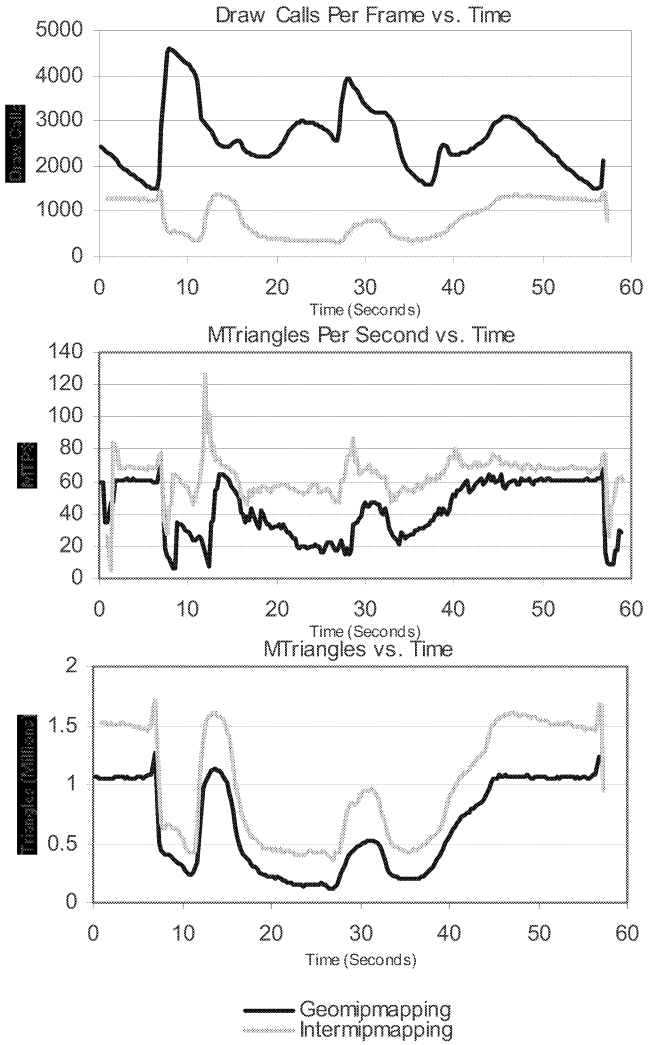
Though intermipmapping may be more efficient than geomipmapping in several respects, ease of implementation is not one of them. Geomipmapping was designed to be simple to understand and easy to implement; while the concept of intermipmapping is reasonably simple to understand, there are some implementation pitfalls that may bring the level of complexity higher.

Quantitative Results

The graphs to the right show the result of a 55-second flythrough loop through the 4k x 4k Puget Sound height dataset with a specified error tolerance of 1.5 pixels on a 2GHz Pentium-4M.

The largest differences are in the draw calls per frame and overall triangle throughput – in these areas, the intermipmapping technique was able to significantly outperform geomipmapping.

Visible in the third graph is the unnecessary detail of the intermipmapping algorithm; while both algorithms meet the specified 1.5 pixel error bound, the geomipmapping technique is able to do so with fewer triangles. Whether this is important is debatable; given the improvements in triangle throughput and draw call overhead, the additional triangles may be acceptable.



REFERENCES

Cignoni, P.; Ganovelli, F.; Gobbetti, E.; Marton, F.; Ponchio, F.; Scopigno, R. 2003. *BDAM – Batched Dynamic Adaptive Meshes for High Performance Terrain Visualization.* In Computer Graphics Forum, vol.22 #3.

de Boer, W.H. 2000. "Fast Terrain Rendering Using Geometrical MipMapping." http://www.flipcode.com/articles/article_geomipmaps.pdf .

Duchaineau, M.A.; Wolinsky, M.; Sigeti, D.E.; Miller, M.C.; Aldrich, C.; Mineev-Weinstein., M.B.. 1997. *ROAMing terrain: Real-time optimally adapting meshes.* In Proceedings IEEE Visualization '97, pages 81–88. IEEE.

Lindstrom, P.; Pascucci, V. 2001. *Visualization of large terrains made easy*. In Proc. IEEE Visualization 2001, pages 363-370, 574. IEEE Press.

Ulrich, T. 2002. *Chunked LOD: Rendering Massive Terrains using Chunked Level of Detail Control.* From http://tulrich.com/geekstuff/chunklod.html

AUTHOR LISTING

AUTHOR LISTING