13TH INTERNATIONAL CONFERENCE

ON

INTELLIGENT GAMES AND SIMULATION

GAME-ON® 2012

EDITED BY

Antonio José Fernandez-Leiva

Carlos Cotta Porras

Raúl Lara Cabrera

NOVEMBER 14-16, 2012

Universidad de Málaga

Málaga

SPAIN

A Publication of EUROSIS-ETI

13TH International Conference

on

Intelligent Games and Simulation

MÁLAGA, SPAIN

NOVEMBER 14 - 16, 2012

Organised by

ETI

Sponsored by

EUROSIS

Universidad de Málaga

Co-Sponsored by

Binary Illusions

BITE

Ghent University

Higher Technological Institute

Larian Studios

LMS

University of Skövde

Hosted by

Universidad de Málaga

Málaga, Spain

EXECUTIVE EDITOR

PHILIPPE GERIL
(BELGIUM)

EDITORS

General Conference Chair

Antonio José Fernandez-Leiva
University of Malaga, Malaga, Spain

General Program Chair

Carlos Cotta Porras
University of Malaga, Malaga, Spain

Local Programme Chairs

José Enrique Gallardo Ruiz, University of Malaga, Malaga, Spain
Raúl Lara Cabrera, University of Malaga, Malaga, Spain

INTERNATIONAL PROGRAMME COMMITTEE

Game Development Methodology

Óscar Mealha, University of Aveiro, Portugal
Esteban Clua, Universidade Federal Fluminense, Brasil

Physics and Simulation

Graphics Simulation and Techniques
Ian Marshall, Coventry University, Coventry, United Kingdom

Facial, Avatar, NPC, 3D in Game Animation
Yoshihiro Okada, Kyushu University, Kasuga, Fukuoka, Japan
Marcos Rodrigues, Sheffield Hallam University, Sheffield, United Kingdom
Joao Manuel Tavares, FEUP, Porto, Portugal

Rendering Techniques
Joern Loviscach, Hochschule Bremen, Bremen, Germany

Artificial Intelligence

Artificial Intelligence and Simulation Tools for Game Design
Stephane Assadourian, UBISOFT, Montreal, Canada
Flavio Soares Correa da Silva, USP, Sao Paulo, Brazil
Patrick Dickinson, Lincoln University, Lincoln, United Kingdom
Antonio J. Fernandez, Universidad de Malaga, Malaga, Spain
Marc-Philippe Huget, University of Savoie, Le-Bourget-du-Lac, France
Joseph Kehoe, Institute of Technology Carlow, Carlow, Ireland
Tshilidzi Marwala, University of Witwatersrand, Johannesburg, South-Africa
David Moffat, Glasgow Caledonian University, Glasgow, United Kingdom
Sam Redfern, National University of Ireland, Galway, Ireland
Oryal Tanir, Bell Canada, Montreal, Canada
Christian Thurau, Universitaet Bielefeld, Bielefeld, Germany
Miguel Tsai, Ling Tung University, Taichung, Taiwan
Ian Watson, University of Auckland, Auckland, New Zealand

INTERNATIONAL PROGRAMME COMMITTEE

Learning & Adaptation

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

Intelligent/Knowledgeable Agents

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

Collaboration & Multi-agent Systems

Victor Bassilious, University of Abertay, Dundee, United Kingdom
Sophie Chabridon, Groupe des Ecoles de Telecommunications, Paris, France

Opponent Modelling

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

Peripheral

Psychology, Affective Computing and Emotional Gaming

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

Artistic input to game and character design

Anton Eliens, Vrije Universiteit Amsterdam, Amsterdam, The Netherlands

Storytelling and Natural Language Processing

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

Online Gaming and Security Issues in Online Gaming

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

INTERNATIONAL PROGRAMME COMMITTEE

MMOG's

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

Serious Gaming

Wargaming Aerospace Simulations, Board Games etc....

Roberto Beauclair, Institute for Pure and Applied Maths., Rio de Janeiro, Brazil
Jaap van den Herik, Tilburg University, Tilburg, The Netherlands

Games for training

Michael J. Katchabaw, The University of Western Ontario, London, Canada

Games Applications in Education, Government, Health, Corporate, First Responders and Science

Russell Shilling, Office of Naval Research, Arlington VA, USA

Games Interfaces - Playing outside the Box

Games Console Design

Chris Joslin, Carleton University, Ottawa, Canada

GAME ON®
2012

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

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

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

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

EUROSIS-ETI Publication

ISBN: 978-9077381-74-8
EAN: 978-9077381-74-8

Preface

Dear participants,

On behalf of all the people and institutions that made this conference possible, I wish to welcome you all here to the Computer Science School of the University of Málaga, Málaga for the 13th edition of the Annual Conference on Simulation and AI in Games. (GAMEON'2012)

In addition to the interesting and varied submitted papers, we are very grateful to our keynote speaker - Juan Julián Merelos Guervos for his talk on "MasterMind or finding a needle in a haystack algorithmically" and to Antonio Mora García for helping to organize the tutorial on "Computational Intelligence applied to videogames; past, present and future"

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

Next to the general conference presentations, we are also proud to be able to show the conference participants demos of games research at the University of Málaga and share our games research experiences first hand with the aforementioned.

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

Last but not least, I also wish to thank our sponsors for supporting the conference.

Finally, I hope you enjoy your stay here in Málaga and you have time to explore the city during and after the event.

Málaga, 14.11.2012

Antonio José Fernandez-Leiva
GAMEON'2012 General Conference Chair

CONTENTS

Preface ..... **IX**
Scientific Programme **1**
Author Listing **89**

GAMES IN SOCIETY

A New Cognitive Classification of Video Games
Alex F. V. Machado, Esteban W. G. Clua, Ismael A. Batista, Marlon C. Santiago, Rafael R. Padovani, Bruno G. Soares and Sandro P. Carvalho **5**

Serious Games in a Social Context
Anton Eliëns **8**

GAME ANALYSIS

A Framework for Quantitative Analysis of User-Generated Spatial Data
Tom Feltwell, Patrick Dickinson and Grzegorz Cielniak **17**

Finding fast solutions to the game of Mastermind
J.J. Merelo, A.M. Mora and C. Cotta **25**

GAME DESIGN

On the Human Visual Perception and Game Design
Adriana Alvarado and Benjamín Hernández **31**

Facilitating Open Plot Structures in Story Driven Video Games using Situation Generation
Gordon Brown and David King **34**

ADAPTIVE GAMES

Improving Software Quality through Design Patterns: A Case Study of Adaptive Games and Auto Dynamic Difficulty
Muhammad Iftekher Chowdhury and Michael Katchabaw **41**

Real-time Adaptive Track Generation in Racing Games
Jake Bird, Tom Feltwell and Grzegorz Cielniak **48**

CONTENTS

Procedural Map Generation for a RTS Game
Raúl Lara-Cabrera, Carlos Cotta and Antonio J. Fernández-Leiva **.....53**

BEHAVIOURAL AI

Self-Organizing Squad and Crowd Formation for Emergency Evacuations in Serious Games
César García-García, Victor Larios-Rosillo and Hervé Luga **.....61**

CHAMELEON: A Learning Virtual Bot For Believable Behaviors In Video Game
Fabien Tencé, Laurent Gaubert, Pierre De Loor and Cédric Buche **.....64**

Hall-of-Fame Competitive Coevolutionary Algorithms for Optimizing Opponent Strategies in a New Game
Mariela Nogueira, Juan M. Gálvez, Carlos Cotta
and Antonio J. Fernández-Leiva **.....71**

Best-First Search with Genetic Algorithm for Space Optimization in Pathfinding Problems
Ulysses O. Santos, Alex F. V. Machado and Esteban W. G. Clua **.....79**

SCIENTIFIC PROGRAMME

GAMES IN SOCIETY

A New Cognitive Classification of Video Games

Alex F. V. Machado and Esteban W. G. Clua
Instituto de Computação, Universidade Federal Fluminense, Niterói, RJ – Brasil
E-mail: alexcataguases@hotmail.com and esteban@ic.uff.br

Ismael A. Batista, Marlon C. Santiago, Rafael R. Padovani, Bruno G. Soares and Sandro P. Carvalho
Departamento de Computação, Instituto Federal de Educação, Ciência e Tecnológia do Sudeste de Minas Gerais - Rio Pomba, MG, Brasil
E-mail: {ismaelsmith| marlonsantiago18| rafael.rpadovani| bgsoares} @gmail.com and sandro.paiva@ifsudestemg.edu.br

KEYWORDS
Cognitive, classification, game, genre.

ABSTRACT

Indicative classification by age group appeared in order to warn up parents how audiovisual works could exercise training of children. Therefore, this classification has contributed to the electronic games industry, while reduced the stigma of marginalization that many authors have attributed to these applications. Likewise, this work presents a cognitive classification which would consolidate and enhance the benefits of gaming in society. We propose a classification for electronic games based on the cognitive processes described in the literature, trying to integrate the features of each game to the benefits provided by the practice.

INTRODUCTION

The video game industry is one of the largest in the world and has entered into a process of expansion, exceeding the billion dollar movie industry in terms of revenue, according to the Electronic Magazine [Landim 2011]. Most people who play electronic games do simply for pleasure or as a way to pass the time. However they do not know the potential of such an act, for example, health benefits [Barthelemy 2009], the individual training [Grace 2005] and development of cognitive functions [Griffiths 2002].

According to [Romão et al. 2006], indicating the classification criteria are established based on research and debate, based on the Country's Federal Constitution and the Statute of Children and Adolescents. This article presents a proposal for classification of the main cognitive models based on the analysis of the genres of games.

Currently, cognitive processes can be classified into categories, according to Norman in 1993 (reviewed in [Preece et al. 2005; Passerino 1998]). Thus, we intend to investigate whether and what links can be established between the processes cognitive and electronic games, to create a classification system.

This game classification can influence in a constructive way:

- In the choice of parents in a particular game for the child, enabling them to analyze whether it contributes to the development of certain cognitive area;

- In the gaming industry, which can be based on the cognitive benefits to improve the quality of its products;

- In psycho-pedagogical areas: selection of games to develop specific skills in the treatment of disorders of learning or psychomotor problems. Thus, this proposal has enormous potential to the gaming market, with applications ranging from educational to therapeutic.

In [Hotz et al. 2012] researchers of the University of Rochester showed that players can focus on six tasks at once without losing focus, while people who do not have the habit of playing, can be focused on only four. In another study, also reviewed by [Hotz et al. 2012], researchers at the University of Michigan, which lasted three years and performed with 491 high school students, found that students who engage more computer games got the best performance on a standardized test of creativity, regardless of gender, race or the style of game played.

COGNITIVE PROCESSES

In this paper we assume that video games influence the cognitive process of the user. This session will discuss the limits and influence of each of these processes.

According to Norman in 1993 (reviewed in [Preece et al. 2005]) cognition can be understood as what happens in our mind when we perform our daily activities, such as thinking, remembering to learn, dress, make decisions, read, watch, talk and write. Preece makes a division of cognitive processes in attention, perception, memory, learning, communication, and cognitive reflective. Part of this study is consolidated by [Matias and Greco 2010], and served as the foundation of our proposed classification of games.

[Passerino 1998] conducted a detailed study relating cognitive processes memory (visual, auditory, kinesthetic), temporal and spatial orientation (in two and three dimensions), manual coordination (wide and thin), perception, auditory, visual perception (size, color, detail, shape, position, handedness, complementation), logical-mathematical, linguistic expression (oral and written), planning and organization as the main processes. This study confirmed many of the concepts defined by Norman in 1993 (reviewed in [Preece et al. 2005]).

RELATIONSHIP BETWEEN ELECTRONIC GAMES GENRE AND COGNITIVE PROCESSES

According to [Foster and Mishra, 2009], organize games by genres is not a new idea. However, the goal is not merely classify different genres of games, but tries to connect these genres for the games in order to develop a systematic approach to the study and evaluation of these types of learning that can occur through the practice of different genres.

The definitions of each gender, which served as the basis for the study of cognitive classification, were made by: [Grace 2005], which defined the genre of action, adventure, puzzle, strategy, RPG and simulation; [Moyles and Janet 2002], which studied the gender education; [Pase and Tietzmann 2009] contextualized musical games; and [Greco and Benda 1998] and [Greco 2002] defined the genre of sport.

Through the work of the authors: [Green and Bavelier, 2007], [Rebetez and Betrancourt 2007], [Bayer and Renou 2011], [Polya 1945], [Rankin et al. 2006], [Greco 2002] and among others, it was shown the existence of the relation between the genres of games and cognitive processes. Each author demonstrated that a particular genre can exercise at least one of the cognitive processes involved in the work.

PROPOSED CLASSIFICATION

From the study of cognitive processes and their relationship to the act of play, it is proposed a classification of cognitive games in a set of six groups (Figure 1), on the foundations of Norman in 1993 (reviewed in [Preece et al. 2005; Matias and Greco 2010]).

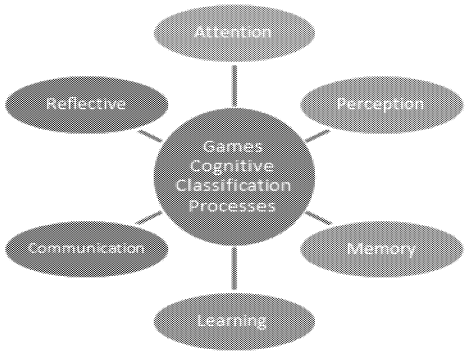


Figure 1: Proposal for classification of cognitive processes covered in electronic games.

Reflective cognition, which is associated with problem solving, planning, reasoning and decision making, is more inclined to be addressed in most games, as well as being included in most genres in his work [Dorval and Pepein 1986] indicated that individuals with these skills tend to be good entrepreneurs and proficient in math and science subjects, due to the fact they can make quick decisions and accurate.

The area of cognitive attention tends to be exercised by the genres of action, action-adventure and education.

The improvement of the spatial distribution of visual attention and visual tracking are some of the supporters of these genres.

Perception has shown an inclination to be worked on action games, educational and musical. Development of a wide variety of perceptual tasks, such as visual perception and sound perception, are one of the highlights of these kinds of games.

The process of memory tends to be exercised in puzzle games in which recalled knowledge to evolve in the game represents one of the characteristics of the genre.

The cognition of learning is exercised by the players of RPG, simulation and educational games. Many of these games explore pedagogy in learning, training and user training.

In turn, the cognitive abilities of communication (speaking, listening and reading) have shown a tendency to be exercised in musical genres and RPG. Perception of audio conversations and practices and interpretation of the characters are some of the features addressed by genres.

Finally, the reflective cognition is inclined to be worked in the genres of action-adventure, puzzle, strategy, simulation and sports. Problem solving, planning, reasoning and decision making justify classification of the genera mentioned.

CONCLUSION

Generally, the classification of products and services can guarantee increased sales and improved quality. Furthermore, in the context of electronic games, a cognitive classification could help in the decision consumer choice and treatment of learning disorders.

In this paper we reaffirm that the moderate use of electronic games can develop at least one cognitive process of the user. Thus, we propose a classification based on studies of [Preece et al. 2005] and [Matias and Greco 2010] composed of processes: Attention, Perception, Memory, Learning, Communication and Reflective.

Through an extensive literature review we demonstrated that there is a relationship between genres of electronic games and certain cognitive processes.

Using two famous games (World of Warcraft and Age of Empires) of the RPG genre proved the possibility of making a qualitative comparison with the cognitive classification proposal.

ACKNOWLEDGEMENT

The authors acknowledge the support of the Instituto Federal de Educação, Ciência e Tecnologia do Sudeste de Minas Gerais and Laboratório de Multimídia Interativa IF Sudeste - MG. Besides financial support provided by Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – CAPES – Brasil, through Programa de Educação Tutorial do Ministério da Educação(PET-MEC).

REFERENCES

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

Romão, J.E., Canela, G., Alarcon, A., 2006: Manual da nova Classificação Indicativa/ Brasília: Ministério da Justiça. Secretaria Nacional de Justiça. Departamento de Justiça, Classificação, Títulos e Qualificação.

Preece, J., Rogers, Y. And Sharp, H. 2005. Design de interação. Porto Alegre: Bookman.

Passerino, L. M., 1998: Avaliação de jogos educativos computadorizados. *Taller Internacional de Software Educativo 98* – TISE' 98. Anais. Santiago, Chile.

Foster, A. N.,Mishra, 2009. P.: Chapter II - Games, Claims, Genres, and Learning. In: *Michigan State University*, USA. Mishigan.

Moyles, Janet R., 2002: Só brincar? O papel do brincar na educação infantil.

Pase, A.F.,Tietzmann, R.: Jogos Musicais, 2009. Novos Caminhos para a Sétima Arte. In: *III Simpósio Nacional ABCiber*. ESPM/SP.

Greco, P.J. e Benda, N.R., 1998: Iniciação Esportiva Universal: da aprendizagem motora ao treinamento técnico, Volume I. Belo Horizonte: Editora UFMG.

Greco, P.J., 2002: Percepção. In: *Samulski, M.D. (Ed.). Psicologia do Esporte: Manual. Para a Educação Física, Psicologia e Fisioterapia* (pp. 55-78). São Paulo: Manole.

Green, C. S., Bavelier, D., 2007: Action-Video-Game Experience Alters the Spatial Resolution of Vision. In: *Department of Brain*

Dorval, M., Pepein, M., 1986. Effect of playing a video game on a measure of spatial visualization. Perceptual Motor Skills, 62, 159-162.

Rebetez and Betrancourt, 2007. Video Games Researth in Cognitive and Educational Sciences. Departament of Psychology and Educational Sciences Genevoa University.

Polya, G., 1945: How to Solve It.: A New Aspect of Mathematical Method. Princeton University Press, Princeton, N.J.

Bayer, R. C., Renou, L., 2011: Cognitive abilities and behavior in strategic-form games. In: University of Leicester.

Rankin, Y., Gold, R. E Gooch, B., 2006. 3D Role-Playing Games as Language Learning Tools. In *European Association for Computer Graphics*(EUROGRAPHICS).

WEB REFERENCES

Landim, W., 2011: Electronic Magazine Tecmundo: http://www.tecmundo.com.br/infografico/9708-o-tamanho-da-industria-dos-video-games-infografico-.htm [Accessed 17 May 2012].

Hotz, Robert Lee, 2005: When Gaming Is Good for You: http://online.wsj.com/article/SB10001424052970203458604577263273943183932.html [Accessed 17 May 2012].

Barthelemy, C., 2009: http://edition.cnn.com/2009/HEALTH/10/27/gaming.health.senses/index.html. [Accessed 10 June 2012].

Grace, L., 2005. Game Type and Game Genre. http://www.lgrace.com/documents/Game_types_and_genres.pdf [Accessed 11 June 2012].

BIOGRAPHIES

ALEX FERNANDES DA VEIGA MACHADO has a PhD in Computer Science at the Universidade Federal Fluminense. He is professor of Bachelor of Computer Science at IFSUDESTE-MG, Rio Pomba Campus. Has experience and serves mainly on the following topics: game development, artificial intelligence. He is currently Coordinator of the Laboratory for Interactive Multimedia IFSUDESTE-MG.

ISMAEL ANTÔNIO BATISTA studied computer science at the IFSUDESTE-MG. He is aresearch assistant at the group of Prof. Alex Fernandes da Veiga Machado, His main research interests are games, artificial intelligence and holograms.

MARLON CUNHA SANTIAGO studied computer science at the IFSUDESTE-MG. Rio Pomba. He is aresearch assistant at the group of Prof. Alex Fernandes da Veiga Machado, His main research interests are games.

RAFAEL RODRIGUES PADOVANI studied computer science at the IFSUDESTE-MG. Rio Pomba. He is a research at the group of Prof. Alex Fernandes da Veiga Machado, His main research interests are games .

BRUNO SOARES GAUDERETO is Professor at the IFSUDESTE-MG, Rio Pomba Campus. Pedagogue from UNIPAC (1998) and Master in Education from UFRRJ (2005).

ESTEBAN WALTER GONZALEZ CLUA Professor at Universidade Federal Fluminense and general coordinator of the UFF Medialab.PhD in Informatics from PUC-Rio. His area of expertise is particularly focused in the area of real-time computer graphics, Video Games, Virtual Reality, GPUs, visualization and simulation. It is one of the founders of SBGames, SBC, Director of the IGDA Rio Academy, responsible for research and academy in the area of digital entertainment in the country.

SANDRO DE CARVALHO PAIVA has specialization in Developing Web Application for the Center of Higher Education of Juiz de Fora. He graduated in Computer Science from the Faculty Ubaense Ozanam Coelho. He is professor of Federal Institute of Education, Science and Technology Southeastern Minas Gerais - Rio Pomba Campus. He has experience in computer science, with emphasis on Web.

SERIOUS GAMES IN A SOCIAL CONTEXT

Anton Eliëns
multimedia @ VU University Amsterdam
Creative Technology University of Twente
email: eliens@cs.vu.nl

KEYWORDS
serious games, education, game design

ABSTRACT

This paper reports about a newly developed course on serious gaming[1], with as a special focus behavioral change in a social or societal context. The purpose of this paper is to share our insights and references so that educational institutes may find inspiration to develop courses in *serious gaming* along this line. In the paper, we provide references to theoretical backgrounds, an overview of the structure and ingredients of the course, as well as a description of a game design workshop, with *civic order* as it's main theme, based on a case study of an actual problem area in the city of Amsterdam.

INTRODUCTION

Serious games are more and more considered to be an effective means to bring about awareness, acquire skills, change behavior, and influence social patterns.
Our students, perhaps even more than we (the lecturers) know what it is to adopt a gaming attitude to institutions, and perhaps even life itself, as eloquently phrased in a quote from McKenzie Wark (2007):

> ... ever get the feeling that life's a game with changing rules and no clear sides, one you are compelled to play, but cannot win. Welcome to gamespace. Gamespace is where and how we live today.

Motivated by the potential interest of students, as well as a growing recognition of the societal relevance of serious gaming, as illustrated by a recent report of the dutch foundation STT (Institute for the Future of Technology) entitled SERIOUS GAMING[2] (in dutch), and the activities of the applied research institute T-Xchange[3], which has as its mission: *serious gaming for innovation and change*, we developed a new course, *serious gaming*, with as its main theme: *serious games in a social context*.
With elementary game development technology, the students will explore the potential of serious games, using casual game mechanics, and what recently has been identified as the dynamics of gamification.
Games may in a general fashion be regarded as means to acquire skills and develop attitudes. The core mechanism of games consists of rules, with which the player interacts by so-called game-mechanics, actions that result in feedback on the player's performance. In such a way the player may develop habits, that lead to improved game playing over time. As observed in Juul (2010), casual games, which usually have a generally acceptable topic and simple mechanics are for *the player in all of us*, and to quote Flanagan (2009):

> games that depict everyday activities such as communication, social negotiation, caring for elements or characters that are part of a game world, or stabilizing precarious situations have become extremely popular with female players.

A distinguishing feature of serious games is, simply, that they are not meant for entertainment only, but that somehow learning, or awareness, occurs, in a particular domain, as exemplified by the topics listed below:

- awareness – world problems / social dilemma(s)
- education – language / mathematics / history
- health & well-being – skill(s) & remediation
- experience(s) – playful application(s)

Examples of serious games that may benefit individual health and well-being, and help to obtain skills as a remediattion for personal problems, include:

- team up – www.girlsinc.org/gc/page.php?id=6.2
- dangerous situation(s) – www.ditto.com.au
- communication method(s) – www.webwisekids.org
- muscle rehabilitation – on the move
- physical exercise(s) – www.silverfit.nl
- fitness – www.virtuagym.com
- overcome fear(s) – www.vrphobia.com

Such games will be more and more important in a society that becomes increasingly complex, stressfull and that imposes high demands on the endurance and stability of individuals, and may be regarded as a complement for serious games that address issues of cooperation and civic order.

[1] serious.eliens.net
[2] www.stt.nl/uploads/documents/219.pdf
[3] www.txchange.nl

The structure of this paper is as follows. We will start by looking at some basic considerations of serious games, and then discuss the theoretical background of serious games in somewhat more depth. After a more extensive description of the structure of the course, including the assignments, we will briefly introduce our *project utopia*, which asks the student for an explicit reflection on the norms and values of our society, underlying our behavior, followed by a brief characterization of a workshop game design, with *civic order* as its theme, based on a case study in the city of Amsterdam. We conclude with a reflection on the value of serious games in a social context, and more general remarks about the content and scope of the course.

BASIC CONSIDERATIONS

Our previous work on serious gaming includes ICT, Eliens & Chang (2007c), a masterclass game development, Eliens & Bhikharie (2006), creating a community of learners in SecondLife, Eliens et al. (2007a), the development of a climate game, Eliens et al. (2007b), learning chinese, Eliens (2010), how to present mathematics using game technology, Eliens & Ruttkay (2009). as well as a more theoretically oriented reflection on using replay to allow for learning from the actual game play by looking back, in a replay mode, at the actual choices made, Eliens & Ruttkay (2008).
The difference between ordinary games and serious games is, or should be, somewhat elusive, that is in terms of fun and entertainment there should not be (too) much difference, and as observed in Koster (2006), *fun in games* often consists of exploring the game space and slowly mastering the skills needed to deal with the challenges presented.
When serious gaming is applied for remedial purposes, in education or health care, or for effecting a change of civic order, as for example in controlling behavior in public spaces, we may regard serious games as (another form of) social technology, and ask the following questions:

- target(s) – which (group of) people?
- sponsor(s) – who initiates/pays?
- goal(s) – what behavior(s)/pattern(s)?
- instrument(s) – by what means/technology?

If we look at a specific category of serious games, as for example health games, we may observe that not only the player might be willing to pay, but also insurance companies or even the employers of the players, simply to assure better health at lower costs.
A common characteristic of many (health) games, is that they not only provide facilities for monitoring exercise and progress, but also offer essential social network support, to motivate the players/users to bring up the discipline to do the actual exercises. Being part of a community or social network has shown to be an effective instrument to encourage behavior, possibly in combination with rewards inspired by gamification dynamic(s), as discussed below.

awareness From the perspective of trans-individual problens, that is problems that concern our living space, our social community, environmental issues, and even world order, serious games that promote awareness are most relevant, with possibly as a result a more altruistic attitude towards sharing wealth, resources, even if only in the form of charitable donations.
As wellknown awareness games, we may mention:

- world hunger – www.food-force.com
- carabella goes to college – www.privacyactivism.org
- real lives – www.educationalsimulations.com/products.html
- refugee(s) – escape from woomera
- eye witness – www.mic.polyu.edu.hk/nanjing
- university politics – www.virtual-u.org
- sudan – www.darfurisdying.com

Apart from creating awareness by a more general audience, we may even ask, as pointed out in McGonigal (2007), how we can benefit from the cognitive effort(s), emotional energy and collective attention(s) of players, and more in general how we can deploy serious gaming to improve our world!
Awareness may well be a pre-condition for change. However, we live in a complex world, and actual change seems to require an adaptation of individual behavior, which may not be without cost, unless it is looked at from a different perspective.

THEORY BACKGROUND

There is a wide range of theory and scientifically interesting topics related to serious gaming, encompassing (not exclusively) the following subjects:

- psychology / behavioral economics
- complex adaptive (social) systems
- essential (economic) game theory
- gamification dynamics

Kahneman (2011) explains human irrationality in decision making as a result of using heuristic shortcuts, for example based on strong representations in memory due to priming and recency effects, and bias, which may result from *framing*, that is the way a question or dilemma is posed.
The limits of (human) rational decision making are also examined in Thaler and Sunstein (2008), but from a more political perspective. The authors introduce the notion of *liberal paternalism*, and examine the ways that decision making can be influenced by a proper

architecture of choice and, while retaining the freedom of choice implied by their liberal orientation, nudge(s), that is a push in *the right direction*, which of course is (always) a matter of perspective!
From (complex) systems theory, Axelrod & Cohen (1999), we learn that there are no easy solutions, and in particular, it is nowadays generally acknowledged that social networks and related mechanisms play an important, if not essential role, in the adoption of ideas and behaviors, Easley & Kleinberg (2010).
Finally, from Klein (2009) we learn that, even if we were able to think rationally about our behavior and decisions, once we are under pressure we might easily forgo our (good) intentions, and rely on our (wrong) habits. Whether rational or intuitive, the formation of the right habits may be considered to be a long process of building of expertise and experience, requiring a suffcient amount of awareness and self-discipline, McGonigal (2012).

essential (economic) game theory More than we perhaps may think, (economic) game theory may be used to analize our daily life, our domestic conflicts, issues of global war and peace, and (for example) meeting with strangers, Fisher (2008).
A typical (symmetric) payoff matrix for a two-person non-zero sum game, as used in game theory looks as follows:

A/B	cooperate	deflect
cooperate	R/R	S/T
deflect	T/S	P/P

where T = temptation, R = reward, P = punishment, S = sucker.
Symmetric in this context means that rewards and punishments are equal for both players.
The most well-known example is, no doubt, the prisoner's dilemma[4], for which $T>R>P>S$, giving a so-called Nash equilibrium for deflection, thus jeopardizing the mutual benefits that may result from cooperation, which is technically known as a Pareto equilibrium, Barash (2003).
It is interesting to note that there is a winning strategy for the prisoner's dilemma, that starts with a cooperative attitude, but is easily provoked into deflection, once the opponent appears to deflect. This strategy is aptly named *tit-for-tat*[5], Barash.
Another, less wellknown dilemma is called *chicken*:

A/B	cooperate	deflect
cooperate	live/live	coward/girl
deflect	girl/coward	dead/dead

The game has become (in)famous from the movie *rebel without a cause*, where two drivers approach a ravine, only to be called chicken when jumping first out of the car, and obtaining the girl otherwise. For *chicken* the order of values is: $T>R>S>P$.
It may be observed that *chicken* is often played on the side-walks or in supermarkts, when meeting strangers, who is the first to go out of (y)our way? Chicken!
Tragically, cooperative behavior never seems to have a stable equilibrium, thus the default action, that is the action with the best payoff, always must be *deflection*. The dilemma between deflection and cooperation can even be more dramatically phrased as the choice of taking the risk to be a sucker or the courage to be a saint, Barash (2003). That this dilemma holds can be seen in the comparative values for the following games:

- prisoner(s) dilemma: $T>R>P>S$
- chicken: $T>R>S>P$
- leader: $T>S>R>P$
- free loader: $R>T>S>P$

The leader game is the familiar situation that you are both waiting to enter a door. Who is the first to go? If no one takes the initiative, there is a deadlock!
The freeloader game represents the behavior of a person that profits from the efforts or resources of a community, and is also known as the *tragedy of the commons*[6].
As an historical aside, both the prisoners dilemma and chicken played a prominent role during the cold war, and were actually developed (that is identified) by the RAND cooperation and deployed in their (serious!) war games, Levone et al. (1991).
Not all hope is lost, though, when we consider the evolutionary need for cooperation, that is, for our survival, Barash (2003). As the *tit-for-tat* prize-winning example indicated, playing games in succession is different from single, one-session, games. Moreover, as argued in Fisher (2008), apart from repetition, kinship and proximity lead to conditions of trust under which cooperation is likely to occur, although not necessarily!
In our current day society, we must however more strongly impose cooperative behavior, by laws, by enforcing civic rules, and, more in general, reinforcement(s), Skinner (1971), that may well be understood as *operant conditioning*, that is by using punishments and rewards, not necessarily with as gentle an approach as *nudges*, Thaler and Sunstein (2008).
Serious games hold the promise of developing proper attitudes and habits, so that exercises in adequate behavior may be internalized by repetiton, and, with sufficient self-discipline, lead to what is called *transformative experience*(s) in martial art(s) and yoga, and why not travel, allowing us to become, indeed, better persons, McGonigal (2012).

gamification dynamics Reward systems may be considered to be the essence of the new trend of *gamification*, whether applied to areas of domestic or office

[4] en.wikipedia.org/wiki/Prisoner's_dilemma
[5] ingrimayne.com/econ/IndividualGroup/TitForTat.html
[6] serious.eliens.net/dilemmas.html

quarrels[7], health[8], exercise and running[9] or practicing for MBA admission exams[10]. An interesting example is the gamification of public space using physical interaction, as explored by *the fun theory*[11]. As explained in Zicherman & Cunningham (2011), the primary goals of gamification are to build *engagement*, *loyalty* and *commitment*, using a proper system of rewards, such as: status, badges, experience points, etcetera. Tricks or mechanisms that may be used in gamification are, among others:

- appointment(s) – in which you must succeed / in time
- influence & status – achievement(s) / I want this!
- progression(s) – towards completion(s) / monitor(s)
- communal discovery – cooperation(s) / reward(s)

To be effective, however, such dynamics must be accompanied by or instrumented using proper rules and game mechanics, since (implicit) rules are usually a better way to modify behavior than words or visual decorations, or as observed in Bogost (2007), what we need, to bring about behavioral change, is *procedural rethoric*, that is *the art of persuasion through rule-based representations and interaction, rather than the spoken word.*

STRUCTURE OF THE COURSE

The course will take a multi-disciplinary approach, accommodating the variety in background and interests of the students, which may cover the range of game concept design, including social game dynamics, societal issues and game technology, including both programming, asset development and delivery and deployment issues. The course will cover two months of intensive work, of which the first month will be devoted to learning elementary game development techniques, and the second month to develop the serious game application, including an assessment of the (potential) effectiveness of the approach.
Apart from theoretical lectures, there will be regular workshops and presentation sessions in which the students present their work and get feedback.
A provisional schedule of the course looks as follows:

1. introduction(s) – the team & finding (y)our topic(s)
2. theoretical background(s) – narrative(s) & human(s)
3. miscellaneous – pitch / design(s) & gamification(s)
4. game design workshop(s) – express (y)our idea(s)
5. student presentation(s) – concept(s) & plan(s)
6. reflection(s) – ethical aspects of serious games
7. final presentation(s) – concept(s) & prototype(s)

[7] www.chorewars.com
[8] healthmonth.com
[9] nikeplus.nike.com/plus
[10] www.beatthegmat.com
[11] www.thefuntheory.com

Students are required to work in teams of 2-4 people, with as a goal the actual development of a serious game, with social network support.

assignment(s) The assignments consist of basic exercises and a final project. As basic exercises we offer a theoretical task, to practice academic skills, as well as a practical task, to gain familiarity with the technology:

- project utopia – brief description of ideal society and potential role of serious games
- moodspace – exercise in visual rethorics, preferably in unity

Our choice for the unity3D[12] is motivated partly by our previous experiences as well as the availability of a free (indie) version for students.
Since we expect a wide variety of backgrounds with students following the course, we have formulated our final project assignments accordingly. Students must make a choice out of one of the following options:

1. prototype(s) – with sufficient documention
2. concept design – with narrative(s) and visual design
3. trailer – promotion clip, with business plan
4. reflection(s) on societal impact – with sufficient depth

Options (1) and (2) are the recommended ones, (3) is viable only for business-oriented students and (4) is actually discouraged, unless the student has a strong theoretical background and interest.

benefits & pitfalls of the course: In summary, in the course as sketched above, students are expected to gain awareness of game design, become familiar with the practical use of game technology, game concept development, and practice communication and project planning, as well as cooperation in a multi-disciplonary team, and the delivery of oral and written reports.
A critical issue is the choice of suitable topics, which is preferably done with an external partner. A sufficient level of technical expertise is required, at least for a majority of students following the course. The structure of supervision should be such that creativity is stimulated, in order to maintain a high level of motivation.

PROJECT UTOPIA

What is an ideal society? And what role(s) can serious games play in the transformation of our society, and help accomplish improvements in, for example, environmental issues, education, health and civic order? For both creative technology students and students multimedia & game development it is worthwhile to reflect on such issues in a purely intellectual fashion,

[12] unity3d.com

a craft too easily forgotton in our academic institutions, and give a brief description of their own ideas, in their own words, of the elements constituting a (potentially) ideal society, that we name, for convenience as well as historical reasons, utopia:

- environment(s) – facilitator(s), infrastructure ...
- system(s) – organization(s), incentive(s) ...
- rule(s) – code(s) of law, civic order ...
- (moral) value(s) – utility, behavior(s), ethic(s) ...

The need for a reflection on the (moral) values underlying our society, became even more clear to me after a year traveling in China, Clearly, as argumented in Sandel (2012), there is more to moral(s) than economic value only!

workshop game design – civic order(s)
Our workshop game design 2012 will have civic order as its main topic, and focus on means to establish citizen's participation in local neighborhood(s), with a case study of one of Amsterdam's city areas as a starting point. The case study and assignment(s) will be presented by a member of the Amsterdam city council. Following the structure of a game design workshop, as described in Eliens (2010), the assignments which must lead to a group presentation in less than one hour and a half are:

- (y)our player(s) ... ?
- what super power(s) ... ?
- invitation(s) – message(s) !
- (mini) game – mechanic(s) ?
- nudge(s) – re-enforcement(s) ... !?

The role of a superpower is here left somewhat ambiguous, since it may refer to a superpower to be acquired by the player (that is the target audience) or a superpower that is represented in the game by (artificial) opponents, Mark (2009).
When selecting (mini) game mechanics, the designers must keep in mind that, as indicated in Zicherman & Cunningham (2011), only a minority of the players consists of *killers* or *achievers*, and (in general) the vast majority participates for *socializing*. However, this division may be different for our target group(s)!

LET'S BE SERIOUS!

In one of my first papers on this topic, Eliens & Chang (2007c), I observed that – ICT is not a (simple) game. When speaking about civic order(s), a similar phrase might be either taken as an understatement or even as a warning, given the need expressed all over the world to guide and control (or nudge) the behavior of citizens, in urban areas as well areas threatened by the effects of consumerism, with as a dramatic example the rural areas of the country I recently visited, China, where pollution due to production and consumption is a number one threat. I gained more insight in the China Dream[13] workshop, in which I participated at the end of my stay in China, the goal of which was *to reimagine prosperity and reshape consumerism in China ... (and) to catalyze a new aspirational lifestyle that is innately sustainable for the emergent middle class in China.*
Our question, as addressed in this paper, is how can we deploy serious games to counteract problems of personal health, order in public spaces, and in general civic behavior that leads to a sustainable society.
Following Bronowski (1956), what morals can science teach us? What is a community of learners, Eliens et al. (2007a), if not one in which truth and freedom of thought are values to be respected by everyone? During my travels[14], and back at home, I see people dump garbage on children's playgrounds, factories polluting the environment, people suffering from diseases due to contaminated food. Can science help serious games to change human behavior?
We may end by asking: why do people play games? According to Zicherman & Cunningham (2011) thst may be for reasons such as mastery (of a skill), to de-stress (from work life), simply for fun, or to socialize, with socializers a clear majority! And rephrasing the question from the perspective of game design, how can we design (serious) games that appeal to the people that play games and at the same time bring about changes both in awareness and (individual) behavior?

CONCLUSIONS

In this paper[15] we have reported on our efforts to set up a course on *serious gaming* that covers the various theoretical and practical areas related to serious games, including behavioral economics, complextity science, gamification dynamics and more general topics in (casual) game design.
As a distinguishing feature of serious games, we emphasize the moral aspect and the intention to bring about an enduring change of behavior, due to increased self-control, guidance and support by social networks using an adequate system of rewards and nudges, or encouragements, and, more in general, an awareness of the issues involved on both a personal and social level, creating the willingness to cooperate without fear of being regarded as a sucker, and preferably without the need or desire to be regarded as a saint.
Hopefully, our approach brings about some clarification with respect to the potential of serious games, if not to the reader, then at least to our students, which is, after all, our primary audience.

[13] www.juccce.org/chinadream
[14] aeliens.wordpress.com/2012/02/10/minimalisms
[15] serious.eliens.net/paper-social.html

ACKNOWLEDGEMENTS
Thanks to my collegues from the University of Amsterdam, Jacobijn Sandberg and Frank Nack, for their positive encouragements and support.

REFERENCES

Axelrod R. & Cohen M.D. (1999), *Harnessing Complexity: Organizational Implications of a Scientific Frontier*, Free Press

Barash D.P. (2003), *The Survival Game: How Game Theory Explains the Biology of Cooperation and Competition*, Henry Holt books

Bogost I. (2007), *Persuasive Games – the expressive power of videogames*, MIT Press

Bronowski J. (1956), *Science and Human Values*, Julian Messner Inc.

Easley D. & Kleinberg (2010), *Networks, Crowds and Markets – Reasoning about a Highly Connected World* , Cambridge Press

Eliens A. and S.V. Bhikharie (2006), *Game @ VU – developing a masterclass for high-school students using the Half-life 2 SDK*, Proc. GAME'ONNA'2006, Monterey, USA

Eliens A. Feldberg F., Konijn E., Compter E. (2007a) , VU @ Second Life – creating a (virtual) community of learners, Proc. EUROMEDIA 2007, Delft, Netherlands

Eliens A., van de Watering M., Huurdeman H., Bhikharie S.V., Lemmers H., Vellinga P. (2007b), Clima Futura @ VU – communicating (unconvenient) science, Proc. GAME-ON 07, Bologna, Italy

Eliens A. & Chang T. (2007c), Let's be serious – ICT is not a (simple) game, Proc. FUBUTEC 07, Eurosis, Delft, April 2007

Eliens A. and Ruttkay Z. (2008), Record, Replay & Reflect – a framework for understanding (serious) game play, Proc. Euromedia 09, Brugge, Belgium

Eliens A. and Ruttkay Z. (2009), Math game(s) – an alternative (approach) to teaching math?, Proc. GAME-ON 2009, Dusseldorf, Germany

Eliens A. (2010), Elements of a chinese language game, Proc. GAME-ON 2010, Shanghai, China

Eliens A. (2010), creative technology – the CTSG: game design in 7 steps, Proc. GAME-ON 2010, Shanghai, China

Fisher L. (2008), *Rock, Paper, Scissors: Game Theory in Everyday Life*, Basic Books

Flanagan M. (2009), *Critical Play – Radical Game Design*, MIT Press

Juul J. (2010), *A Casual Revolution – reinventing video games and their players*, MIT Press

Kahneman D. (2011), *Thinking, Fast and Slow*, FSG Books

Klein G. (2009), *Streetlights and Shadows: Searching for the Keys to Adaptive Decision Making*, MIT Press

Koster R. (2006), *A Theory of Fun for Game Design*, Paraglyph Press

Levine R., Schelling T, Jones W. (1991), Crisis Games 27 Years Later: plus c'est dejavu, RAND Corporation

Mark D. (2009), *Behavioral Mathematics for Game AI*, Charles River Media

McKenzie Wark (2007), *Gamer Theory*, Harvard University Press

McGonigal J. (2007), *Reality Is Broken: Why Games Make Us Better and How They Can Change the World*, Perseus Press

McGonigal K. (2012), *The Willpower Instinct : How Self-Control Works, Why It Matters, and What You Can Do To Get More of It*, Penguin Group

Miller J.H. & Page S.E. (2007), *Complex Adaptive Systems: An Introduction to Computational Models of Social Life*, Princeton University Press

Sandel M.J. (2012), *What Money Can't Buy: The Moral Limits of Markets*, D&M Publishers Inc.

Skinner B.F. (1971), *Beyond Freedom & Dignity* , B.F. Skinner Foundation

Thaler R.H. & Sunstein C.R. (2008), *Nudge: Improving Decisions About Health, Wealth, and Happiness*, Caravan Books

Zicherman G. & Cunningham C. (2011), *Gamification by Design: Implementing Game Mechanics in Web and Mobile Apps*, O'Reilly Media Inc.

BIOGRAPHY

Anton Eliëns (PhD) is lecturer and coordinator of multimedia @ VU University Amsterdam, and works at the University of Twente as professor creative technology / new media. He has experience in web-based interactive media, interactive video, game engines and the application of such technologies in serious games.

GAME ANALYSIS

A Framework for Quantitative Analysis of User-Generated Spatial Data

Tom Feltwell, Patrick Dickinson, Grzegorz Cielniak
School of Computer Science,
University of Lincoln, UK
email: tfeltwell,pdickinson,gcielniak@lincoln.ac.uk

KEYWORDS

Game Metrics, Spatial Data Analysis, First Person Shooter, Histogram Comparison

ABSTRACT

This paper proposes a new framework for automated analysis of game-play metrics for aiding game designers in finding out the critical aspects of the game caused by factors like design modifications, change in playing style, etc. The core of the algorithm measures similarity between spatial distribution of user generated in-game events and automatically ranks them in order of importance. The feasibility of the method is demonstrated on a data set collected from a modern, multiplayer First Person Shooter, together with application examples of its use. The proposed framework can be used to accompany traditional testing tools and make the game design process more efficient.

INTRODUCTION

The constantly increasing complexity of modern video games poses new challenges for game designers. The design teams not only have to fulfil the growing demands with respect to quality, realism and detail of digital assets but also ensure that the game is fun, challenging and free from major loopholes. The design process is traditionally verified by different testing stages allowing for detection and identification of most critical shortcomings. Traditional testing tools usually involve questionnaire-based feedback or biofeedback provided by testers (Ambinder 2011) and analysis of so called game metrics information generated by users during the actual game-play. The amount of information from the testing process necessary to be analysed by the designers is growing together with the complexity of games and can quickly lead to data overload, where there are too many factors to be considered and correlate with each other. Therefore there is a growing need for new automated methods for metrics analysis that are scalable and efficient.

This paper proposes a new framework for automated analysis of game-play metrics that will aid game designers in finding out the critical factors caused by factors like design modifications, change in playing style, etc. The core of the algorithm measures similarity between spatial distribution of critical game events and feeds that information back to the designer indicating critical differences in the generated data. The proposed framework can be used to accompany traditional tools and make the design process more efficient.

The paper is organised as follows: the next section describes relevant work in the area of quantitative analysis of user-generated data, then the details of the proposed framework are presented. The applicability of the method is then investigated in the experimental section followed by conclusions and discussion of future work.

RELATED WORK

Statistical summaries of different in-game events generated by the player (also called game metrics) provide invaluable information to the game designers. For example, unbalanced weapon usage, high number of deaths or long level completion time can indicate serious design issues and can be quickly identified during the testing stage of the game development. Indirectly, this information can also provide insights into player's experience and can be used together with traditional methods for measuring player's experience that are based on subjective questionnaires or biofeedback (Davis et al. 2005, Tychsen 2008). The importance of statistical analysis of user generated data has been recently noted by the game industry and resulted in a number of existing systems for data collection and analysis (Kim et al. 2008, Wallner and Kriglstein 2012).

Many of the important in-game events relate to a specific location in the environment, such as defensive positions, attack routes and supply points. In such a case, not only overall frequency of these events is important but also their spatial distribution. Spatial histograms (or so called heatmaps) are a popular tool in video game industry for representing such data as they can be conveniently visualised and used for further inspection (Drachen 2011, Thompson 2007). Alternative approaches for spatial data visualisation include for exam-

ple clustering algorithms that group closely occurring events into meaningful nodes (Tychsen 2008).

The majority of the current methods for in-game spatial analysis rely on visual inspection by the game designer, who needs to take into account not only the shape of event distribution, but also *changes* in that distribution. With so many factors to be taken into account the process can be very tedious and time consuming. In this paper, we propose a step toward automated techniques for comparing and characterising changes in in-game event distribution. Our approach is inspired by techniques popular in other research areas like computer vision and pattern recognition. Example applications include image indexing (Swain and Ballard 1991), object tracking (Bradski and Kaehler 2008) and object recognition (Gevers and Smeulders 1997).

Figure 1: Screenshot from Red Orchestra: Ost Front 41-45. Image Copyright: Tripwire Interactive

METHOD

This section presents details of the proposed system. First, the game scenario is presented that was chosen for illustrating concepts and techniques used in the system. However, the proposed techniques are not scenario, nor event specific and can be applied to any game type, single- or multi-player and any virtual environment where spatial data can be collected. The following descriptions present details of the four-stage framework for data processing (see Fig. 2). During stage one (S1), spatially distributed data is collected and suitably formatted from a virtual environment/game. Stage two (S2) sees the generation of spatial distribution of various events in a form of a histogram. Different histograms are then compared in stage three (S3) and finally the results of these comparisons are ranked and characterised in stage four (S4).

Game Scenario

The proposed game scenario is Red Orchestra: Ost Front 41-45 (see Fig. 1). Developed in 2006 by Tripwire Interactive, the player takes charge of various roles within either a Russian or German military unit on the Eastern Front during World War 2 (Tripwire Interactive 2006). The game is a purely multiplayer First Person Shooter (FPS), with heavy emphasis on team work. Within each team, a set of player "classes" are available, which each player must chose from before commencing play. These classes are designed to encourage the team playing aspect, as they each have unique weapons and abilities. Furthermore, some classes such as Assault Trooper are restricted to maintain balance of the multiplayer combat, and to represent realistic military squad hierarchy.

Realism is one of the key features of the game, with many simulation aspects implemented, such as bullet flight time, wound ballistics and bullet drop. Due to the complex nature of the game system, it was decided that the game would provide a good source of data for characterisation and analysis. Based on the Unreal 2.5 Engine, support and documentation was readily available via Epic Games (2004). Further more, an open source Software Development Kit was available for Red Orchestra, allowing full access to source code and level editor, which is well support by both the community and Tripwire Interactive.

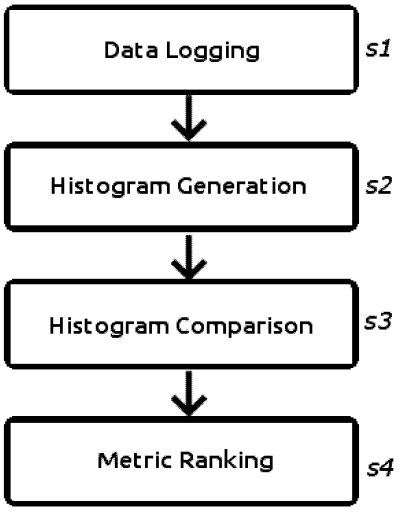


Figure 2: Stages of the Proposed Framework

Data Logging (*S1*)

The initial stage of the framework is the collection of relevant data for analysis. The collected data must have a spatial element associated within it. Most virtual environments use a coordinate system, where the spatial element of the data would consist of X,Y and Z coordinates. Some examples of spatially distributed events include the change of direction of the car wheels in a racing game or the combat engagement between two units in a real time strategy. In Red Orchestra, the players typically perform five basic actions: shooting, moving, dying, taking damage and reloading. Other relevant information related to these key events include crouching, sprinting and using a specific weapon. These events were therefore chosen for logging and further analysis.

Logging of so called *user initiated events* is commonly used in the industry and is known as *instrumentation* (Kim et al. 2008). The data logging system created for the proposed system was written in UnrealScript, the native language of Unreal Engine 2.5. In our implementation the game server stores and records all logs, with no log information transferred across the network. Per match, each player has their own log file, with a unique file name. There is an overall match log, which stores events such as changing team or capturing objectives - these are however not spatially distributed events and are recorded only as additional information. To record Move events, the player's position is polled in regular intervals (1 s. in the proposed implementation). All other events are recorded immediately as they happen. Each event contains a standard set of information: position, player name, time, event type and player stance. Depending which event is being recorded, an extra event specific set of information is appended onto the log entry. The basic event log is formatted as follows: *eventType* | *eventTime* | *playerName* | *X,Y,Z coords* | *crouched* | *prone* | *sprinting* | *limping* | *stamina*. An example of an actual event log can be seen in Fig. (3.

```
1|17:24:23|Stinkfoot|1829.50,-4232.90,283.30|1|0|0|False|13.97
1|17:24:24|Stinkfoot|1829.78,-4262.36,287.18|1|0|0|False|14.92
1|17:24:25|Stinkfoot|1859.57,-4226.42,284.05|1|0|0|False|15.79
0|17:24:25|Stinkfoot|1880.33,-4203.43,282.73|1|0|0|RO-LyesKrovy.STG44Weapon|1|0,654
1|17:24:26|Stinkfoot|1880.33,-4203.43,282.73|1|0|0|False|16.60
4|17:24:26|Stinkfoot|1880.33,-4203.43,282.73|1|0|0|el_mysterio|1883.69,-4188.46,306
0|17:24:26|Stinkfoot|1880.33,-4203.43,282.73|1|0|0|RO-LyesKrovy.STG44Weapon|1|0,118
0|17:24:26|Stinkfoot|1880.33,-4203.43,282.73|1|0|0|RO-LyesKrovy.STG44Weapon|1|0,42,
0|17:24:26|Stinkfoot|1880.33,-4203.43,282.73|1|0|0|RO-LyesKrovy.STG44Weapon|1|0,132
0|17:24:26|Stinkfoot|1880.09,-4204.52,282.73|1|0|0|RO-LyesKrovy.STG44Weapon|1|0,44,
1|17:24:27|Stinkfoot|1879.34,-4205.52,282.73|1|0|0|False|17.56
1|17:24:28|Stinkfoot|1859.37,-4256.04,287.82|1|0|0|False|18.39
1|17:24:29|Stinkfoot|1868.35,-4373.83,297.36|1|0|0|False|18.04
```

Figure 3: Example Player Event Log collected during a Data Gathering Session

Python scripts were written to parse the events contained within the log files and to store them in a SQL database, which encapsulates the events, game matches and other information. SQL format was chosen due to its popularity, simple syntax of queries and built-in optimisation for handling large numbers of records. The file names of the logs contain unique information such as time the player joined, map being played and player name. These are used to populate the SQL database initially.

Histogram Generation (*S2*)

The next step of the procedure is generation of spatial histograms from the collected data. A spatial histogram is a discrete representation of the environment, 2d in the presented case, but can easily be extended to 3d, and consisting of $n \times m$ so called bins. Each bin corresponds to a rectangular region of the environment and the value of each bin stores the total count (i.e. frequency) of the specific event that occurred in that region (see Fig. 4). In result, the spatial histogram not only provides information about the frequencies of events, but also their *spatial distribution*.

The resulting histogram can be represented as a "heatmap" - a histogram using "hot" and "cold" colours mapped to the high and low (respectively) frequency values. This gives the impression that areas with higher event counts are "hotter" than low event counts, which are "colder" (see Fig. 4c and 4d).

Two critical parameters affecting the shape of the resulting histograms are the size of the region corresponding to each bin, and the total number of events considered. Assuming that the size of the environment is fixed, larger bin size results in fewer bins or in other words, in lower histogram resolution (see Fig. 4d). On the other hand, larger bins require less events to populate the bins (i.e., result in sufficient count of events).

There are existing methods that calculate the optimal bin size depending on underlying data distribution (Shimazaki and Shinomoto 2007). However, they are usually based on some strong assumptions (e.g. Normal distribution) which do not hold in all cases. Therefore in the proposed system the number of bins is an experimentally tuned parameter.

Histogram Comparison (*S3*)

Spatial histograms of different events or generated under different conditions can then be compared to provide a single metric value that indicates their similarity/dissimilarity. Examples of different conditions are data from different time periods, (i.e. first five games compared to last five games), selection of matches with particular player numbers, or with differing sample populations, such as different teams of players. Before the comparison, each histogram has to be normalised by dividing the value of each bin by the total sum of all bins. The resulting histogram becomes a discrete approximation of probability distribution for the event. It is im-

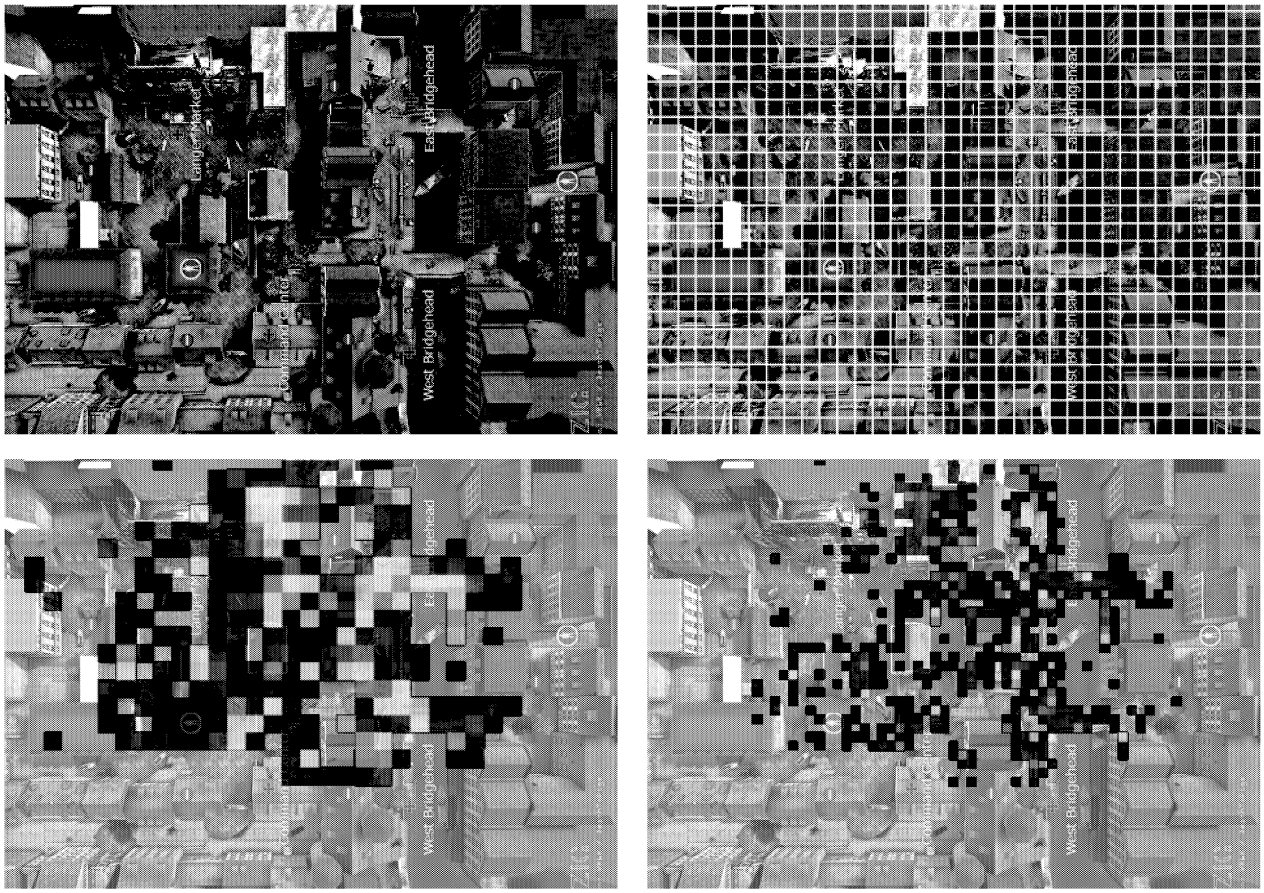


Figure 4: a) Danzig Map b) Danzig Map with White Grid Lines Depicting Histogram Bins c) The Overlaid Histogram of Shoot events (32 Bins) d) The Same Histogram with Higher Resolution (64 Bins)

portant to note that most comparison methods require histograms of the same resolution.

There are several methods that can be used for comparison of histograms (Abramowitz and Stegun 1965). Here, two popular metrics are proposed. The L_1 metric measures the amount of difference between the histograms by summing together all absolute differences between the histograms bins:

$$L_1 = \sum_{i=1}^{B} |P(b_i) - Q(b_i)|, \tag{1}$$

where $P(b_i)$ and $Q(b_i)$ are the normalised values stored in the i-th bin of the P and Q histogram, respectively, and $B = n \times m$ is the total number of bins for each histogram. The resultant metric can assume values from [0-2] range, with 0 value corresponding to two identical histograms.

A metric based on Bhattacharyya distance is used to measure the amount of overlap between two histograms (Dubuisson 2010, Thacker et al. 1997). The modified formula that expresses that value in terms of similarity for [0-1] range is as follows:

$$L_{Bh} = 1 - \sum_{i=1}^{B} \sqrt{P(b_i)Q(b_i)}. \tag{2}$$

Metric Ranking (*S4*)

The proposed comparison metrics summarise differences of a pair of histograms in a single number. Metric values can then be directly compared and provide instant information about the amount of change for each condition/event type. Higher metric values might indicate significant changes in critical variables while lower metric values might help identifying similar patterns occurring in the data. This information can direct the designer in further analysis of the results, for example by visual inspection. This way the designer avoids looking at all various possible combinations of histograms ("data overload") and can concentrate on the critical variables only.

EXPERIMENTS

The following experiments were conducted to demonstrate the feasibility of the proposed framework. This section presents details about data gathering, data sets collected and results obtained from applying the presented framework to the selected game scenario.

Data Collection

Figure 5: One of the Data Gathering Sessions

The data collection was scheduled as regular sessions in a designated area of the computer lab at the Lincoln School of Computer Science (see Fig. 5). The sessions were advertised amongst the School's staff and students. Unfortunately, due to the restrictive University network policies, it was not possible to collect data through the Internet and address a wider audience.

Each session consisted of several multiplayer matches of Red Orchestra: Ost Front 41-45, with players joining one of the two available teams, such that matches were approximately balanced. Each session was held for approximately two hours, with three maps (Danzig, Basovka and Lyes Krovy) played in a randomised order every time. Approximately 50 sessions were run over the duration of one year, with most sessions taking place during term time, due to staff and student availability.

From all available data we have chosen a representative subset consisting of 30 matches on a single map, Danzig. The number of players per match varied between 5 and 20 (see Fig. 6). The events considered for further analysis include the 5 aforementioned types including Shoot, Move, Death, Damage and Reload. The average of each event per player can be seen in Fig. (7). The collected data was split roughly in half into two sets, D_1 and D_2 for further analysis.

D_1 contained 88,048 data points, and D_2 contained 98,701 data points. The breakdown of these data sets into their respective events can be seen in Table (1). The difference between event counts for the two data

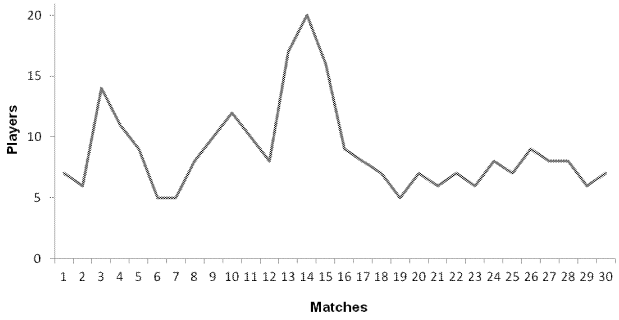


Figure 6: Player Count Per Match over the Entire Data Set

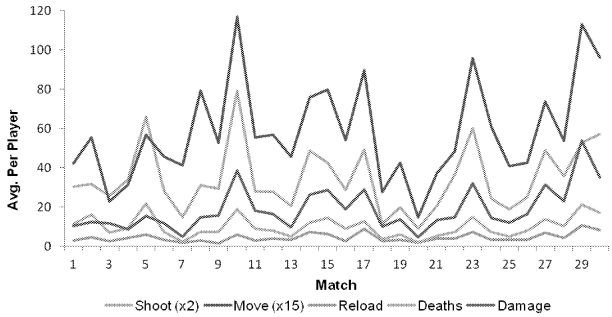


Figure 7: Average Event Count Per Player Per Match over the Entire Data Set

sets can be attributed to multiple factors, for example strong correlation can be seen between the number of players (see Fig. 6) and event counts (see Fig. 7). The Reload and Shoot event are correlated to the type of weaponry being used by players, as some weapons require reloading more often and some are able to fire shots more quickly than others.

Table 1: Total Number of Events in each Data Set

	Shoot	Move	Reload	Death	Damage
D_1	5,968	79,111	391	844	1,734
D_2	6,661	88,435	513	955	2,137

Results

With D_1 and D_2 data sets isolated, comparisons were generated for each of the five events including both proposed metrics. Fig. (8) presents L_1 and L_{Bh} metric values for each event type with respect to varying histogram resolution. The bin size parameter varied from 2 to 64 bins along the X axis; the corresponding number of bins along the Y axis has been chosen such that the resulting regions were square (e.g. for Danzig 64×44 bins) It can be seen that for small bin size the difference between different events is difficult if not impossible to

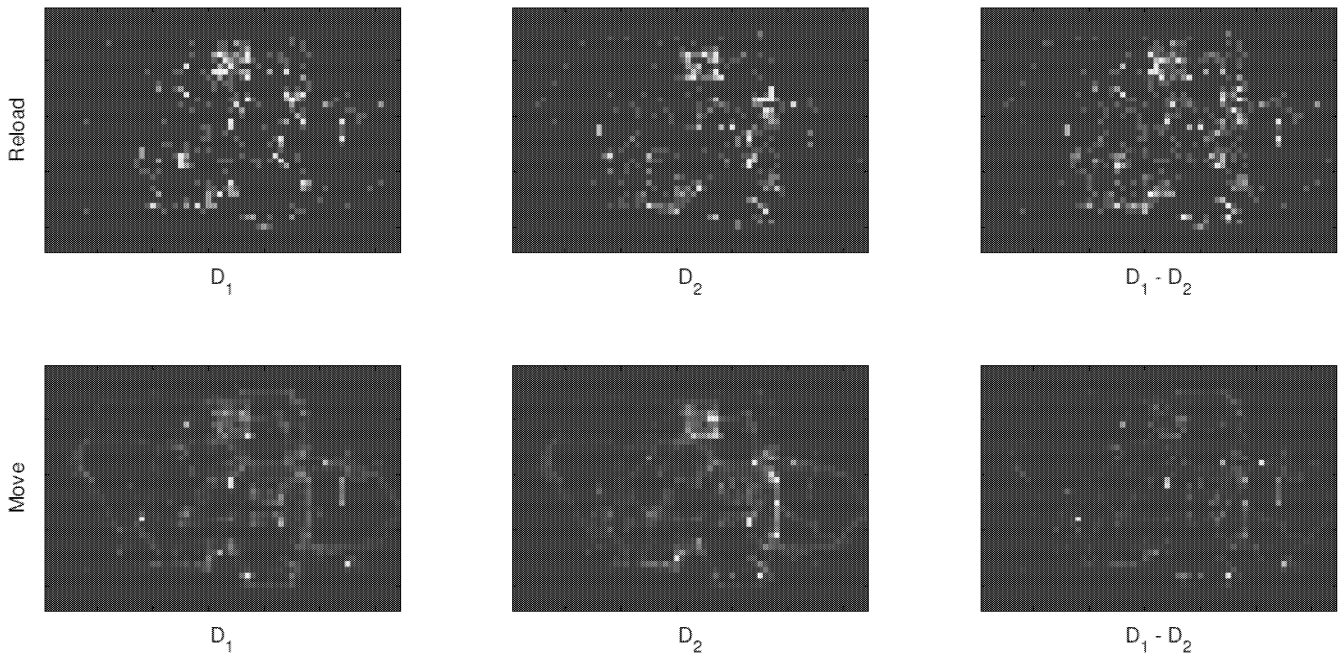


Figure 9: D_1, D_2 Histograms for Reload and Move Events, and their Differences (64×44)

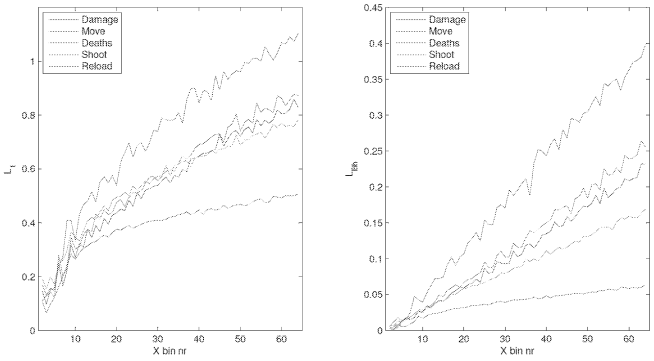


Figure 8: L_1 and L_{Bh} Comparisons over Varying Histogram Resolution

identify, due to low resolution of spatial histograms. As the number of bins increases, metric values for the different events take on their own separate characteristics. Over the remainder of the range, they maintain similar relationship to one another. With 10 bins and or less, Bhattacharyya metric presents itself as more stable, when compared to L_1. It is interesting to note in both comparison metrics, the relationship between events is broadly similar from 10 bins upward.

The results indicate that metrics for the Reload event have the highest value, which is due to large differences in the two distributions. The movement event has the lowest metric values, which might indicate similar patterns in distribution for this event. Fig. (9) provides visual guidance in the form of plotted histograms for the Reload and Move events, confirming the presented metric results. It can be seen that indeed changes in the Reload event are more visible than for the Move event. It can be seen for Reload that the main cluster at the top of the map in D_1 has shrunk in size in D_2, with another cluster appearing on the right. These two clusters match up with defensible areas of the Danzig map, indicating players defending/attacking these in different ways between the two data sets. This is validated by the comparison, which shows difference in both locations.

For Move events, "pathways" can be seen in D_1 and D_2 histograms, which indicate the walkable areas of the levels. Some pathways are used more often than others, and are represented on the histograms in "hotter" colours. The bright clusters seen on both D_1 and D_2 are areas where players spend more time. These points correspond to vantage points, spawn areas and objectives within the game world. It is interesting to note that the comparison histogram for the Move event shows overall a large number of small differences. These can be explained by gameplay conditions and player behaviour. A small number of high value differences are also present, which represent particular vantage points visited frequently by players.

The presented framework can also be used to observe temporal changes of various events. To demonstrate that functionality, the entire data set (D_1 + D_2) was divided into 8 distinct sets, and comparisons were run for each possible pair of the resulting subsets. Table (2) presents a rectangular similarity/dissimilarity matrix for each of the 8 data sets, using L_1 metric for the

Table 2: D_1 and D_2 Divided into 8 Sections, Compared with each Section using L_1 Metric, for the Move Event

	D_1^1	D_1^2	D_1^3	D_1^4	D_2^1	D_2^2	D_2^3	D_2^4
D_1^1	0.00	0.91	0.86	0.86	0.92	0.92	0.79	0.95
D_1^2		0.00	0.77	0.84	0.85	0.82	0.80	0.86
D_1^3			0.00	0.79	0.86	0.76	0.76	0.83
D_1^4				0.00	0.81	0.90	0.79	0.93
D_2^1					0.00	0.73	0.71	0.80
D_2^2						0.00	0.70	0.70
D_2^3							0.00	0.75
D_2^4								0.00

Move event. It can be noted the dissimilarities are lower for adjacent subsets from the same set (e.g., $D_1^2 - D_1^3$, $D_2^2 - D_2^3$) and in general are higher for subsets further apart (e.g. $D_1^1 - D_2^4$). Such a matrix can be used to identify similar or reoccurring event distributions for different time intervals.

Perhaps the most interesting comparison is for adjacent regions. Fig. (10) plots the comparison values for adjacent regions for each event type. It is interesting to note the relationships and changes between each metric over time, with Death and Damage events showing a small degree of an inverse relationship to the other metrics.

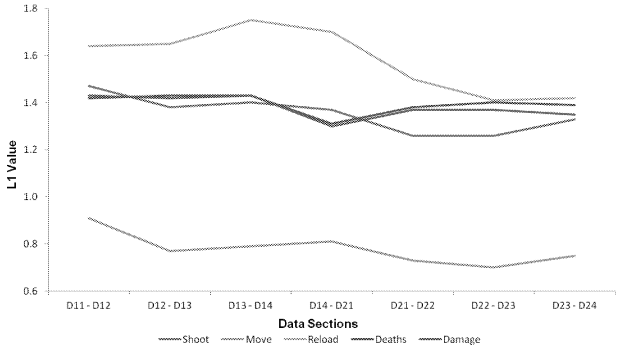


Figure 10: Comparison Metrics for Adjacent Data Sections for Different Event Types

CONCLUSIONS

This paper introduced a framework for quantitative analysis of user-generated spatial data. The presented experiments demonstrate the feasibility of the proposed method for quantifying differences between different data sets and provide example scenarios for its use. However, the presented work is still in its initial stage and further investigation is needed to address its full potential. With possibility of quantifying changes in two different data sets, it should be possible for example, to detect and identify design changes introduced by the game designer or analyse typical distributions of events for different player teams (e.g. beginners vs. advanced). This step would close the loop in the system by providing an instant feedback mechanism to the designer regarding the impact of their introduced changes on different gameplay factors. The proposed metrics provide global summaries of changes in distributions. Further work will seek to automatically identify specific regions where the most changes occur. The influence of other factors, like for example the number of players per match or a total number of events on the resulting metrics should also be investigated.

REFERENCES

M. Abramowitz and I. Stegun. *Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables*. Dover Publications, 1965.

M. Ambinder. Biofeedback in gameplay : How valve measures physiology to enhance gaming experience. *Game Developers Conference*, 2011.

G. Bradski and A. Kaehler. *Learning OpenCV: Computer Vision with the OpenCV Library*. O'Reilly, Cambridge, MA, 2008.

J. P. Davis, K. Steury, and R. Pagulayan. A survey method for assessing perceptions of a game: The consumer playtest in game design. *Game Studies*, 5:1–14, 2005.

A. Drachen. Evaluating motion: spatial user behaviour in virtual environments. *International Journal of Arts and Technology*, 4:294–314, 2011.

S. Dubuisson. The computation of the bhattacharyya distance between histograms without histograms. *Image Processing Theory Tools and Applications IPTA 2010 2nd International Conference on (2010)*, 2010.

Epic Games. Unreal Engine 2.5. http://www. unrealengine.com, 2004.

T. Gevers and A. W. M. Smeulders. Color based object recognition. *Pattern Recognition*, 32:453–464, 1997.

J. H. Kim, D. V. Gunn, E. Schuh, B. C. Phillips, R. J. Pagulayan, and D. Wixon. Tracking real-time user experience (true): A comprehensive instrumentation solution for complex systems. *Proceeding of the twentysixth annual CHI conference CHI 08*, pages 443–451, 2008.

H. Shimazaki and S. Shinomoto. A method for selecting the bin size of a time histogram. *Neural Computation*, 19:1503–1527, 2007.

M. J. Swain and D. H. Ballard. Color indexing. *International Journal of Computer Vision*, 7:11–32, 1991.

N. A. Thacker, F. J. Aherne, and P. I. Rockett. The bhattacharyya metric as an absolute similarity measure for frequency coded data. *Kybernetika*, 34:363–368, 1997.

C. Thompson. Halo 3 : How microsoft labs invented a new science of play. *Wired*, 15:1–7, 2007.

Tripwire Interactive. Red Orchestra: Ost Front 41-45. http://www.redorchestragame.com, 2006.

A. Tychsen. Crafting user experience via game metrics analysis. *NORDICHI 2008*, pages 1–5, 2008.

G. Wallner and S. Kriglstein. A spatiotemporal visualization approach for the analysis of gameplay data. *Proceedings of the 2012 ACM annual conference on Human Factors in Computing Systems*, pages 1115–1124, 2012.

AUTHOR BIOGRAPHIES

TOM FELTWELL completed his B.Sc in Games Computing at the University of Lincoln in 2010. Upon his graduation he commenced the Masters by Research program at the same institute. He now works full time within the School of Computer Science as a Technician and is completing his degree part-time. His research areas are games related, particularly in the areas of simulation, realism and gameplay analysis. `tfeltwell@lincoln.ac.uk`

PATRICK DICKINSON has been a Senior Lecturer at the University of Lincoln since 2008. He has a Ph.D from Lincoln for his work in computer vision, and has previously studied at the Universities of Oxford and Southampton. His current research interests include the application of computer vision to the surveillance of wildlife, and games AI. He has previously worked at Rebellion Developments, and Awesome Developments, and worked on a number of published game titles. `pdickinson@lincoln.ac.uk`

GRZEGORZ CIELNIAK is a Senior Lecturer at the School of Computer Science, University of Lincoln, UK. He obtained his Ph.D. in Computer Science from Örebro University, Sweden in 2007 and his M.Sc. in Robotics from Wrocław University of Technology, Poland in 2000. His research interests include physics simulation and metric-based assessment of game play, mobile robotics, machine perception, behaviour analysis and monitoring of humans and other species. `gcielniak@lincoln.ac.uk`

Finding fast solutions to the game of Mastermind

JJ Merelo, A.M. Mora
Dept. Computer Architecture and Technology + CITIC
University of Granada
email: (jmerelo|amorag)@geneura.ugr.es
C. Cotta
Dept. of Computer Sciences and Languages
University of Málaga
email:ccottap@lcc.uma.es

KEYWORDS
Mastermind, oracle games, puzzles, evolutionary algorithms

ABSTRACT

There are two main research issues when playing the game of Mastermind: one of them is finding solutions that are able to minimize the number of turns needed to find the solution, and another is finding methods that scale well when the size of the search space is increased. There is a trade-off between these two paths, and in this paper we will present a method that uses evolutionary algorithms to find fast and scalable solutions to the game of Mastermind. The advantages of this method are the few parameters that are needed to tune it and its robustness even with different parameter settings. We will show the results for different sizes and compare them with published results, showing that there is not a big difference with the best solutions found so far but the number of evaluations needed to find them is considerably reduced and it even scales much better than solutions tuned for minimization of the number of turns needed to win.

Introduction

Mastermind [Knuth (1976-77), Aldebert and Widenius (1995), Montgomery (1992)] is a puzzle in which one player A hides a combination of κ symbols and length ℓ, while the other player B tries to find it out by playing combinations coded in the same alphabet and length. Player A answers to every combination includes the number of symbols in the combination that are in the correct position and the number of colors that have been guessed correctly. Player B then plays a new combination, until the hidden one is found. The objective of the game is to play repeatedly minimizing the number of turns needed to find the solution.

Most solutions so far use the concept of *eligible* or *consistent* combinations: those that, according to responses by player A, could still be the hidden combination or, in another words, those that match the played combinations as indicated by the answer. Exhaustive methods would eliminate all non-consistent solutions and play a consistent one, while non-exhaustive methods would sample the set of consistent solutions and play one of them. Those solutions are guaranteed to reduce the search space at least by one, but obviously combinations have a different capability of reducing the search space. This *capability* is reflected by a score. However, scores are heuristic and there is no rigorous way of scoring combinations. To compute these scores, every combination in turn is compared with the rest of the combinations in the set; the number of combinations that get every response (there is a limited amount of possible responses) is noted. Eventually this results in a series of *partitions* in which the set of consistent combinations is divided by its *distance* (in terms of common positions and colors) to every other.

These partitions are used to assign a score to every combination; several scores have been proposed:

- *Worst case* was proposed by Knuth as the first rigorous strategy to play Mastermind; combinations are scored according to the size of the biggest partition; the bigger, the worse.
- *Most parts*, proposed in Kooi (2005), takes into account only the number of non-zero partitions.
- *Entropy* which has been used in Neuwirth (1982), Bestavros and Belal (1986), Cotta et al. (2010), computes the entropy of partitions, and tries to maximize it
- *Expected size* used by Berghman et al. (2009), Irving (1978-79) tries to minimize the expected size.

Experiments so far show that the best strategies are Most parts and Entropy, with no distinct advantage, for all problem sizes, of any one of them over the others [Guervós et al. (2012)]. In fact, combinations of them are possible, according to Guervós et al. (2012), but have not, so far, been proved in problems where a sample of the consistent set is used. Most strategies, however,

concentrate on finding the right size for minimization of the number of turns. The complexity of the solving algorithm, however, increases quadratically with the set size, since it involves comparing every combination with the rest, which implies that the time needed to find the solution and also number of evaluations increase too fast with the space size, resulting in a bad scaling.
However, even as you can find a good solution using only a sample of the consistent set size as proved in Runarsson and Merelo (2010), Berghman et al. (2009), different set sizes to not have an influence on the outcome. When you reduce the size to the minimum it is bound to have an influence. However, in this paper we will prove that good solutions can be found by using small and, what is more, a common set size across all Mastermind problem sizes.
In the next section we will present the experiments carried out and its results for sizes from 4,8 to 6,9.

Experiments and results

The experiments presented in this paper extend those published previously, mainly by Merelo-Guervós et al. (2011). It uses the method called, simply, *Evo*. Evo, which has been released as open source code at CPAN (`http://search.cpan.org/dist/Algorithm-Mastermind/`), is an evolutionary algorithm that has been optimized for speed and to obtain the minimal number of evaluations possible. An evolutionary algorithm [Goldberg (1989), Michalewicz (1994), Eiben and Smith (2003)] is a nature-inspired search and optimization method that, modeling natural evolution and its molecular base, uses a (codified) population of solutions to find the optimal one. Candidate solutions are scored according to its closeness to the optimal solution (called *fitness*) and the whole population evolved by discarding solutions with the lowest fitness and making those with the highest fitness reproduce via combination (crossover) and random change (mutation). It is proved that these algorithms are able to find better solutions in time and, given enough time and resources, optimal solutions to several problems in engineering and even in games (such as the one presented by Fernández-Ares et al. (2011)).
Evo, which is explained extensively at Merelo-Guervós et al. (2011) searches consistent combinations until a prefixed amount of them has been found. It uses Most Parts to score consistent combinations, and the *distance to consistency* for non-consistent ones, so that the fitness directs search towards finding consistent solutions and to find solutions with better score. The solutions are quite promising, but the main problem is that the number of evaluations needed to find the solution increases rapidly with problem size (fortunately, not as fast as the problem size itself or the solution would not be convenient) and a new parameter is introduced: the optimal size of the set of consistent combinations.

Table 1: Values for the Evo parameters that obtain the best result. Permutation, crossover and mutation are *priorities*; they are normalized to 1 to convert them to rates.

Parameter	Value
Crossover	8
Mutation	1
Permutation	1
Replacement rate	0.75
Tournament size	7

In this paper we will set this size to a common for all sizes and minimal value: 10. This value has been chosen to be small enough to be convenient, but not so small that the scoring methods are rendered meaningless. This will reduce the parameters needed by one, leaving only population size to be set. The parameters in table 1 have been found to be the best in a previous study.
For every problem size, a fixed set of 5000 combinations were generated randomly, There is at most a single repetition in the smallest size, and no repetition in the rest. The set can be downloaded from `http://goo.gl/6yu16`; these sets are the same that have been used in previous papers. A single game is played for every combination.
The results for this fixed parameter setting are shown in tables 2(a) and 2(b). These tables compare previous results (with its corresponding consistent set size) and the results presented in this paper, labeled Evo10 (10 as in the size of the consistent set).
As it can be seen, results are quite similar. They are always worse, but in half the cases the difference is not statistically significant. There is a significant difference for the two smaller sizes (4,8 and 5,8), but it is not for the bigger sizes shown. However, the difference in the number of evaluations, that is, the total population evaluated to find the solution is quite significant, going from a bit less than half to a third of the total evaluations for the bigger size. This means that the time needed scales roughly in the same way, but it is even more interesting to note that it scales better for a fixed size than for the best consistent set size. Besides, in all cases the algorithm does not examine the full set of combinations, while previously the number of combinations evaluated, 6412, was almost 50% bigger than search space size for that problem.

Discussion

This paper has shown that using a small and fixed consistent set size when playing mastermind using evolutionary algorithms does not imply a deterioration of results, while cutting in half the number of evaluations

Table 2: Comparison among this approach (*Evo10*) and previous results published by ourselves (Evo) in Merelo-Guervós et al. (2011).

(a) Mean number of guesses and the standard error of the mean, the quantities in parentheses indicate population and consistent set size (in the case of the previous results).

	$\ell = 4$	$\ell = 5$		$\ell = 6$
	$\kappa = 8$	$\kappa = 8$	$\kappa = 9$	$\kappa = 9$
Evo++	(400,30) 5.15 ± 0.87	(600,40) 5.62 ± 0.84	(800,80) 5.94 ± 0.87	(1000,100) 6.479 ± 0.89
Evo10	(200) 5.209 ± 0.91	(600) 5.652 ± 0.818	(800) 6.013 ± 0.875	(800) 6.504 ± 0.871

(b) Mean number of evaluations and the standard error of the mean.

	$\ell = 4$	$\ell = 5$		$\ell = 6$
	$\kappa = 8$	$\kappa = 8$	$\kappa = 9$	$\kappa = 9$
Evo++	6412 ± 3014	14911 ± 6120	25323 ± 9972	46483 ± 17031
Evo10	2551 ± 1367	7981 ± 3511	8953 ± 3982	17562 ± 135367

needed to find them. This makes the shown configuration of the algorithm quite suitable for real-time games such as mobile apps or web games; the actual time varies from less than one second for the smallest configuration to a few seconds for the whole game in the biggest configuration shown; the time being roughly proportional to the number of evaluations, this is at least an improvement of that order; that is, the time is reduced by 2/3 for the $\kappa = 6, \ell = 9$ problem. This allows also to extend the range of feasible sizes, and gives a robust configuration that can be used throughout any mastermind problem.

The explanation for this result lies mainly in the fact that the consistent size is mainly an upper limit for the actual size the consistent sets have. This size varies as the turns proceed, as shown in Guervós et al. (2012), and go to a considerable fraction of the total search space to just a few combinations after several turns have been played. This implies that, in fact, there are just a few turns or even a single one in which the evolutionary algorithm is able to actually find a number of combinations over the set limit. At the beginning of the game there are naturally many combinations, and in fact the evolutionary algorithm only kicks in when the number of available combinations falls below the set size. However, once that happens the total size of the consistent set is smaller than the limit, which implies that, being impossible to attain that size, the algorithm stops when the number of consistent combinations does not increase for three turns.

That also explains the low number of evaluations obtained by the new algorithm. In fact, most games will be played using the combinations present in the initial population, with just a few games needing several generations to find the hidden combination. Please note that, in the case of the 6,9 problem, the average number of combinations played imply that just a few dozens of generations have actually run.

As future lines of work, we will try to reduce even more this size and try to check whether it offers good results for bigger sizes such as 7,10 or even 8,12.

Acknowledgements

This work is supported by projects TIN2011-28627-C04-02 and TIN2011-28627-C04-01 (ANYSELF), awarded by the Spanish Ministry of Science and Innovation and P08-TIC-03903 and TIC-6083 (DNEMESIS) awarded by the Andalusian Regional Government.

REFERENCES

Aldebert M. and Widenius R., 1995. *MasterMind*. Newsgroup `comp.ai.games`, archived at `https://groups.google.com/d/topic/comp.ai.games/FVOEABY4gbQ/discussion`.

Berghman L.; Goossens D.; and Leus R., 2009. *Efficient solutions for Mastermind using genetic algorithms. Computers and Operations Research*, 36, no. 6, 1880–1885.

Bestavros A. and Belal A., 1986. *Mastermind, a game of diagnosis strategies. Bulletin of the Faculty of Engineering, Alexandria University*. URL `http://citeseer.ist.psu.edu/bestavros86mastermind.html`.

Cotta C.; Merelo Guervós J.; Mora García A.; and Runarsson T., 2010. *Entropy-Driven Evolutionary Approaches to the Mastermind Problem*. In R. Schaefer; C. Cotta; J. Kolodziej; and G. Rudolph (Eds.), *Parallel Problem Solving from Nature PPSN XI*. Springer Berlin / Heidelberg, *Lecture Notes in Computer Science*, vol. 6239, 421–431. URL `http://dx.doi.org/10.1007/978-3-642-15871-1\_43`.

Eiben A.E. and Smith J.E., 2003. *Introduction to Evolutionary Computing*. Springer.

Fernández-Ares A.; Mora A.M.; Guervós J.J.M.; García-Sánchez P.; and Fernandes C.M., 2011. *Optimizing Strategy Parameters in a Game Bot*. In J. Cabestany; I. Rojas; and G.J. Caparrós (Eds.),

IWANN (2). Springer, *Lecture Notes in Computer Science*, vol. 6692. ISBN 978-3-642-21497-4, 325–332.

Goldberg D.E., 1989. *Zen and the Art of Genetic Algorithms*. In J.D. Schaffer (Ed.), *ICGA*. Morgan Kaufmann. ISBN 1-55860-066-3, 80–85.

Guervós J.J.M.; Mora A.M.; Cotta C.; and Runarsson T.P., 2012. *An experimental study of exhaustive solutions for the Mastermind puzzle*. *CoRR*, abs/1207.1315.

Irving R.W., 1978-79. *Towards an optimum Mastermind strategy*. *Journal of Recreational Mathematics*, 11, no. 2, 81–87.

Knuth D.E., 1976-77. *The Computer as Master Mind*. *J Recreational Mathematics*, 9, no. 1, 1–6.

Kooi B., 2005. *Yet another Mastermind strategy*. *ICGA Journal*, 28, no. 1, 13–20.

Merelo-Guervós J.J.; Mora A.M.; and Cotta C., 2011. *Optimizing worst-case scenario in evolutionary solutions to the MasterMind puzzle*. In *IEEE Congress on Evolutionary Computation*. IEEE. ISBN 978-1-4244-7834-7, 2669–2676.

Michalewicz Z., 1994. *Genetic Algorithms + Data Structures = Evolution programs*. Springer-Verlag, 2nd ed.

Montgomery G., 1992. *Mastermind: Improving the search*. *AI Expert*, 7, no. 4, 40–47.

Neuwirth E., 1982. *Some strategies for Mastermind*. *Zeitschrift fur Operations Research Serie B*, 26, no. 8, B257–B278.

Runarsson T.P. and Merelo J.J., 2010. *Adapting Heuristic Mastermind Strategies to Evolutionary Algorithms*. In *NICSO'10 Proceedings*. Springer-Verlag, Studies in Computational Intelligence, 255–267. Also available from ArXiV: `http://arxiv.org/abs/0912.2415v1`.

GAME DESIGN

ON THE HUMAN VISUAL PERCEPTION AND GAME DESIGN

Adriana Alvarado[1] and Benjamín Hernández[1,2]
Computer Science Department
[1]Tecnológico de Monterrey, Campus Ciudad de México
[2]Barcelona Supercomputing Center, Spain
email: alvaradogarcia.adriana@gmail.com, hbenjamin@itesm.mx

KEYWORDS

Visual perception, game design, change blindness, inattentional blindness

ABSTRACT

This presentation outlines a game design alternative based on the human visual perception. Supported on observations regarding change blindness, inattentional blindness and the relation between attention and awareness, we present some considerations in order to establish a different game design point of view. Based on human vision, we will discuss dissimilar visual styles used to depict video game's virtual environments and how these styles take into account the qualities and weaknesses of human visual perception in order to do a critical review of current game design trends.

INTRODUCTION

An important consideration during design process of virtual environments is the role of human visual perception [McNamara et al. (2011)]. How we "see" details in an image can directly impact the user's efficiency and effectiveness. Visual cognition research has been associated with video games, in order to investigate perceptual and cognitive abilities [Rebetez and Betrancourt (2007)]. Recent experiments demonstrate the positive cognitive effects in video game players, especially in improving attention, spatial cognition and mental rotation. Also recent research has shown that avid action video game players outperform non-video game players on a variety of attentional and perceptual tasks.

Even if, there has been an exploration concerning visual and cognitive effects applied to video game development, this exploration has focused on the gameplay development or the psychological effects of playing video games. Hence it is necessary to apply visual cognition focused on games visual design. Therefore our contribution is particularly focused on stating the weakness of our visual perception that causes change blindness, inatentional blindness and other kind of attention and visual failures to propose observations which can be used as a new tool in the visual design of different game genres.

PREVIOUS WORK

Human beings have a complex perceptual system that can capture and process vast amounts of complex data, nevertheless it has some surprising nuances and limitations relevant to our purpose. An initial consideration was the discovery of a limited set of visual properties that are detected very rapidly by low-level and fast-acting visual processes. These properties were initially called preattentive, since their detection seemed to precede focused attention [McNamara et al. (2011)].

Recent evidence suggests that the viewer's current state of mind can play a critical role in determining what is seen and what is not. To explain this phenomenon and relate it to our purpose, we describe three theories: change blindness, inattentional blindness, and attentional blink.

Change blindness is defined as the failure of observers to detect large, sudden changes in a display [Rensink et al. (1997)]. Cognitive tests explain how people underperform at noticing these changes on objects, photographs, and motion pictures from one instant in time to the next (dynamic phenomenon). Simons and Levin (1997) demonstrate the generality of the effect.

Inattentional Blindness refers to the failure of observers to notice some static stimuli even when these are fully visible, however the stimuli are usually unexpected (static phenomenon).

Attentional blink. Human attention is not limited only in its ability to perceive the details of a scene, as occurs in change blindness, or limited in its ability to perceive multiple objects at the same time as inattentional blindness. Attention is also severely limited in its ability to process information that arrives in quick succession, even when that information is presented at a single location in space. The term "blink" derives from the finding that when two targets are presented in rapid succession, the second of the two targets cannot be detected or identified when it appears within approximately 100 to 500 ms following the first target [Raymond et al. (1992)].

In order to explain the different phenomena previously introduced, researchers have been designed several cognitive test; for example, to detect change blindness *flicker* and the *forced choice detection* trials are per-

formed. The effects of these tests result in the next observations [Simons (2000)]:

Nothing is stored. This model argues that nothing related with the visual world is stored internally. Only information that has been abstracted from the visual perception will be retained once the image is gone. Therefore change detection requires an abstraction process to be detected.

Overwriting. This concept refers to the information that wasn't abstracted from the initial image and then replaced by new information coming from a second image, i.e. an initial image is overwritten by the following image or by the blank interval, as a result, no visual information of the initial scene remains.

First Impression. In this case, observers accurately encode the features of the initial object, but when the scene has changed observers fail to re-encode the details.

Attention and something. Our visual system's perception is limited by attention, i.e. attention contributes to reproduce or construct a description of our surroundings. According to change detection tests' results, attention is necessary to retain scene's details, if attention is focused on the changing feature, it will be abstracted. This suggests that objects in a scene, which receive attention, will be easily encoded and remembered [Rensink et al. (1997), Simons and Rensink (2005)].

VISUAL PERCEPTION AND VIDEO GAME'S VISUAL DESIGN

Base on the fact that observers can only remember visual information when compare it with an abstracted representation of the initial perceived or changed objects, in a typical video game environment, which is full of objects and details, we can claim that a considerable amount of props and details will not be remembered across a video game's sequence. Actually there is only a single change that can be perceived at the moment, although attention can be distributed in forty five items at a time [Rensink (2002)].

Taking into account the human visual perception's characteristics, we can recommend that video game design may introduce these discussed elements, for example, by reducing the complexity of the scenarios. Currently there are two main trends in video game scenery design, the realistic one, which involves full detailed, real-life based environments, and the stylistic option, which depicts fictional scenery, both are being build with a lot of props which results in an overloaded design. There is the opportunity to create an entire environment that provokes an immersion feeling without being too overloaded. Although we are aware of the relation between game narrative and their visual language; we believe it is possible to sustain an attractive story with less overloaded visuals. In order to show the differences between both visual styles (overloaded and less overloaded) we are going to compare three games which propose dissimilar designs.

The first game we introduce is Flower, designed by "That Game Company"; this game illustrates some of the ideas we have been proposing. The entire design isn't fully loaded with extra elements (figure 1), it is designed with enough elements to create an immersion into the narrative and gameplay; therefore the aim of the game and design is consistent. In the other hand there are elements inside the environment which guide the player and make him focus in the game's goal. The entire environment turns pleasant at the time the player tries to reach as many possible petals. The game combines gaze and challenge according with the players progress.

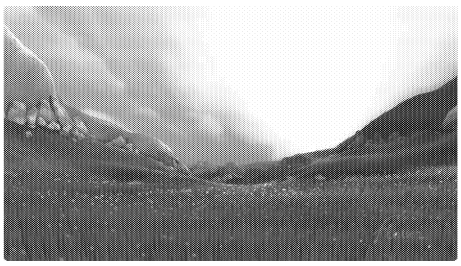


Figure 1: One of the scenes in Flower game showing a landscape and the way the player should follow a path.

The second game is called Journey, it was also designed by "That Game Company". The player is surrounded by enormous dunes and the sand is the protagonist inside the stage, that's how this particular game starts (figure 2). The interesting part of this minimal visual design is how the background turns into the shape of the environment; at the same time the color turns into space. The entire background has the necessary elements; there are not extra elements, because existing ones have their own role within the game.

The balance observed in the scene composition may sustain the ideas that we exposed before, in relation with the concepts of our human visual perception, i.e. the player has the opportunity of paying full attention to the game objectives and also the player is able to detect each minor change inside the game. Another particularity is the visual effect that is created in the background (figure 2 bottom), as if it was made of multiple layers and all together create the sensation of space inside the game. This game is a clear example of how to make the player feel immersed, without an overloaded background.

The game we are going to use as a counterexample is Gears of War 3, developed by "Epic Games". This third-person shooter game has an overloaded and a realistic visual design, however as we mentioned before it's hard to really notice all the details (figure 3) because player's attention is focused on the game's plot.

Figure 3 shows backgrounds full of detailed items which makes the game as realistic as possible in order to make

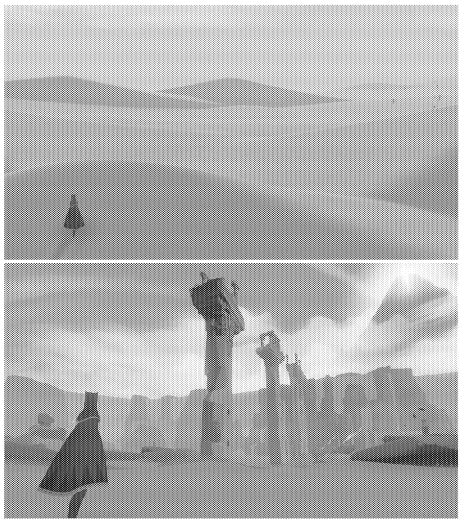


Figure 2: *Top.* One of the first stages in Journey. *Bottom.* Notice background's image was built from multiple layers and all together create the sensation of wide spaces.

a better game experience but according to the human visual perception research, players won't be able to perceive all these elements in the environment while they are playing. They may lose many of these elements, because they are mainly focused in a strategy to win.
Finally, to sustain this observation we can perform an empirical test, left as an exercise to the reader: Look at these game pictures for three seconds and answer the following questions: Which one was easier to remember?, How many details can you remember from the pictures?, Which details can you remember?; now, think about what would happen if you are watching or actually playing the real game.

CONCLUSIONS AND FUTURE WORK

We have made reference to the human visual perception weaknesses and qualities and introduced initial ideas that can be followed to design games accordingly to our visual perception behavior, i.e. having in mind its nuances and limitations, designers can create game environments only with perceived elements, allowing the player to focus on game objectives, in addition, a significant amount of resources can be saved, since all the assets required to fully generate detailed environments, which usually are expensive to purchase, time-consuming to create and make extensive use of memory and computing resources, won't be needed.
Further research will need to be carried out to sustain the hypotheses presented in this work: first, a set of test will be designed to determine change blindness, inattentional blindness and attentional blink in video games, in particular, games which follows overloaded and less overloaded visual design. Second, according to these results, we will modify the visual style of a game demo, such as one built-in the Unity Engine, and perform the same test, to establish the optimal balance between the assets needed to depict sceneries which adapt to our visual perception.

Figure 3: Different scenes from the game Gears of War 3 developed by Epic Games.

REFERENCES

McNamara A.; Mania K.; and Gutierrez D., 2011. *Perception in graphics, visualization, virtual environments and animation.* In *SIGGRAPH Asia 2011 Courses*. ACM, New York, NY, USA, 17:1–17:137.

Raymond J.E.; Shapiro K.L.; and Arnell K.M., 1992. *Temporary suppression of visual processing in an RSVP task: an attentional blink? . Journal of experimental psychology Human perception and performance*, 18, no. 3, 849–860.

Rebetez C. and Betrancourt M., 2007. *Video Game Research in cognitive and educational sciences. Cognition, Brain, Behaviour, v11 (1), p131-142.*

Rensink R.A., 2002. *Change Detection. Annual Review of Psychology*, 53, 4245–277.

Rensink R.A.; ORegan J.K.; and Clark J.J., 1997. *To See or not to See: The Need for Attention to Perceive Changes in Scenes. Psychological Science*, 8, no. 5, 368–373.

Simons D. and Levin D., 1997. *Change Blindness. Trends in Cognitive Science(1), p261-267.*

Simons D.J., 2000. *Current Approaches to Change Blindness. Visual Cognition*, 7, no. 1, 1–15.

Simons D.J. and Rensink R.A., 2005. *Change blindness: past, present, and future. Trends in cognitive sciences*, 9, no. 1, 16–20.

FACILITATING OPEN PLOT STRUCTURES IN STORY DRIVEN VIDEO GAMES USING SITUATION GENERATION

Gordon Brown
Codeplay Software Ltd.
45 York Place, Edinburgh, United Kingdom, EH1 3HP
Email: gordon@codeplay.com

David King
University of Abertay, Dundee
40 Bell Street, Dundee, United Kingdom, DD1 1HG
Email: d.king@abertay.ac.uk

KEYWORDS

Story Driven Video Games, Interactive Narrative, Emergent Plot Structures, Behaviour Trees, Player Choice, Agency.

ABSTRACT

Story driven video games are rising in popularity, along with the players desire to make meaningful choice within the plot and therefore become more involved and immersed within the experience. This paper investigates the problems which arise from implementing interactive narrative within video games and potential techniques to solve those problems. The main focus of the study was the situation generation technique, used to maintain the continuity within open, emergent plot structures, using behaviour trees as a means to implement and traverse plot sequences. The ISGEngine was developed during the course of this study in order to implement and evaluate the situation generation technique.

INTRODUCTION

Imagine you are playing a typical role playing game, you approach a village that is under attack and a soldier comes to you and asks for your help. You turn and walk away then come back one month later (in game time), the village is still under attack and the soldier still has the exact same scripted response. In this game, the graphics are incredible, the action is intense, the plot is plentiful and there is a massive, open world for you to explore. There is one thing however that keeps you from becoming completely immersed in this fantastic game world; the realisation that the decisions you make will never have any meaningful effect on the plot of the game or on the game world itself. This is the case for many video games today. Now picture the same game however this time every choice you make, no matter how insignificant, carries with it cause and effect, even if you are unaware you made the choice. Imagine if, when you chose to walk away, the village burned to the ground and this caused a chain reaction within the plot that was unpredictable and unique to you as a player.

Background

In recent years story driven video games have grown in popularity, emerging with a dominant place in the video games industry. These games stretch across various genres from first person shooters like Deus Ex: Human Revolution (Eitros Montreal 2011) to survival horrors like Silent Hill 2 (Konami 1999) and from role playing games like Fable 2 (Lionhead Studios 2008) to interactive dramas like Heavy Rain (Quantic Dream 2010). With this rise in popularity it is now becoming increasingly desirable for players to be able to interact with the plot and make the story their own. This desire is however not being met by current video games. There have been many recent attempts in games to create the illusion that the player's choices have a deep and meaningful impact on the games plot however these techniques are usually transparent and ultimately disappointing (Jubert 2010). This is where interactive narrative can be utilised. Interactive Narrative is the area of study involved in developing meaningful interactive human-computer narratives and dramas (Laurel 1991). With the graphical quality available today, the next logical step for the increasingly popular story driven video game seems to be to fully embrace interactive plot. The lack of development in the field, however, means that it is a huge risk for any developer to undertake and not many have tried to do so. There have been many projects involving interactive narrative; these have been developed primarily for research purposes, however if the techniques were to be implemented within a video game successfully it could lead the way for a whole new genre of video game.

The Element of Choice

Video games are, by their very nature, interactive experiences. This can vary greatly depending on the genre and style of game play. Alike any interactive experience, game play is one which emerges as a result of the player actively taking part and incorporating their own desires, anticipations and personal perceptions. This means that the experience is subject to each player's unique interpretation (Ermi and Mayra 2005). This makes the element of choice an extremely powerful and important factor of creating a better and more immersive game play experience (Carless 2009, Hydramyst 2012, Johnson 2008). There is however a big difference between choice and interaction, simply giving the player a choice is not enough, the player must also be given an appropriate and meaningful representation of their choice (Murray 1997). This however is no easy task as there are many problems which arise from trying to do so. Firstly there has to be a means for both creating large amounts of choice and having it meaningfully and appropriately represented. In addition to this the continuity and cohesion of the plot must be maintained throughout the game and all of the choices made by the player. Finally care must also be taken in order to constrain the requirement for new content.

INTERACTIVE NARRATIVE

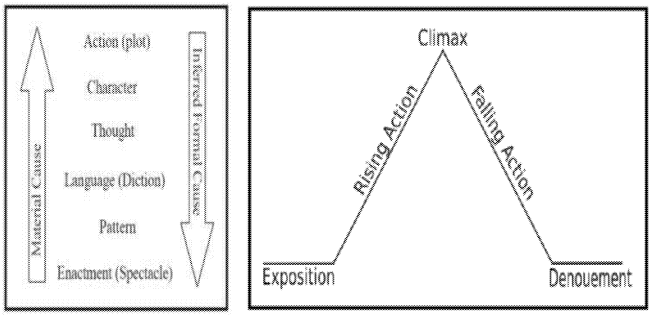
Since the 1990s interactive narrative has seen an abundance of interest from many different regions of academia involving work in artificial intelligence, computer based story-telling, and more recently, video games, however the ideology is still very young (Szilas 2003). As of yet there are no general techniques for designing and implementing an interactive narrative system. The main reasons it has proven so difficult is that interactive narrative is an oxymoron in itself as the term narrative refers to a static story predetermined by an author, while the term interaction refers to a dynamic process (Mateas 1997, Johnson 2008). However this is not to say there has not been a great deal of invaluable research and progress in the field. The most well known example is the Facade Interactive Drama (Mateas and Stern 2005), although there are many others (Aylett et al. 2005, Bangso et al. 2004, Ermi and Mayra 2005, Mateas and Stern 2005, Szilas 2003).

Murray's Aesthetics

An approach to defining the interactive narrative experience was proposed by Janet Murray; by dividing it into three aesthetic categories (Murray 1997). Immersion; which is the term used to describe the feeling of being physiologically submerged within another reality. Agency; which is the term used to describe the ability to make choices and have them represented in a meaningful and appropriate way. Transformation; which is the term used to describe the ability to experience all the different aspects of a plot.

Interactive Narrative Theory

There have been various different approaches taken when developing interactive narrative systems, these generally fall into two categories (Szilas 2002, Laurel 1991). Structuralism; which is the theory that interactive narrative can be developed by studying narrative analytically to understand the basic structures and ideas; and poetics; which is the theory that interactive narrative can be developed by studying the principles of drama and theatre. The most well known interactive narrative theory is the Aristotelian interactive drama theory; first introduced by Brenda Laurel in 1991 and generally considered the best foundation for poetics based interactive narrative systems(Laurel 1991). The theory adapts Aristotle's theories of the qualitative structure and causality (Aristotle 300BC) (Figure 1) as well as Freytag's theory of dramatic potential (Figure 2). The theory consists of material causality moving from enactment to action and formal causality moving from action to enactment (Laurel 1991).



Figures 1 & 2 : Aristotelian interactive narrative theory & Fretag's triangle

Plot Structure

One of the most important aspects of developing an interactive narrative system is the plot structure; of which there are two main forms (Johnson 2008, Murray 1997, Laurel 1991). Embedded; which involves inserting choice points into the game in order to modulate between different plot sequences; and emergent; which involves creating autonomous characters and a set of rules and allowing the plot to develop naturally from the player's interaction. A plot structure, however, can be neither fully embedded as there would be no room for interaction nor fully emergent as there would be no direction and the plot would fall apart, it must therefore be a careful balance of the two. Generally video games tend to utilise a more embedded approach as this is easier to maintain.

Agency

Many feel that agency is the most important aspect of an interactive narrative system (Pearce 1997, Ermi and Mayra 2005). True agency is achieved when the player's choices have a meaningful impact on the plot or on the game world itself (Murray 1997). Agency can be broken down in local agency; low level interactions, and global agency; high level plot direction (Mateas and Stern 2005).

Dramatic Potential

When developing an interactive narrative system it is extremely important to maintain the dramatic potential. Agency expands the potential, therefore there must be constraints in order to control it and ultimately bring it to it a conclusion (Laurel 1991). The dramatic potential can be portrayed with Aristotle's theory of dramatic probability, expressed in the form of a flying wedge. Moving through time the range of possible outcomes starts off completely open and narrows down the probability to one or more specific outcomes (Figure 3).

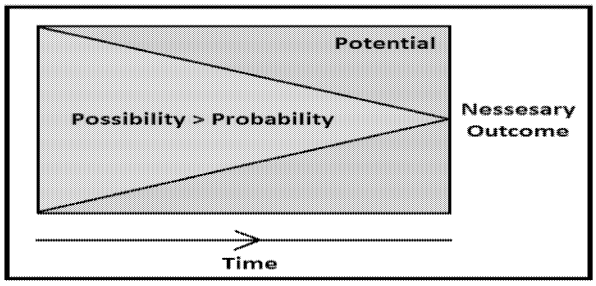


Figure 3: Aristotelian dramatic probability theory

Choice must therefore be constrained, but not in any way which would disrupt the immersion or transformation of the experience (Murray 1997, Ward 2004). Constraints can be anything from a suggestion to a rule and do not have to be invasive to the plot. They can be introduced in the form of natural events, situations or coincidences (Laurel 1991). In order to maintain the dramatic potential it is common to have a director or drama manager, however this can be problematic as if the director forces an inappropriate action within the plot the immersion can be broken (Mateas 1997, Mateas and Stern 2005). It would be great if a video game could give the player complete free will; however this is impossible as no designer could anticipate every action a player may wish to perform. Therefore this free will is more

of an illusion, a balance of agency and constraints which allow the player to perform all the actions which are appropriate for the current simulation. Therefore the aim is not to limit what the player can do but limit what the player thinks of doing (Laurel 1991).

METHODOLOGY

The motivation behind this study was to develop interactive narrative within video games in order to create game play experiences where the player can make meaningful choices, take control of the direction of the plot and even make it unique to them. The first major problem with this idea is that generally video games follow very strict embedded plot structures where the player is unable to deviate from the authored story. Therefore in order to make this possible, techniques must be developed which allow video games to incorporate an emergent plot structure and a strong element of agency but also solves the problems discussed earlier. The main focus of the study was the situation generation technique. In order to evaluate this fully, an interactive narrative engine was designed and implemented; the ISGEngine, and a short demo was created.

Interactive Story-Game Theory

Before developing an interactive narrative system it was crucial to have a strong theoretical foundation. The interactive story-game theory (Figure 4) was derived from the Aristotelian interactive drama theory (Laurel 1991). The theory was developed in order to aid the study and further the development of the ISGEngine. The theory was designed towards story driven video games and incorporates an equal representation of both story and game elements as well as a balance of agency and constraints.

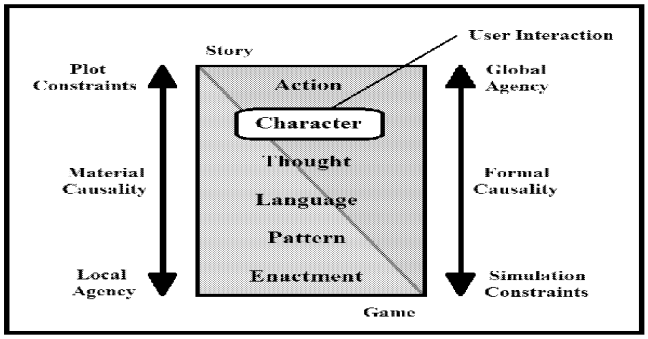


Figure 4: Interactive Story-Game Theory

In this theory both the formal causality and material causality have been broken down into two separate components; global agency and simulation constraints and local agency and plot constraints respectively. Each element in the qualitative structure remains the same and the user interaction is placed within the character element, however now each element is both the global agency and the plot constraint of the element below and the local agency and the simulation constraint of the element above. Each line of causality works in parallel as the player progresses through the game and time passes, affecting both the story and the game. The four main components (global agency, local agency, simulation constraints and plot constraints) all must be balanced in order to achieve appropriate and meaningful agency and maintain an impressive experience. This model is used in two ways, firstly as implemented functionality of the ISGEngine and secondly as a set of heuristics to guide the authoring process. As an example imagine a game where the player is in a gun dual in the wild west. The plot constitutes the background, the characters, the setting, the time period and much more. The game constitutes an action game where the aim is to avoid dying and attempt to kill the opponent. In this example the four elements of the interactive story-game theory can be used as follows. Local agency could be used to allow the player to aim freely at their opponent and choose when to shoot. Global agency could be used to allow the player to choose whether or not they shoot at all, and maybe instead find a different solution to the confrontation. Plot constraints could prevent the player from being able to use an unrealistic weapon as this would break the immersion of the story. Simulation constraints could prevent the player from being able to kill themselves as this would break the rules of the game.

ISGEngine

The ISGEngine is a basic story game engine containing plot incidents and sequences, characters, behaviours, quests, items and a dialogue system. The most important aspects are: plot sequencing, incident conditioning and situation generation. The aim of the ISGEngine was to solve the main problems with implementing interactive narrative in video games. It permits the player to be given choice in a wide range of ways including siding with a character during a conflict, where to go and who to speak to, which quests to accept and complete, the manner and content of dialogue and how quickly a quest is performed. It lets the player experience a meaningful response to their choices through character behaviour, content of character dialogue towards player, manner of character dialogue towards player and the availability of quests. It allows the continuity of the plot to be maintained through situation generation, and plot cohesion to be maintained through plot incident conditioning. It also allows control over the requirement for new content by placing the majority of the agency locally (dialogue and quests) and having the global agency (larger plot direction) guided by the local agency.

Creating Choice

Creating choice for the player is relatively easy, as they do not even need to know they are making it. It can be anything from how they interact with another character to the fact that they forget to finish a quest on time. Choices that the player does not realise they are making are the most effective as it means the player is unaware of what they did, therefore increasing the re-playability. In the ISGEngine the choice is mainly the completion or failure of a quest, the content and manner of the dialogue with other characters and the order of events encountered, throughout the game world.

Characters

The characters in the ISGEngine, have a persona; a set of values corresponding to their opinion of the player. They also have a basic behaviour system involving moving around the game world.

Dialogue System

The dialogue system is very similar to that of any role playing game, although it is a vital component of the ISGEngine. It allows the player to interact with the other

characters in relation to topics, items and quests and also to choose the manner in which they do.

Plot Incidents

Plot incidents are the main component of the ISGEngine, they are used to track the choices unique progress of the player throughout the plot. Incidents can be anything from a character performing a specific dialogue or behaviour to a quest being completed or failed. Incident conditioning involves using pre-/post-conditions to connect specific incidents with changes in character persona and behaviour, the availability of quests and dialogue topics and character responses to dialogue. Much of this is forced to allow for strong cohesion in any authored plot, for example the player cannot ask a character about a topic, item or quest they do not yet know about. By creating an open emergent plot structure of local agency incidents guided towards more dramatically relevant global agency incidents, the ISGEngine aims to allow the player the freedom to choose their own path, whilst at the same time guide them on a dramatic and interesting storyline (Figure 5).

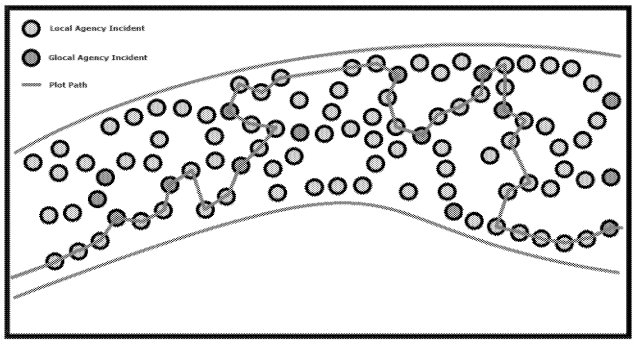


Figure 5: ISGEngine Plot Structure

Situation Generation

The ISGEngine was designed to allow for open emergent plot structures with a large amount of agency for the player whilst being non-disruptive to the experience. Most interactive narrative models utilise a director of some kind which monitors the player's choices and selects an appropriate plot sequence for them to follow, however this does not always guarantee agency and can be disruptive to the immersion and transformation of the experience. Situation generation allows the opposite of this. A web of plot incidents and the required content are created and the player can play the game, moving through the plot however they feel. Instead of forcing the player to follow a specific plot the director will merely guarantee the availability of the plot sequences and provide gentle guidance. This means that the player can feel in control of their decisions and the plot they are following, while safe in the knowledge there will always be an ultimate conclusion. The way this is implemented is by using behaviour trees to structure plot sequences that the author intends to be followed based upon a final incident. These trees can then be traversed in order to calculate the current continuity and progress (Figure 6). The reason behaviour trees were chosen as opposed to any other technique is that they are very simple and quick to implement, fast to traverse and are made up of AND and OR nodes, which is ideal for the incident conditioning required (Millington and Funge 2009). By measuring the continuity and progress of the different plot sequences the director can obtain a detailed analysis of the player's unique plot path and can aid and guide the player further. For example if the continuity of a specific plot sequence is getting low, meaning that either the player is losing interest or cannot find their way, the director can activate special plot revival incidents which can invoke a situation such as a chance meeting with another character which can introduce a new quest or dialogue topic to get the player back on track, if they so wish. The only real downfall of this technique is that if a player sets out to break the continuity on purpose then eventually they will, meaning that a generic, all be it less dramatic or interesting plot sequence, that is always possible, must exist. It is assumed however that most players will not be interested in doing this.

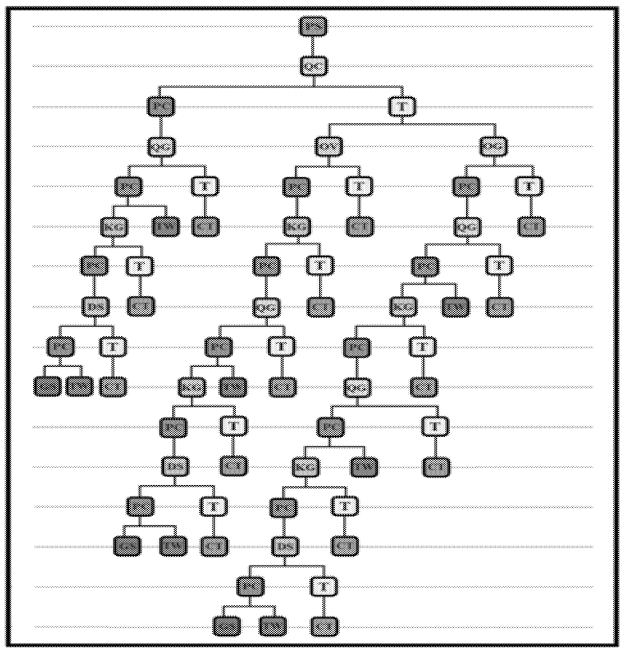


Figure 6: Plot Sequence Tree

EVALUATION

In order to evaluate this technique a 20 minute demo was created and both metric and survey data were collected from 20 user testers (Figure 7).

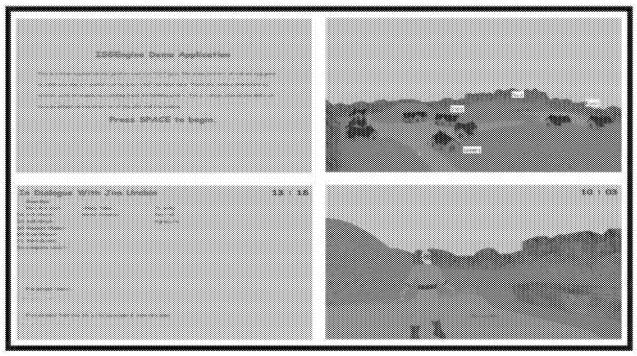


Figure 7: Demo Screen Shots

The Demo

In the demo the player was a traveller visiting a town for the first time. There were 7 other characters and various simple quests to perform, in the first half of the demo the quests all involved local agency, however towards the end there was a

larger global agency plot sequence involving 3 of the characters. This plot sequence consisted of one character attempting to rob a tavern and depending on the degree of help the player gave to this person and at what point the player alerted the guard if they did, various different endings could occur including the tavern owner dying, that character being arrested, that character fleeing and the player being arrested. Each player's play through the game was categorised under one of these endings using a simple algorithm which evaluated the incidents which had been triggered. In order to evaluate the situation generation technique one of the plot sequences was designed so that it was relatively easy to break the continuity. If the player follows a certain plot sequence the player is given the opportunity to help the character attempting to rob the tavern, however if the character misses the opportunity the situation generation creates a chance meeting with the character to give the player another possibility to complete the quest.

Results

The metric data showed that the plot paths that each player took varied greatly even when the same ending was encountered. In the diagram below, the coloured lines represent the paths of three user testers who all encountered the same ending (Figure 8).

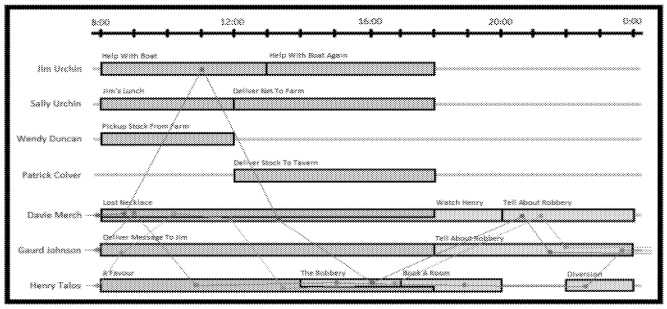


Figure 8: User Test Plot Paths

CONCLUSION

This study has discovered that creating emergent open plot structures is a very difficult task, although it is possible. With techniques such as situation generation such a plot structure can be implemented and its continuity maintained. As future work it will interesting to see how the ISGEngine performed with a much larger plot, as this would involve very large plot trees and a level of detail technique would have to be implemented in order to maintain performance. One addition which would greatly improve the ISGEngine would be to introduce randomly generated game content, alleviating the need for authoring, a feature which is being investigated in current video games (Bathesda Studios 2011). Interactive narrative is a rapidly growing area in the video games industry and there will definitely be a place on the shelves for story driven video games, however there is still much work to be done.

REFERENCES

Aristotle, 330BC. *La Poetique (The Poetics).* London: Penguin Group.

Aylett, R. S., Louchart, S., Dias, J., Paiva, A. and Vala, M. 2005. *FearNot! Intelligent Virtual Agents*, vol. 3661, chap. 26.

Bangso, O., Jensen, O. G., Jensen, F. V., Anderson, P. B. and Kocka, T. 2004. *Non-Linear Interactive Storytelling Using Object-Oriented Bayesian Networks*. [article]. Available From: http://vbn.aau.dk/files/132449/nolist.pdf.

Carless, S. 2009. *In-Depth: Peter Molyneux On The Importance Of Choice.*[online] Available From: http://www.gamasutra.com/php-bin/news_index.php?story=24924.

Eitros Montreal. 2011. *Deus Ex: Human Revolution*. [disk]. Windows, Mac OS. Xbox 360. Play Station 3.Square Enix.

Ermi, L. and Mayra, F. 2005 *Fundamental Components of the Gameplay Experience Analysing Immersion.* DIGRA.

Hydramyst. 2012. *Importance of Choice in Video Games.* [online] Available From: http://hydramist.tv/

Johnson, K. 2008. *Lost Cause: An Interactive Movie Project*. [MSc]. Available From : http://www.motionpieces.ca/attachments/thesis.pdf

Jubert, T. 2010. *Plot is Gameplays Bitch*. [online]. Available From: http://tom-jubert.blogspot.com/2010/05/heavy-rain-decision-making-in-games.html.

Konami. 1999. *Silent Hill*. [disk]. Play Station. Konami Digital Entertainment.

Lionhead Studios. 2008. *Fable 2*. [disk]. Xbox 360. Microsoft Game Studios.

Laurel, B. 1991. *Computers as Theatre*. Boston, MA: Addison-Wesley Professional.

Mateas, M. 1997. *An Oz-Centric Review of Interactive Drama and Believable Agents*. Computer Science 1600.

Mateas, M and Stern, A. 2005. *Structuring Content in the Facade Interactive Drama Architecture*. [article]. Available From: http://www.interactivestory.net/papers/MateasSternAIIDE05.pdf.

Millington, I. and Funge, J. 2009. *Artificial Intelligence for Games.* 2nd ed. Burlington, MA: Morgan and Kaufman.

Murray, J. H. 1997. *Hamlet on the Holodeck*. Cambridge, MA: The MIT Press.

Pearce, C. 1997. *The Interactive Book*. Indianapolis, IN: Macmillan Technical Publishing.

Quantic Dream. 2010. *Heavy Rain*. [disk]. Play Station 3. Sony Computer Entertainment.

Szilas, N. 2002. *Structural Models for Interactive Drama.* 2nd International Conference on Computational Semiotics for Games and New Media

Szilas, N. 2003. *IDtension: A Narrative Engine for Interactive Narrative.* Technologies for Interactive Digital Storytelling and Entertainment TIDSE'03 24-26

Ward, J. A. 2004. Interactive Narrative: Theory and Practice. [BSc]. Available From : www.fuzzybinary.com/ thesis/INThesis-JW.doc.

BIOGRAPHY

GORDON BROWN recently graduated from the University of Abertay, Dundee with a first class honours degree in Computer Games Technology. He is now currently working on compiler technology and GPU acceleration, at Codeplay Software Ltd in Edinburgh, Scotland.

ADAPTIVE GAMES

IMPROVING SOFTWARE QUALITY THROUGH DESIGN PATTERNS: A CASE STUDY OF ADAPTIVE GAMES AND AUTO DYNAMIC DIFFICULTY

Muhammad Iftekher Chowdhury and Michael Katchabaw
Department of Computer Science
University of Western Ontario
London, Ontario,
Canada
E-mail: {mchowd2, katchab}@uwo.ca

KEYWORDS
Auto dynamic difficulty, game balancing, software design patterns.

ABSTRACT

Auto dynamic difficulty (ADD) is the technique of automatically changing the level of difficulty of a video game in real time to match player expertise. Recreating an ADD system on a game-by-game basis is both expensive and time consuming, ultimately limiting its usefulness. Thus, we leverage the benefits of software design patterns to construct an ADD framework. In this paper, we discuss a number of desirable software quality attributes that can be achieved through the usage of these design patterns, based on a case study of two video games.

INTRODUCTION

In the last 30 years, the scope of video games has expanded considerably in terms of platforms, genres and size. Unfortunately, we still struggle with keeping players engaged in a game for a long period of time. According to a recent article (Snow 2011), 90% of game players never finish a game. One of the key engagement factors for a video game is an appropriate level of difficulty, as games become frustrating when they are too hard and boring when they are too easy (Hao et al. 2010). From the point of view of skill levels, reflex speeds, hand-eye coordination, tolerance for frustration, and motivations, video game players may vary drastically (Bailey and Katchabaw 2005). These factors together make it very challenging for video game designers to set an appropriate level of difficulty in a video game. Traditional static difficulty levels (e.g., easy, medium, hard) often fail in this context as they expect the players to judge their ability themselves appropriately before playing the game and also try to classify them in broad clusters (e.g., what if easy is too easy and medium is too difficult for a particular player?).

Auto dynamic difficulty (ADD), also known as dynamic difficulty adjustment (DDA) or dynamic game balancing (DGB), refers to the technique of automatically changing the level of difficulty of a video game in real time, based on the player's ability (or, the effort s/he is currently spending) in order to provide them with an "optimal experience", also sometimes referred to as "flow". If the dynamically adjusted difficulty level of a video game appropriately matches the expertise of the current player, then it will not only attract players of varying demographics but also enable the same player to play the game repeatedly without being bored. Popular games such as "Max Payne", "Half-Life 2" and "God Hand" use the concept of auto dynamic difficulty. While others have studied ADD in games, this has been done in an ad hoc fashion in terms of software design and is therefore not reusable or applicable to other games. Recreating an ADD system on a game-by-game basis is both expensive and time consuming, ultimately limiting its usefulness. For this reason, we leverage the benefits of software design patterns (Gamma et al. 1995) to construct an ADD framework and system that is reusable, portable, flexible, and maintainable.

In (Chowdhury and Katchabaw 2012), we introduced a collection of four design patterns originally from self-adaptive system literature (Ramirez and Cheng 2010), derived in the context of enabling auto dynamic difficulty in video games. Unfortunately, to date, the literature on the usage of software design patterns in developing video games is relatively scarce. Work in this area is mostly limited to using video games as a means for teaching software design patterns in undergraduate computer science courses (e.g., Gestwicki and Sun 2008; Antonio et al. 2009). Very little, if any, motivation of using software design patterns for implementing ADD is found in the video game literature. Thus, in this paper, we discuss the improvements to overall software quality that can be achieved through the usage of these design patterns, based on empirical evidence acquired through a case study involving implementation and source code analysis of two proof-of-concept video games.

The rest of this paper is organized as follows. In the next section, we overview key literature from the area. We then describe our design patterns for enabling auto dynamic difficulty in video games, as well as our case study. Finally, in the remaining sections, we present the results from our case study and conclude the paper.

RELATED WORK

Considering the variety of contexts and the focus of related research, we divide our related work discussion into three sub-sections. First we highlight the research that explores the use of ADD in video games. Afterwards, we discuss the literature on using software design patterns in video games. Finally, we discuss the research gap and put our work in the context of this other work.

Auto Dynamic Difficulty

In recent years, ADD has received notable attention from numerous researchers. Some of this research is primarily focused on knowledge seeking, whereas other works present solutions such as frameworks and algorithms. Additionally, in some research, new solutions are presented together with empirical validations. Here, we review some of these works.

(Bailey and Katchabaw 2005) developed an experimental testbed based on Epic's Unreal engine that can be used to implement and study ADD in games. It allows development of new ADD algorithms as well. A number of mini-game gameplay scenarios were developed in the test-bed and these were used in preliminary validation experiments.

(Rani et al. 2005) suggested a method to use real time feedback, by measuring the anxiety level of the player using wearable biofeedback sensors, to modify game difficulty. They conducted an experiment on a Pong-like game to show that physiological feedback based difficulty levels were more effective than performance feedback to provide an appropriate level of challenge. Physiological signals data were collected from 15 participants each spending 6 hours in cognitive tasks (i.e., anagram and Pong tasks) and these were analyzed offline to train the system.

(Hunicke 2005) used a probabilistic model to design ADD in an experimental first person shooter (FPS) game based on the Half-life SDK. They used the game in an experiment on 20 subjects and found that ADD increased the player's performance (i.e., the mean number of deaths decreased from 6.4 to 4 in the first 15 minutes of play) and the players did not notice the adjustments.

(Orvis et al. 2008), from an experiment involving 26 participants, found that across all difficulty levels, completion of the game resulted in an improvement in performance and motivation. Prior gaming experience was found to be an important influence factor. Their findings suggested that for inexperienced gamers, the method of manipulating difficulty level would influence performance.

(Hao et al. 2010) proposed a Monte-Carlo Tree Search (MCTS) based algorithm for ADD to generate intelligence of non player characters. Because of the computational intensiveness of the approach, they also provided an alternative based on artificial neural networks (ANN) created from the MCTS. They also tested the feasibility of their approach using Pac-Man.

(Hocine and Gouaïch 2011) described an ADD approach for pointing tasks in therapeutic games. They introduced a motivation model based on job satisfaction and activation theory to adapt the task difficulty. They also conducted preliminary validation through a control experiment on eight healthy participants using a Wii balance board game.

Software Design Patterns in Video Games

In a number of works, video games have been proposed as a tool to teach software engineering in general and design patterns in particular. On the other hand, unfortunately, work focusing on how game developers can benefit from the usage of software design patterns is relatively rare. Here we discuss examples of both types of research.

(Gestwicki and Sun 2008) presented a video game based approach to teach software design patterns to computer science students. They developed an arcade style game, EEClone, which consists of six key design patterns and then used these patterns in their case study. Student participants analyzed the game to learn the usage of those patterns.

(Antonio et al. 2009) described their experience in teaching software design patterns using a number of incremental abstract strategy game design assignments. In their approach, each assignment was completed by refactoring and using design patterns on previous assignments.

(Narsoo et al. 2009) described the usage of software design patterns to implement a single player Sudoku game for the J2ME platform. They found that through the use of design patterns, new requirements could be accommodated by making changes to fewer classes than otherwise possible.

Research Gap

As we can see from above discussion, the work on ADD in video games focuses on tool building (e.g., framework (Bailey and Katchabaw 2005), algorithm (Hunicke 2005; Hao et al. 2010) etc.) and empirical studies (e.g., Rani et al. 2005; Orvis et al. 2008 etc.), but they all use an ad-hoc approach from a software design point view. On the other hand, research on using software design patterns in video games is mostly limited to using video games as a means for teaching design patterns in undergraduate computer science courses (e.g., Gestwicki and Sun 2008; Antonio et al. 2009). In contrast, much work has been done towards game design patterns, such as the foundational work of (Björk and Holopainen 2004) and many others, but the focus there is game design and not software design, which is a subtle, yet important distinction. Thus, in this paper, we discuss the software quality attributes that can be achieved through the usage of software design patterns in the context of ADD, based on an empirical study.

DESIGN PATTERNS

In this section, we briefly discuss the four software design patterns for enabling ADD in video games. For further details, the reader is encouraged to refer to (Chowdhury and Katchabaw 2012) for elaborated discussion and examples.

Sensor Factory

The sensor factory pattern is used to provide a systematic way of collecting data while satisfying resource constraints, and provide those data to the rest of the ADD system. *Sensor* (please see Figure 1) is an abstract class that encapsulates the periodical collection and notification mechanism. A concrete sensor realizes the Sensor and defines specific data collection and calculation. The *SensorFactory* class uses the "factory method" pattern to

provide a unified way of creating any sensors. It takes the *sensorName* and the *object* to be monitored as input and creates the sensor. Before creating a sensor, the *SensorFactory* checks in the *Registry* data structure to see whether the sensor has already been created. If created, the *SensorFactory* just returns that sensor instead of creating a new one. Otherwise, it verifies with a *ResourceManager* whether a new sensor can be created without violating any resource constraints.

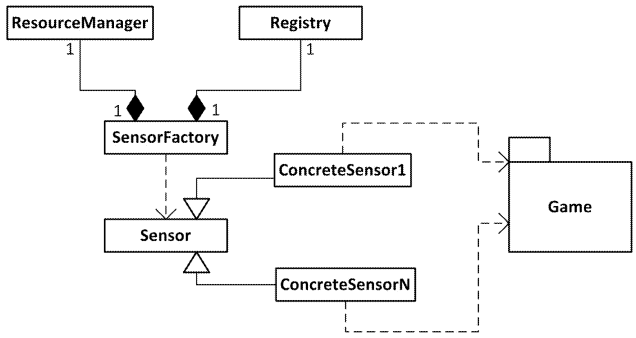


Figure 1: Sensor Factory Design Pattern

Adaptation Detector

With the help of the sensor factory pattern, the *AdaptationDetector* (please see Figure 2) deploys a number of sensors in the game and attaches observers to each sensor. *Observer* encapsulates the data collected from sensor, the unit of data (i.e., the degree of precision necessary for each particular type of sensor data), and whether the data is up-to-date or not. *AdaptationDetector* periodically compares the updated values found from *Observer*s with specific *Threshold* values with the help of the *ThresholdAnalyzer*. Each *Threshold* contains one or more boundary values as well as the type of the boundary (e.g., less than, greater than, not equal to, etc.). Once the *ThresholdAnalyzer* indicates a situation when adaptation might be needed, the *AdaptationDetector* creates a *Trigger* with the information that the rest of the ADD process needs.

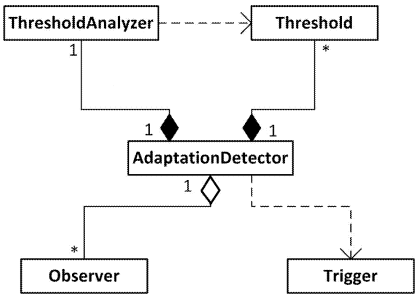


Figure 2: Adaptation Detector Design Pattern

Case Based Reasoning

While the adaptation detector determines the situation when a difficulty adjustment is required by creating a *Trigger*, case based reasoning (please see Figure 3) formulates the *Decision* that contains the adjustment plan. The *InferenceEngine* has two data structures: the *TriggerPool* and the *FixedRules*. *FixedRules* contains a number of *Rules*. Each *Rule* is a combination of a *Trigger* and a *Decision*. The *Trigger*s created by the adaptation detector will be stored in the *TriggerPool*. To address the triggers in the sequence they were raised in, the *TriggerPool* should be a FIFO data structure. The *FixedRules* data structure should support search functionality so that when the *InferenceEngine* takes a *Trigger* from the *TriggerPool*, it can scan through the *Rule*s held by *FixedRules* and find a *Decision* that appropriately responds to the *Trigger*.

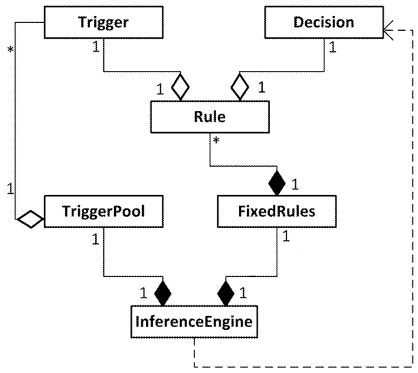


Figure 3: Case Based Reasoning Design Pattern

Game Reconfiguration

Once the ADD system detects that a difficulty adjustment is necessary, and decides what and how to adjust the various game components, it is the task of the game reconfiguration pattern to facilitate smooth execution of the decision. The *AdaptationDriver* receives a *Decision* selected by the *InferenceEngine* (please see case based reasoning in previous section) and executes it with the help of the *Driver*. *Driver* implements the algorithm to make any attribute change in an object that implements the *State* interface (i.e., that the object can be in ACTIVE, BEING_ACTIVE, BEING_INACTIVE or INACTIVE states, and outside objects can request state changes). As the name suggests, in the active state, the object shows its usual behavior whereas in the inactive state, the object stops its regular tasks and is open to changes. The *Driver* takes the object to be reconfigured (default object used if not specified), the attribute path (i.e., the attribute that needs to be changed, specified according to a predefined protocol such as object oriented dot notation) and the changed attribute value as inputs. The *Driver* requests the object that needs to be reconfigured to be inactive and waits for the inactivation. When the object becomes inactive, it reconfigures the object as specified. After that, it requests the object to be active and informs the *AdaptationDriver* when the object becomes active. The *GameState* maintains a *RequestBuffer* data structure to temporarily store the inputs received during the inactive state of the game. (If the reconfiguration is done efficiently, however, it should be completed within a single tick of the main game loop, and this buffering should be largely unnecessary.) The *GameState* overrides Game's event handling methods and game loop to implement the *State* interface.

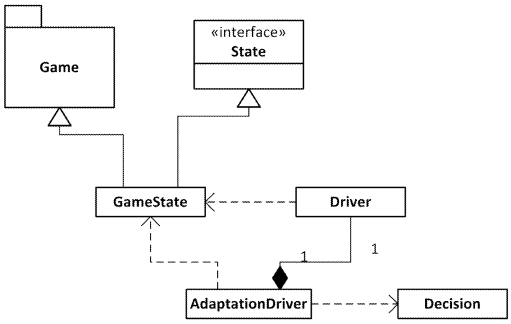


Figure 4: Game Reconfiguration Design Pattern

Integration of ADD Design Patterns

In this Section, we briefly re-discuss how the four design patterns discussed in previous sub-sections work together to create a complete ADD system (please see Figure 5). The sensor factory pattern uses *Sensor*s to collect data from the game so that the player's perceived level of difficulty can be measured. The adaptation detector pattern observes *Sensor* data using *Observer*s. When the adaptation detector finds situations where difficulty needs to be adjusted, it creates *Trigger*s with appropriate additional information. Case based reasoning gets notified about required adjustments by means of *Trigger*s. It finds appropriate *Decision*s associated with the *Trigger*s and passes them to the adaptation driver. The adaptation driver applies the changes specified by each *Decision* to the game, to adjust the difficulty of the game appropriately, with the help of the *Driver*. The adaptation driver also makes sure that the change process is transparent to the player. In this way, all four design patterns work together to create a complete ADD system for a particular game.

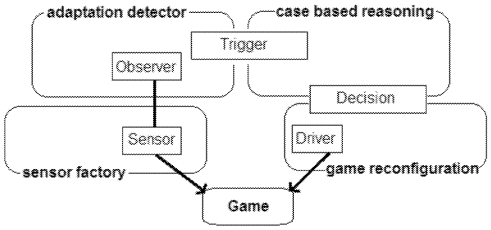


Figure 5: Four Design Patterns Working Together in a Game

CASE STUDY

In this section, we describe the case study used to assess software quality improvements achieved through our design patterns for ADD. We begin with a brief description of each of the two games that were used in the case study. We then describe the case study methods and the quality metrics that were collected from the case study.

Case Study Games

We used two arcade style single player games developed in Java for the case study. The first game is a variant of Pac-Man and will be referred to as Game-P from here onwards. Game-P was developed for the purposes of this research. The level structure and gameplay of the second game is similar to the popular Super Mario game series and will be referred to as Game-S from here onwards. Game-S is a slightly modified version of a platform game described in (Brackeen et al. 2004). In sub-sections below, we briefly describe the game logic and ADD logic of these two games.

Game-P

In this game, the player controls Pac-Man in a maze (please see Figure 6). There are pellets, power pellets, and 4 ghosts in the maze. Pac-Man has 6 lives. Usually, ghosts are in a predator mode and touching them will cause the loss of one of Pac-Man's lives. When Pac-Man eats a power-pellet, it becomes the predator for a certain amount of time. When Pac-Man is in this predator mode and eats a ghost, the ghost will go back to the center of the maze and will stay there for a certain amount of time. Eating pellets gives points to Pac-Man. The player tries to eat all the pellets in the maze without losing all of Pac-Man's lives. The player is motivated to chase the ghosts while in predator mode, as that will help them by removing the ghosts from the maze for a time, allowing Pac-Man to eat pellets more freely. Ghosts only change direction when they reach intersections in the maze, while Pac-Man can change direction at any time. A ghost's vision is limited to a certain number of cells in the maze. Ghosts chase the player if they can see them. If the ghosts do not see Pac-Man, they try to roam the cells with pellets, as Pac-Man needs to eventually visit those areas to collect the pellets. If the ghosts do not see either Pac-Man or pellets, they move in a random fashion.

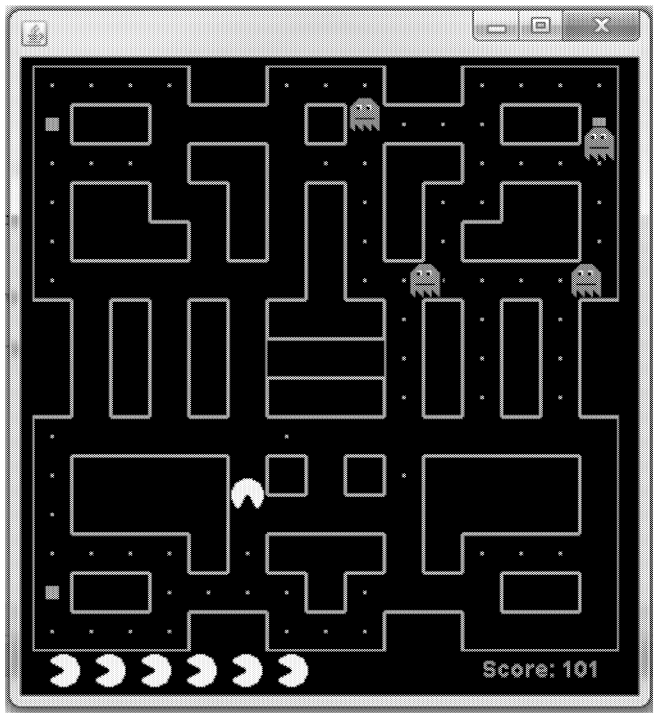


Figure 6: Screen Captured from Game-P

Usually, a Pac-Man game is multi-level, but our implementation (i.e., Game-P) has only one level. The maximum possible score is 300 in our case, so the player will try to achieve the score of 300 without losing all of Pac-Man's lives. Our assumption is that if the player loses all lives (i.e., 6) before finishing the game, then the average score per life (i.e., total score / number of lives lost to achieve the score) would be less than 50 and the game would seem overly difficult to them. On the other hand, if the player finishes the game losing half of the lives or less, then the average score would be greater than or equal to 100, and the game would seem too easy to them. Thus, in this case, the ADD system monitors the average-score-per-life and changes game difficulty accordingly. It starts increasing the game difficulty when the monitored value is more than 50 and the game become most difficult when the value is more than 100. (Corresponding logic decreases the game difficulty when the average-score-per-life is less than 50.) The attributes of ghost speed, ghost vision length, duration of Pac-Man's predator mode, and the amount of time that a ghost stays in the centre of the maze after being eaten by Pac-Man in predator mode are increased or

decreased to change the game difficulty. Each of these attributes has lower and upper limits, so that the game includes the option of someone playing extremely well or extremely poorly.

Game-S

In this game, the player controls the player character in a platform world (please see Figure 7). There are three levels, each having different tile based maps. There are power ups and non-player characters (i.e., enemies) in each level. There are three different types of power ups: basic power ups, bonus power ups, and a goal power up. Basic power ups and bonus power ups give certain points to the player. In each level there is one goal power up that can be found at the end of the level. The goal power up takes the player from one level to another. There are two different types of non-player characters: ants and flies. Ants and flies move in one direction and change direction when blocked by the platforms. The player character can run on and jump from platforms. When the player character jumps on (i.e., collides from above) non-player characters, the non-player character dies. If the player character collides with a non-player character in any other direction, then the player character dies instead. The player character has 6 lives. When the player character dies, it loses one life and the game restarts from the beginning of that level. The player character and ants are affected by gravity; flies are only affected by gravity when they die.

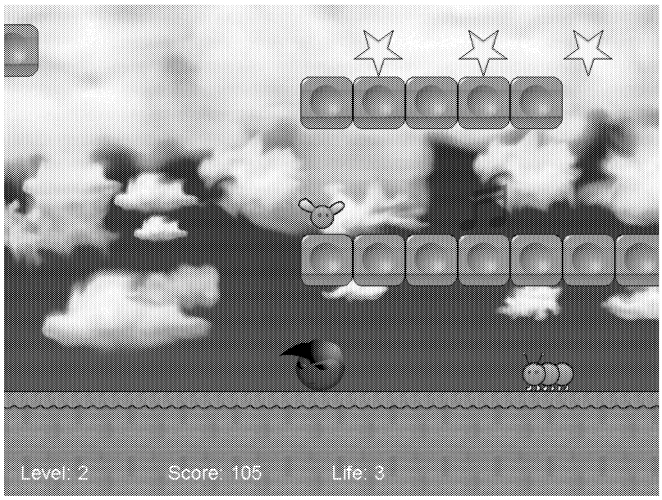


Figure 7: Screen Captured from Game-S

In this game, three map variants were created for each level. For a particular level, the same objects were placed in the map, but positioned slightly differently. One map variant was the default version and other two were easier and harder versions of the default map. The ADD system monitors score-per-level and life-lost-per-level, and adapts difficulty accordingly. One possible adaptation is the modification of the speed of the non-player characters and takes place during the game. Another adaptation is a change in level structure (i.e., loading a different version of the map) and takes place when the player character goes to the next level or in the next loading of the same level (i.e., when the player character dies). The modifications are minor and assumed to be transparent to the player, but altogether alter the game difficulty. Apart from this, each level is more difficult and lengthier than the previous level, but has more points to give the player a sense of progress and accomplishment. Similar to Game-P, modifications in Game-S have lower and upper limits, so that the game includes the option of someone playing extremely well or extremely poorly.

Case Study Method

Here we briefly discuss the steps that were taken during the course of the case study. Firstly, Game-P was developed without our pattern-based ADD system. Our ADD system was then developed and integrated with Game-P. The source code for the ADD system was then refactored within the scope of the design patterns. We manually tested Game-P separately and with the ADD system. A player simulation (i.e., a simple artificial intelligence playing the game itself using heuristic functions) was also created to test the game. Game-S was then chosen for study as it used the same Java platform as Game-P, but substantially differed in terms of gameplay, and was freely available and well documented in a book (Brackeen et al. 2004). Two default maps accompanied Game-S originally. We created one more default map ourselves, as well as two different variants of each map, as discussed earlier. There was no scoring mechanism in Game-S as originally written, so we developed scoring logic ourselves. After this, we took the source code of our ADD system used with Game-P and extended its abstract base classes (*Sensor*, and so on) to adapt the system for Game-S. We manually tested Game-S separately and with the ADD system. We then analyzed and compared the source code of the ADD systems of Game-P and Game-S to assess software quality according to a few key software metrics, as discussed in the next sub-section.

Analysis Tool and Metric

During the development of the ADD system for Game-P, we realized that much of its source code would be reusable across various games. During the extension of the ADD system for Game-S, we did not need to make modifications to many of the classes from the system for Game-P. To assess this quantitatively, we selected a metric and a tool. As a metric, we used Source Lines of Code (SLOC), as it is a widely accepted software metric and helps in estimating the development effort of a software product. For a tool, we used Unified Code Count (UCC) developed by the University of Southern California Center for Systems and Software Engineering. Features of UCC include both counting SLOC and comparing two versions of source code. UCC counts both the logical and physical SLOC. As seen from the two examples in Table 1, since logical SLOC disregards code formatting, it is more representative of the size of the software, and so we used logical SLOC as our metric in this study.

Table 1: Difference Between Physical and Logical SLOC

Example	Physical vs. Logical SLOC
if(a = = 0) foo();	Physical SLOC = 1,Logical SLOC = 2

if(a = = 0)
{
foo();
}
	Physical SLOC = 4,Logical SLOC = 2

SOFTWARE QUALITY ASPECTS

In this section, we describe how different desirable software quality attributes can be achieved through using the design patterns described earlier in this paper. For this discussion, we refer to case study results and observations.

Reusability

Reusability refers to the degree to which existing applications can be reused in new applications. Reusability of source code reduces implementation time and increases the probability that prior testing has eliminated defects.

In Table 2, we show our reusability analysis of the source code of the ADD systems of Game-P and Game-S. In the first column, we show the class name or pattern name. In the next four columns we show information related to Game-P. In the first Game-P column we show the number of classes in each category (i.e., specified in column 1). In the second column we show the corresponding total logical SLOC in Game-P. In the third column we show the reusable Logical SLOC (i.e., code that remained unchanged in Game-S) and the associated percentage. In the fourth column we show the game specific Logical SLOC (i.e., specific to Game-P and cannot be reused) and the associated percentage. The remaining columns report similar data, this time from the perspective of Game-S. For clarity, we combined 100% reusable classes within a particular pattern. After all the rows of a particular pattern we show the summary of that pattern. The last row of the table is the summary across all the patterns.

We can see from Table 2 that *SensorFactory*, *Sensor*, *Registry* and *ResourceManager* classes in the sensor factory design pattern are completely reusable. Similarly, classes required to implement the *Observer*, *Trigger*, *Threshold* and *ThresholdAnalyzer* in the adaptation detector pattern are completely reusable. Three classes (i.e., *Rule*, *FixedRules* and *Decision*) in the case based reasoning pattern, and three classes (i.e., *Driver*, *AdaptationDriver* and *State*) in the game reconfiguration pattern are also completely reusable. Furthermore, the classes required to implement *AdaptationDetector*, *InferenceEngine* and *GameState* are partially reusable. Only the concrete sensors (6 classes in Game-P and 3 classes in Game-S) and the concrete decisions (2 classes in Game-P and 5 classes in Game-S) are specific to the game and not reusable.

As we can see from the last row in Table 2, the ADD system in Game-P contains 27 classes comprised of 774 logical SLOC. Similarly, the ADD system in Game-S contains 753 logical SLOC in 27 classes. Between these two systems, 600 logical SLOC (77.52% in Game-P; 79.68% in Game-S) are exactly the same and thus are considered reusable. Only 174 (22.48%) logical SLOC in Game-P and 153 (20.32%) logical SLOC in Game-S are specific to the games. Overall, more than three fourths (75%) of the logical SLOC required to implement the ADD systems are considered reusable.

Integrability

Integrability refers to the ability to make the separately developed components of the system work correctly together. As we can see in Figure 5, the integration points among the design patterns and with the game are clearly defined. *Observer*s, *Trigger*s and *Decision*s are the integration points between the four design patterns. *Sensor*s and *Driver*s are the integration points between a game and the ADD system. *Sensor*s function as accessors to the game whereas *Driver*s function as mutators to the game. Because of these clearly defined integration points, the four design patterns can be integrated with each other and a game easily. One of the games in the case study (Game-S) was already developed without any prior consideration of these design patterns or even any ADD system. Regardless, we easily managed to extend and add our ADD system to that game using our design patterns, which demonstrates the integrability of the patterns (and also Game-S).

Table 2: Reusability Analysis of the Source Code of ADD Systems in Game-P and Game-S

Class/ Pattern Name	Game-P				Game-S			
	# of Classes	Logical SLOC			# of Classes	Logical SLOC		
		Total	Reusable(%)	Specific(%)		Total	Reusable(%)	Specific(%)
SensorFactory, Sensor, Registry, Resource Manager	4	218	218(100)	0(0)	4	218	218(100)	0(0)
ConcreteSensors	6	68	0(0)	68(100)	3	44	0 (0)	44(100)
Sensor Factory	**10**	**286**	**218(76.22)**	**68(23.78)**	**7**	**262**	**218 (83.21)**	**44(16.79)**
Observer, Trigger, Threshold, ThresholdAnalyzer	5	97	97(100)	0(0)	5	97	97(100)	0(0)
AdaptationDetector	1	68	21(30.88)	47(69.12)	1	65	21(32.31)	44(67.69)
Adaptation Detector	**6**	**165**	**118(71.52)**	**47(28.48)**	**6**	**162**	**118(72.84)**	**44(27.16)**
Rule, Decision, FixedRules	3	75	75(100)	0(0)	3	75	75(100)	0(0)
InferenceEngine	2	50	46(92)	4(8)	2	51	46(90.20)	5(9.80)
ConcreteDecisions	2	29	0(0)	29(100)	5	30	0(0)	30(100)
Case-based Reasoning	**7**	**154**	**121(78.57)**	**33(21.43)**	**10**	**156**	**121(77.56)**	**35(22.44)**
Driver, AdaptationDriver, State	3	99	99(100)	0(0)	3	99	99(100)	0(0)
GameState	1	70	44(62.86)	26(37.14)	1	74	44(59.46)	30(40.54)
Game Reconfiguration	**4**	**169**	**143(84.62)**	**26(15.38)**	**4**	**173**	**143(82.66)**	**30(17.34)**
Grand Total	**27**	**774**	**600(77.52)**	**174(22.48)**	**27**	**753**	**600(79.68)**	**153(20.32)**

Portability

Portability is the ability of a system to run under different computing environments. A framework- or middleware-based approach for creating a self-adaptive system (such as ADD in video games) is usually specific to a particular programming language and or platform, whereas a design pattern-based approach is highly portable across different platforms and programming languages (Ramirez and Cheng 2010). These design patterns were derived from the self-adaptive system literature in the context of ADD in video games. This indicates the portability of these design patterns across domains. Also, in our case study, we managed to port them (as a solution) from one game to another within the platform (Java). This indicates portability across systems on the same platform. In the future, we plan to examine the portability of these design patterns across platforms as well.

Maintainability

Maintainability refers to the ease of the future maintenance of the system. As discussed earlier, different parts of the design patterns have specific concerns (e.g., *Sensor*s will collect data, *Driver*s will make changes to the game, etc.), and so the resulting source code will have high traceability and maintainability. Furthermore, as the use of these design patterns provides source code reusability (please see Table 2), this will increase the probability that prior testing has eliminated defects when being used in a new game.

CONCLUDING REMARKS

Design patterns are a formal approach of describing reusable solutions for a design problem. To date, the literature on the usage of software design patterns in video games is relatively scarce. Little or no motivation of using software design patterns for implementing ADD is found in video game literature. Thus, in this paper, we presented a case study involving implementation and source code analysis of two proof-of-concept video games. We discussed how desirable software quality attributes such as reusability, integrability, portability, and maintainability can be achieved through the usage of these design patterns. Our case study results and methods have implications on both research and practice, giving practitioners motivation to use these design patterns for implementing ADD systems. Our analysis technique (i.e., a source code analysis to compare games) can be used in further research. Even though our context of discussion was ADD, these patterns can be used in any situation where a game needs to be adaptive and reconfigures itself based on monitoring.

REFERENCES

Antonio, M.; Jiménez-Díaz, G.; and Arroyo, J. 2009. "Teaching Design Patterns Using a Family of Games". In *Proceedings of the 14th Annual ACM SIGCSE Conference on Innovation and Technology in Computer Science* (Paris, France, Jul. 6-9). 268-272.

Bailey, C.; and Katchabaw, M. 2005. "An Experimental Testbed to Enable Auto-Dynamic Difficulty in Modern Video Games". In *Proceedings of the 2005 North American Game-On Conference* (Montreal, Canada, Aug. 22-23). 18-22.

Björk, S., Holopainen, J. 2004. *Patterns in Game Design*. Charles River Media, Inc. Massachusetts, USA.

Brackeen, D.; Barker, B.; and Vanhelsuwé, L. 2004. *Developing Games in Java*. New Riders.

Chowdhury, M. I.; and Katchabaw, M. 2012. "Software Design Patterns for Enabling Auto Dynamic Difficulty in Video Games". In *Proceedings of the 17th Intl. Conf. on Computer Games* (Louisville, Kentucky, USA, Jul. 30 - Aug. 1). 76 - 80.

Gamma, E.; Helm, R.; Johnson, R.; and Vissides, J. 1995. *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison - Wesley.

Gestwicki, P.; Sun, F. 2008. "Teaching Design Patterns Through Computer Game Development". *Journal on Educational Resource in Computing* 8, No. 1 (Mar), 1 -22.

Hao, Y.; He, S.; Wang, J.; Liu, X.; Yang, J.; and Huang, W. 2010. "Dynamic Difficulty Adjustment of Game AI by MCTS for the Game Pac-Man". In *Proceedings of the Sixth International Conference on Natural Computation* (Yantai, China, Aug. 10-12). 3918-3922.

Hocine, N.; and Gouaïch, A. 2011. "Therapeutic Games' Difficulty Adaptation: An Approach Based on Player's Ability and Motivation". In *Proceedings of the 16th Intl.Conf. on Computer Games* (Louisville, Kentucky, USA, Jul. 27-30). 257 - 261.

Hunicke, R. 2005. "The Case for Dynamic Difficulty Adjustment in Games". In *Proceedings of the 2005 ACM SIGCHI International Conf. on Advances in Computer Entertainment Technology* (Valencia, Spain, Jun. 15-17). 429-433.

Narsoo, J.; Sunhaloo, M. S.; and Thomas, R. 2009. "The Application of Design Patterns to Develop Games for Mobile Devices using Java 2 Micro Edition". *Journal of Object Technology* 8, No. 5 (Jul.,Aug.), 153 - 175.

Orvis, K. A.; Horn, D. B.; and Belanich, J. 2008. "The Roles of Task Difficulty and Prior Videogame Experience on Performance and Motivation in Instructional Videogames". *Computers in Human Behavior* 24, No.5 (Sep), 2415 -2433.

Ramirez, A. J.; and Cheng, B. H. C. 2010. "Design Patterns for Developing Dynamically Adaptive Systems". In *Proceeding of the 2010 ICSE Workshop on Software Engineering for Adaptive and Self-Managing Systems* (Cape Town, South Africa, May. 3-4). 49 - 58.

Rani, P.; Sarkar, N.; and Liu, C. 2005. "Maintaining Optimal Challenge in Computer Games Through Real-time Physiological Feedback". In *Proceedings of the 11th Int.l Conf. on Human-Computer Interaction* (Las Vegas, USA, July. 22-27). 184-192.

Snow, B. 2011. "Why Most People Don't Finish Video Games" Online publication in CNN, August 17, 2011. Retrieved from: http://www.cnn.com/2011/TECH/gaming.gadgets/08/17/finishing.videogames.snow/. Last accessed: July 04, 2012.

BIOGRAPHIES

MUHAMMAD IFTEKHER CHOWDHURY is a PhD candidate working in the area of game design at the University of Western Ontario. He completed his Masters in software engineering from the same university.

MICHAEL KATCHABAW is an Associate Professor at the University of Western Ontario. His research interests include game design and development. He co-founded the Digital Recreation, Entertainment, Art, and Media (DREAM) research group at Western.

Real-time Adaptive Track Generation in Racing Games

Jake Bird, Tom Feltwell and Grzegorz Cielniak
School of Computer Science
University of Lincoln, UK
email: birdyjake@aol.com; tfeltwell,gcielniak@lincoln.ac.uk

KEYWORDS
Procedural Content Generation, Real-time Adaptability, Racing Car Simulation

ABSTRACT

This paper addresses the problem of adaptability in the context of racing games and proposes a real-time track generation system automatically adjusting to the player's performance. The system modifies the track difficulty according to a heuristic model of the player's experience. The conducted experiments demonstrate the feasibility of real-time adaptability in the racing game context and show that this feature can positively influence the player's perception of challenge and fun.

INTRODUCTION

Player's gaming experience is the most critical aspect of modern game development. The so called "fun" factor captures the essence of that experience and provides intuitive understanding of what is expected from game designers. In context of racing games, the "fun" factor is affected largely by the amount and varied type of challenge, which can be attributed to characteristics such as ability to drive fast, varied tracks (i.e., avoiding long straight sections or continuous bends), realistic simulation of the car, including drifting and skidding, and continuous improvement of skills both for beginner and expert players (Togelius et al. 2006). It is important to note that the player's perception of challenge is not static but dynamically changes together with their improving abilities.

Commercial games are usually not tailored to incorporate any individual playing style, at most providing a fixed set of generic difficulty settings. The lack of *adaptability* is an inherent characteristic of a traditional, manual design process which negatively affects the "fun" factor, re-playability, potential for exploring the game to its full extent, but also associated asset creation costs.

RELATED WORK

Recently, Procedural Content Generation (PCG) techniques have been proposed to address various challenges in video game context (see (Togelius et al. 2011) for an extensive survey). PGC methods are being used during the game development phase to aid designers in automated asset creation, including generation of decorative elements like weather conditions, lighting or textures (e.g., (Whitehead 2010)), but also of terrain, foliage or game levels (e.g., (Hastings et al. 2009, Pedersen et al. 2010)). In such scenarios, the quality of created assets is judged by the designer, who decides if a particular result is suitable for the specific game.

PCG methods can also be used during the actual gameplay, enabling adaptability of various design aspects (e.g. assets, NPCs or game mechanics). In this case, the feedback information to the adaptive algorithm consists of some measure of player's experience, and the general aim is to improve that experience by introducing varying challenge and novelty into the game-play (e.g., (Yannakakis and Togelius 2011)). There are several ways of measuring the player's experience including questionnaire-based feedback, by taking physiological (i.e., biometric) responses or extracted directly from the game-play, by monitoring metrics that approximate player's performance, skill and engagement (Bernhaupt 2010). The former two approaches pose obvious limitations including subjectivity of the questionnaire response or additional apparatus required for measuring biofeedback.

The analysis of in-game generated metrics for adaptation is particularly interesting as it can be applied in real-time. There are however only a few examples of such systems implemented so far, including for example automatically adjusted difficulty in physical games (Yannakakis and Hallam 2009) or FPS games (e.g. Left4Dead (Booth 2009)). In racing context, the adaptation has been explored for modelling AI opponents (e.g. Drivatar system (Microsoft Research 2012)) or adapting tracks through evolutionary algorithms (Togelius et al. 2006). However, adjustment in these systems does not occur in real-time.

This paper presents a system for generating racing tracks which are adjusted in real-time according to the player's performance. The presented experiments demonstrate that the real-time adaptability is feasible in the racing game context and that it can positively influence the player's perception of challenge and fun.

THE METHOD

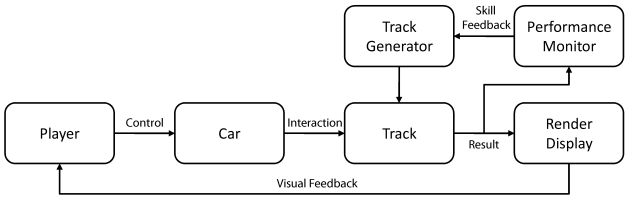


Figure 1: Overview of the proposed system.

This section presents the core components of the proposed game system (see Fig. 1). The system consists of two real-time feedback loops: the main loop is a classic player-game loop with the output (i.e., feedback to the user) being displayed on the screen. There is also an additional loop consisting of a module which assesses the player's performance and feeds this information back to the track generator, resulting in an adaptively adjusted track which in turn affects the player actions.

The Car Model

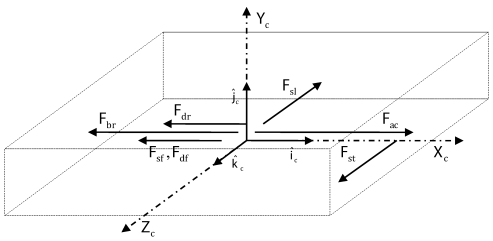


Figure 2: The model of the car used in the simulation together with all forces implemented.

The car is simulated as a rigid cuboid of length l_c, width w_c, height h_c and mass m_c approximately matching that of a real car (see Fig. 2). The forward and backward movement of the car is controlled by applying an acceleration force $\boldsymbol{F}_{ac}$ along the local X_c axis, which is generated by a simulated automatic gearbox. To simulate brakes, a braking force $\boldsymbol{F}_{br}$ is applied in opposite direction to the car movement. To improve the responsiveness of brakes for higher velocities, a modified formula for $\boldsymbol{F}_{br}$ that takes into account the car speed v_c has been used. The car is controlled by a steering force $\boldsymbol{F}_{st}$ proportional to the car speed v_c and applied to the front of the car along the Z_c axis. The other simulated forces consist of gravity $\boldsymbol{F}_g$ and resistive forces including air drag $\boldsymbol{F}_{dr}$, static friction $\boldsymbol{F}_{sf}$, dynamic friction $\boldsymbol{F}_{df}$ and sliding friction $\boldsymbol{F}_{sl}$ to simulate skidding. All forces with an exception of $\boldsymbol{F}_{st}$ are applied to the car's centre of mass. Table 1 presents detailed formulas for each modelled force while Table 2 presents parameter values used in the simulation.

Table 1: Simulated forces, where v_c denotes the car speed and $\hat{\boldsymbol{i}}_c$, $\hat{\boldsymbol{j}}_c$, $\hat{\boldsymbol{k}}_c$ define the local car coordinates.

Force	Descritpion
$\boldsymbol{F}_g = -gm_c\hat{\boldsymbol{j}}$	gravity (global force)
$\boldsymbol{F}_{ac} = F_{ac}\hat{\boldsymbol{i}}_c$	acceleration force
$\boldsymbol{F}_{br} = -c_{br}v_c\hat{\boldsymbol{i}}_c$	braking force
$\boldsymbol{F}_{st} = \pm c_{st}v_c\hat{\boldsymbol{k}}_c$	steering force
$\boldsymbol{F}_{dr} = -c_d v_c^2\hat{\boldsymbol{i}}_c$	air drag
$\boldsymbol{F}_{sf} \leq -\mu_s gm_c\hat{\boldsymbol{i}}_c$	static friction
$\boldsymbol{F}_{df} = -\mu_d gm_c\hat{\boldsymbol{i}}_c$	dynamic friction
$\boldsymbol{F}_{sl} = F_{sl}\hat{\boldsymbol{k}}_c$	sliding friction

Table 2: Physical parameter values used in the simulation.

Parameter	Name	Value
g	gravitational constant	9.8 m/s^2
l_c	car length	4.45 m
w_c	car width	1.70 m
h_c	car height	0.95 m
m_c	car mass	1182 kg
c_{br}	braking coeff.	10^5 kg/s
c_{st}	steering coeff.	$3 \cdot 10^4$ kg/s
c_d	air drag coeff.	300 kg/m
$\mu_s = \mu_d$	friction coeff.	0.3
F_{sl}	sliding force	10^6 N

The Track Model

The track is composed of alternating segments, called "straights" and "curves" (see Fig. 3). Straight segments are simply rectangles with varying length l, width w and gradient α parameters. Curved segments are partial annuli with varying turn radius l, turn angle θ, width w, gradient α and camber β parameters. The gradient of a segment α is the slope of the road measured along its length; on a segment with a positive gradient, the start of the segment is lower than the end. It is measured as the angle the length makes with the horizontal plane. The camber β of a curve is the slope of the road measured across its width; on a curve with a positive camber, the inside of the curve is lower than the outside. It is measured as the angle the width makes with the horizontal plane. All segment parameters are embedded into a single vector $\boldsymbol{x}_s = \{l, \theta, w, \alpha, \beta\}$ which is generated in real-time by the track generation algorithm as the game progresses.

Track Generation

The track generation algorithm continuously generates alternating straight and curved segments according to Alg. 1. After each segment, the track parameters are ad-

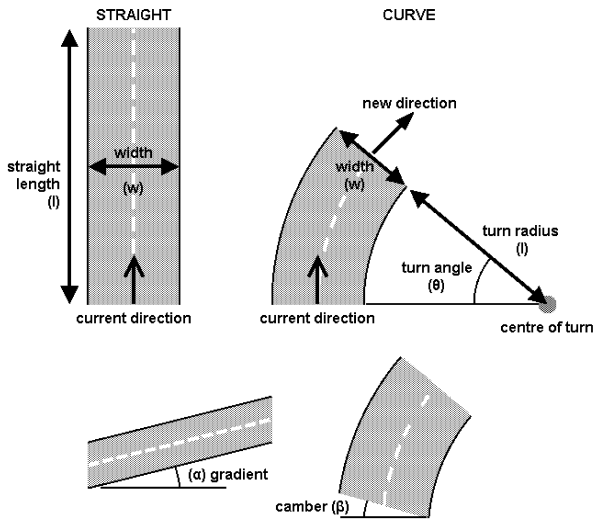


Figure 3: Track segments and their parameters.

Algorithm 1: Track generation algorithm.

1. Initialise all segment parameters $\boldsymbol{x}_s$ with default values.
2. Generate a new straight segment $\boldsymbol{x}_s = \{l, -, w, \alpha, -\}$.
3. Adjust the track parameters according to Algorithm 2.
4. Generate a new curve segment: $\boldsymbol{x}_s = \{l, \theta, w, \alpha, \beta\}$.
5. Adjust the track parameters according to Algorithm 3.
6. Go to Step 2.

justed according to the estimated player's performance which takes into account a skill level metric and selected events that might have occurred over the course of the completed segment. Values for l and θ are generated from probability distributions - uniform in the presented case, but could be of any type in principle. The distribution parameters for turn angle can be determined from other variables: $\theta_{min} = l_{MIN}/l$ and $\theta_{max} = \theta_{MAX}$, while l_{min} and l_{max} are being incrementally adjusted together with the values of segment parameters w, α and β according to Algorithms 2 and 3.

The presented algorithm modifies the track variables based on a set of heuristics which follow observations that the main difficulty in racing games arise from bendy, narrow tracks of variable turn directions. Gradient α depends on calculated skill level p (see Skill Level Estimation Section for details), while other parameters are adjusted when the player hits one of the

Algorithm 2: Track adjustment after the straight segment.

- Width adjustment: $w = w \pm \Delta w$. If the player hit the edge of the track, increase w, otherwise decrease w.
- Gradient adjustment: $\alpha = \alpha \pm \Delta\alpha$. Calculate skill level p (see Skill Level Estimation Section for details). If p exceeds the specified threshold p_t decrease α, otherwise increase α.
- Generate new turn radius and angle from the uniform distribution: $l = \mathcal{U}(l_{min}, l_{max})$, $\theta = \mathcal{U}(\theta_{min}, \theta_{max})$.

walls, or/and drives with the speed different to the nominal speed v_n. The nominal speed $v_n = [\sqrt{l \cdot n_{lo}}, \sqrt{l \cdot n_{hi}}]$ is an experimentally derived measure that defines an "appropriate" speed interval for a given length/radius of a curve segment.

Algorithm 3: Track adjustment after the curved segment.

- Minimum length/radius adjustment: $l_{min} = l_{min} \pm \Delta l_{min}$. If the player hit the outside wall with the speed higher than the nominal speed v_n, increase l_{min}, otherwise decrease l_{min}.
- Maximum length/radius adjustment: $l_{max} = l_{max} \pm \Delta l_{max}$. If the player approached the corner with the speed lower than the nominal speed v_n decrease r_{max}, otherwise increase r_{max}.
- Camber adjustment: $\beta = \beta \pm \Delta\beta$. If the player hit the inside wall, decrease camber (i.e., make road lean more towards the outside of corners). If the player hit the outside wall with the nominal speed v_n, increase camber (i.e., make road lean more towards the inside of corners). If the player did not hit a wall, reduce absolute camber.
- Gradient adjustment: $\alpha = \alpha \pm \Delta\alpha$. Calculate skill level p (see Skill Level Estimation Section for details). If p exceeds the specified threshold p_t decrease α, otherwise increase α.
- Generate new length value from the uniform distribution: $l = \mathcal{U}(l_{min}, l_{max})$.

Parameter adjustment can be performed in a number of ways. Here, three methods are proposed:

1. Linear Adjustment: a constant value (different for each variable) is added or subtracted to that variable, e.g., $\Delta\alpha = c_\alpha$.

2. Proportional Adjustment: a constant value (different for each variable) is multiplied or divided by that variable, e.g., $\Delta\alpha = \alpha(c_\alpha - 1)$. This method is not applicable to camber and gradient where both positive and negative values are used and therefore these parameters will be adjusted using Linear Adjustment.

3. Width-weighted Linear Adjustment: a constant value (different for each variable) is multiplied by the normalised width value before being added or subtracted to that variable, e.g., $\Delta\alpha = c_\alpha(w - w_{MIN})/(w_{MAX} - w_{MIN})$. This method should place a greater importance on avoiding the walls.

Skill Level Estimation

The skill level p is calculated by comparing the minimum and maximum values of length/radius, width and camber against their respective absolute values. A highly skilled player will have l_{min} and w close to their lower limits, l_{max} at its upper limit, and $\beta = 0$. Conversely, a less skilled player will have l_{min} and w close to their upper limits, l_{max} at its lower limit, and $\beta = \beta_{MAX}$. The resulting values are then normalised and their weighted sum is used as the skill level for a given segment.

Implementation Details

The described components were integrated into a car racing game implemented using the NVIDIA PhysX library for physics simulation (ver. 2.8.4) and OpenGL for basic rendering. Each track segment has a wall on both left and right sides, perpendicular to the road of the track itself in order to prevent the player from leaving the track. The graphical model of the car is based on Lotus Evora. Since the generated track could be of arbitrary length, only part of the track is kept in memory: this includes two generated segments ahead and two previous ones with respect to the current segment. To avoid misalignments between the segments when $\beta \neq 0$, there is also an additional short linking component that connects the vertices of adjoining segments.

EXPERIMENTS

Data Collection

A random sample of 16 students and lecturers at the University of Lincoln were asked to play four tracks, each consisting of 50 segments and taking approximately 2:30 min. to complete. Three of the tracks are generated adaptively, using each of the three adjustment methods listed previously, and the fourth pre-generated control track that does not adapt to the player's actions. The tracks are presented in a random order to the participants so as to prevent bias.

After completing all four tracks, each of the participants was asked to complete a questionnaire asking participants to rate each track's "fun" and "challenge" level and also to select their preferred track. The questionnaire also asked participants about the frequency with which they play racing games (i.e. number of titles played in the last six months) as well as how proficient they felt they are at racing games. The ratings were expressed in five-point Likert scale.

Results

Table 3: User responses to the questionnaire with respect to tracks generated using different adjustment methods (1-3) and the control track (C).

	Adjustment Method			
Rating	1	2	3	C
Fun	3.88	3.56	2.75	2.50
Challenge	3.44	3.36	3.56	2.44
Preference	8	4	3	1

The summary of results from the questionnaire are presented in Table 3. On average, the tracks generated using Linear Adjustment have the highest average "fun" value, and the control track the lowest. The Proportional Adjustment method had a similar value to that of Linear Adjustment, whereas the Width-weighted Adjustment tracks scored comparably to the control track. This suggests that the width-weighted method contributed less to the perceived fun factor than the other two methods. The average difficulty remains about the same throughout the tracks generated adaptively. This was expected, as the track generation attempts to match the player's skill level. The control track is lower in difficulty, as it does not adapt to the player's skill level. This may suggests that either the control track is too easy or the adaptive tracks are too difficult for the average player. Majority of the participants preferred the tracks generated using Linear Adjustment, which indicates that this is the most suitable track generation method to satisfy the player's gaming experience.

Figure 4 presents the mean skill level p for a given questionnaire response type and its value. Four different skill levels were calculated; the "challenge" parameter applies to the track the skill level was calculated for, whereas the "frequency" and "proficiency" responses are simply duplicated across the four tracks.

On tracks which participants felt that were more challenging, their calculated skill level was lower, and vice versa. However, given that the track should adapt to the player's skill level, the challenge should remain approximately constant, or at least have little correlation. This suggests that while the track is adapting, it is not

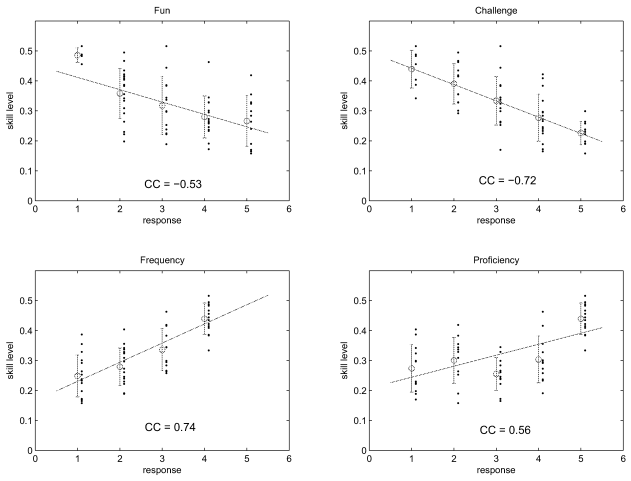


Figure 4: Skill level with respect to different response type and value (CC - sample correlation coefficient).

doing so enough with respect to both the beginners and expert players.

As there was a strong correlation between skill level and playing frequency, it can be said both that frequency of playing racing games is a good indication of skill level, and that the skill level calculation is an accurate assessment of the player's ability. This shines a somewhat interesting light on how players perceive their skill level, as there is a much lower correlation between skill level and proficiency. Many racing games use a difficulty level to allow the player to choose for themselves; it may be that this is a flawed idea, as players may not be able to accurately determine their own skill level.

It is worth noting that no conclusion can be drawn from these results on the comparative playing experience between an adaptive track and a well-designed prescribed track. However, a designer can only make a finite number of tracks, thus making adaptive track generation a more viable choice for creating a very large number of tracks, or for creating a track which has no ending.

CONCLUSIONS

The conducted experiments demonstrate that the proposed method, despite its relative simplicity, is a feasible option for generating adaptive racing tracks in real-time. It seems that adaptive track generation in racing games can provide the player with a greater level of "fun factor" than a non-adaptive track generation, but more research is needed in order to confirm its effectiveness in replacing/assisting a human designer. Other shortcomings of the presented work include a relatively low sample of the collected data which may have significantly affected some of the derived conclusions; data collected from an on-line release of the game would allow for more reliable results. The game itself could be integrated into the existing racing simulators (e.g., (TORCS 2005)) that could lead to more experimental data available, but also would enable enhanced physics modelling and additional game functionality. The presented adaptive models are ad-hoc and manually crafted for a generic user. The model parameters (weights, increments, etc.) could be learnt directly from data and tailored to a specific user type, brining it closer to other work on adaptive racing tracks (e.g. (Togelius et al. 2006)).

REFERENCES

R. Bernhaupt. *Evaluating User Experience in Games: Concepts and Methods.* Springer Publishing Company, Incorporated, 1 edition, 2010.

M. Booth. The AI Systems of Left 4 Dead. In *Keynote, Artificial Intelligence and Interactive Digital Entertainment Conference*, Palo Alto, CA, USA, 2009.

E. J. Hastings, R. K. Guha, and K. O. Stanley. Automatic Content Generation in the Galactic Arms Race Video Game. *IEEE Transactions on Computational Intelligence and AI in Games*, 1(4):245–263, 2009.

Microsoft Research. Drivatar Technology. Website, 2012. URL `http://research.microsoft.com/en-us/projects/drivatar/default.aspx`.

C. Pedersen, J. Togelius, and G. N. Yannakakis. Modeling player experience for content creation. *IEEE Transactions on Computational Intelligence and AI in Games*, 2(1):54–67, 2010.

J. Togelius, R. D. Nardi, and S. M. Lucas. Making racing fun through player modeling and track evolution. In *Proc. of the Workshop on Adaptive Approaches for Optimizing Player Satisfaction in Computer and Physical Games*, 2006.

J. Togelius, G. N. Yannakakis, K. O. Stanley, and C. Browne. Search-based procedural content generation: A taxonomy and survey. *IEEE Trans. Comput. Intellig. and AI in Games*, 3(3):172–186, 2011.

TORCS. The Open Racing Car Simulator. Website, 2005. URL `http://www.torcs.org`.

J. Whitehead. Toward proccedural decorative ornamentation in games. In *Proc. of Workshop on Procedural Content Generation in Games*, PCGames '10, pages 9:1–9:4, New York, NY, USA, 2010. ACM.

G. N. Yannakakis and J. Hallam. Real-time Game Adaptation for Optimizing Player Satisfaction. *IEEE Transactions on Computational Intelligence and AI in Games*, 1(2):121–133, June 2009.

G. N. Yannakakis and J. Togelius. Experience-driven procedural content generation. *T. Affective Computing*, 2(3):147–161, 2011.

PROCEDURAL MAP GENERATION FOR A RTS GAME

Raúl Lara-Cabrera, Carlos Cotta and Antonio J. Fernández-Leiva

Department "Lenguajes y Ciencias de la Computación"
University of Málaga
Louis Pasteur, 35, 29071, Málaga – Spain
{raul,ccottap,afdez}@lcc.uma.es

KEYWORDS

Procedural content generation (PCG), Evolutionary Computation, Real-time Strategy Game (RTS)

ABSTRACT

Procedural content generation (PCG) is the programmatic generation of game content using a random or pseudo-random process that results in an unpredictable range of possible game play spaces. This methodology brings many advantages to game developers, such as reduced memory consumption. In this paper we introduce a procedural map generator for a real-time strategy (RTS) game. The main component of this generator is a genetic algorithm devoted to create and evolve balanced maps, i.e. maps where no player has any map related advantage with respect to other players. The selected RTS game is called *Planet Wars* and it was used in the *Google AI Challenge 2010*. It is a space conquest game whose objective is to take over all the planets on the map.

INTRODUCTION

This paper introduces a map generation method for a RTS game. This method can be classified as a procedural content generation method (PCG). PCG refers to creating game content automatically, through algorithmic means. This content refers to all aspects of the game that affect game-play other than non-player character (NPC), such as maps, levels, dialogues, characters, rule-sets and weapons. PCG is interesting for the game developing community due to several reasons, such as reduced memory consumption and the saving in the expense of manually creating game content. Our map generation method can be categorized (using the taxonomy proposed in Togelius, Yannakakis, Stanley & Browne 2011) as an off-line method that generates necessary content, using random seeds and deterministic generation and following a generate-and-test schema.

Procedural content generation has been used in many well-known video-games. *Borderlands* Gearbox Software 2009 uses a PCG system to create weapons and items, which can alter their firepower, rate of fire, and accuracy, add in elemental effects such as a chance to set foes on fire or cover them in burning acid, and at rare times other special bonuses such as regenerating the player's ammo. PCG system is also used to create the characteristic of random enemies that the player may face. Another example of a game that uses PCG is *Minecraft* Mojang 2011, a sandbox-building game with an infinite map which is expanded dynamically. *Spore* Maxis 2008 is a god game simulation that contains multiple levels of play, from starting as a multi-celled organism in a tide pool, up to exploring a dynamically generated universe with advanced UFO technology. The music of the game is also procedurally generated.

From an academic point of view, there are several papers related to procedural map generation. In Togelius, De Nardi & Lucas 2007 the authors designed a system for offline/online generation of tracks for a simple racing game. A racing track is created from a parameter vector using a deterministic genotype-to-phenotype mapping. A search-based procedural content generation (SBPCG) algorithm for strategy game maps is proposed in Togelius, Preuss & Yannakakis 2010 from a multi-objective perspective. A multi-objective evolutionary algorithm is used for searching the space of maps for candidates that satisfy pairs of these multiple objectives. Another search-based method for generating maps is presented in Togelius, Preuss, Beume, Wessing, Hagelback & Yannakakis 2010. In this case, the maps are generated for the game *Starcraft* Blizzard Entertainment 1998. Frade et al. have introduced the idea of terrain programming, namely the use of genetic programming to evolve playing maps for video-games, using either subjective human-based feedback Frade, de Vega & Cotta 2008, Frade, de Vega & Cotta 2009 or automated quality measures such as accessibility Frade, de Vega & Cotta 2010*a* or edge-length Frade, de Vega & Cotta 2010*b*. In Mahlmann, Togelius & Yannakakis 2012 the authors describe a search-based map generator for an abstract version of the real-time strategy game *Dune 2*. Map genotypes are represented as low-resolution matrices, which are then converted to higher-resolution maps through a stochastic process involving cellular automata.

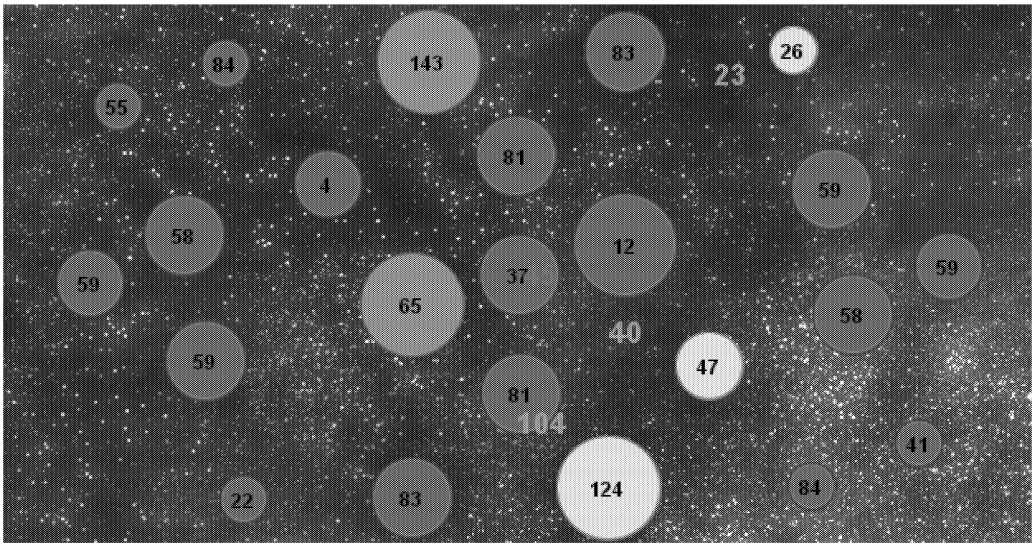


Figure 1: A screenshot of Planet Wars

In the next section we describe the RTS game that has been used in the experiments. Then, we describe a procedural map generator and a genetic algorithm that creates and evolves balanced map. Right after this description, there is a section where we report the results we have obtained from the experiments. Finally, we present our conclusions and future work.

GAME DESCRIPTION

Planet Wars is a real-time strategy (RTS) game based on *Galcon* and used in the *Google AI Challenge 2010* (a screenshot is shown in figure 1). It is set in outer space and its objective is to take over all the planets on the map, or alternatively eliminate all of your opponents ships. A game of Planet Wars takes place on a map which contains several planets, each of which has some number of ships on it. Each planet may have a different number of ships. The planets may belong to one of three different owners: you, your opponent, or neutral. The game has a certain maximum number of turns. The game may end earlier if one of the players loses all his ships, in which case the player that has ships remaining wins instantly. If both players have the same number of ships when the game ends, its a draw. On each turn, the player may choose to send fleets of ships from any planet he owns to any other planet on the map. He may send as many fleets as he wishes on a single turn as long as he has enough ships to supply them. After sending fleets, each planet owned by a player (not owned by neutral) will increase the forces there according to that planets growth rate. Different planets have different growth rates. The fleets will then take some number of turns to reach their destination planets, where they will then fight any opposing forces there and, if they win, take ownership of the planet. Fleets cannot be redirected during travel. Players may continue to send more fleets on later turns even while older fleets are in transit. Despite players make their orders on a turn-by-turn basis, they issue these orders at the same time, so we can treat this game as a real-time game.

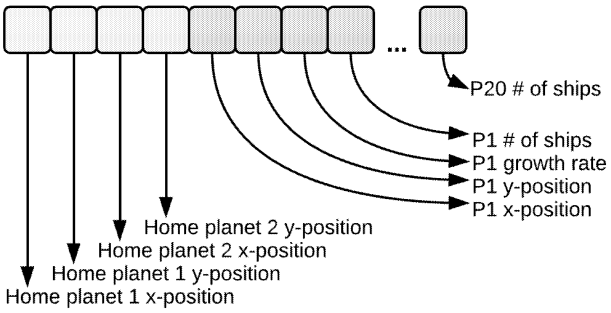


Figure 2: Structure of the individual

Maps have no particular dimensions and are defined completely in terms of the planets and fleets in them. They are defined in plain text files, with each line representing a planet or a fleet. Planet positions are specified relative to a common origin in Euclidean space. The coordinates are given as floating point numbers. Planets never move and are never added or removed as the game progresses. Planets are not allowed to occupy the exact same position on the map. The owner of a planet can be neutral, player 1, or player 2. The number of ships is given as an integer, and it may change throughout the game. Finally, the growth rate of the planet is the number of ships added to the planet after each turn. If the planet is currently owned by neutral, the growth rate is not applied. Only players can get new ships through growth. The growth rate of a planet will never change.

It is given as an integer.

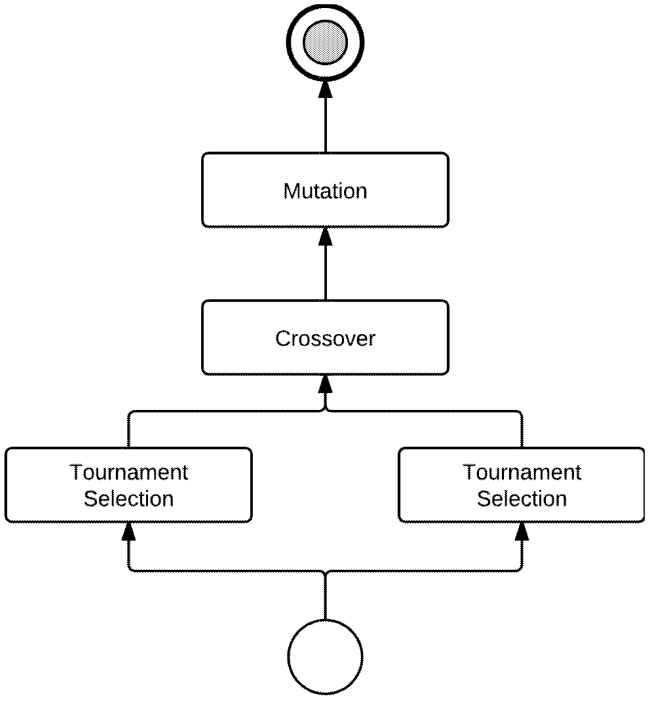


Figure 3: Genetic algorithm's breeding pipeline

PROCEDURAL MAP GENERATOR

In this section, we present a search-based procedural map generator that is capable of generating balanced maps for the real-time strategy game *Planet Wars*. It is composed of two systems, a genetic algorithm responsible for generating and evolving maps and a system responsible for playing Planet Wars games and evaluating the maps. The evaluator is a tool developed by *Google* for the *Google AI Challenge 2010*. It has been developed using Java and is a console application. It runs a customizable game between various players and generates a game trace. The game can be viewed with a visualizer (included in these tools) which reads this game trace. We have created a script that calls the evaluator with a specified map and stores the game trace to a file. Later, this file is processed to compute the fitness of the generated map. We have used a Java-based evolutionary computation research system, called ECJ[1], for constructing the genetic algorithm (a review of this system can be found in White 2012). It supports multithread evaluation and breeding, a master-slave architecture, island models, and even experimental support for GPGPU through a third-party extension. It is easily configurable because of it simple text-based parameter files. Its implementation in Java makes ECJ very portable. Unless ECJs graphical user interface (GUI) is needed, ECJ is self contained. The integration of networking and serialization support that Java provides makes developing new parallel architectures and checkpointing methods much easier than starting from scratch or using a third-party library.

[1] `http://cs.gmu.edu/~eclab/projects/ecj/`

The genetic algorithm generates maps with 20 neutral planets and two starting planets (one for each player), i.e. maps with 22 planets. The proposed genetic algorithm follows a generational scheme with elitism (the best solution always survive). As described on the previous section, every planet has five properties: x-position, y-position, owner, growth rate and number os ships. To obtain a balanced map, we have fixed the growth rate of the two starting planets. Planets' owners have been also fixed, so we got finally 84 parameters (4 for every neutral planet, and 2 for every starting planet). Each individual of our genetic algorithm is made of these 84 parameters, grouped into a vector of floating point numbers in the range between 1 and 5. The first two parameters are the x and y position of the home planet for player 1, while the next two parameters correspond to the position of the home planet for player 2. Then, there are 20 groups of 4 parameters, one group for each neutral planet, whose parameters are the x position, y position, growth rate and number of ships respectively (see figure 2).

Number of generations	100
Number of individuals	40
Crossover probability	0.75
Mutation probability	0.70
Replacement policy	1-elitism

Table 1: Genetic algorithm's parameters

The algorithm (see table 1) uses a population of 40 individuals on each generation with a runtime of 100 generations. It uses tournament selection (i.e. the algorithm selects two individuals and selects the one with the higher fitness) as the selection method, and crossover and mutation as breeding operators (figure 3 shows a detailed view of the breeding pipeline). The crossover method selected for our algorithm performs a line recombination: The two individuals are treated as points in space. A straight line is drawn through both points, and two children are created along this line. If the individuals are $\overrightarrow{x}$ and $\overrightarrow{y}$, we draw two random values α and β, each between $-p$ and $1+p$ inclusive. Then the two children are defined as $\alpha\overrightarrow{x} + (1-\alpha)\overrightarrow{y}$ and $\beta\overrightarrow{y} + (1-\beta)\overrightarrow{x}$ respectively (with $p = 0.75$). We have used Gaussian mutation as mutation operator. It adds Gaussian noise to the current value, this way, planets may be displaced or its size may be changed. If the result is outside the bounds of minimum and maximum legal values, another Gaussian noise is tried instead, and so on, until a legal value is found (or a certain number of retries is reached).

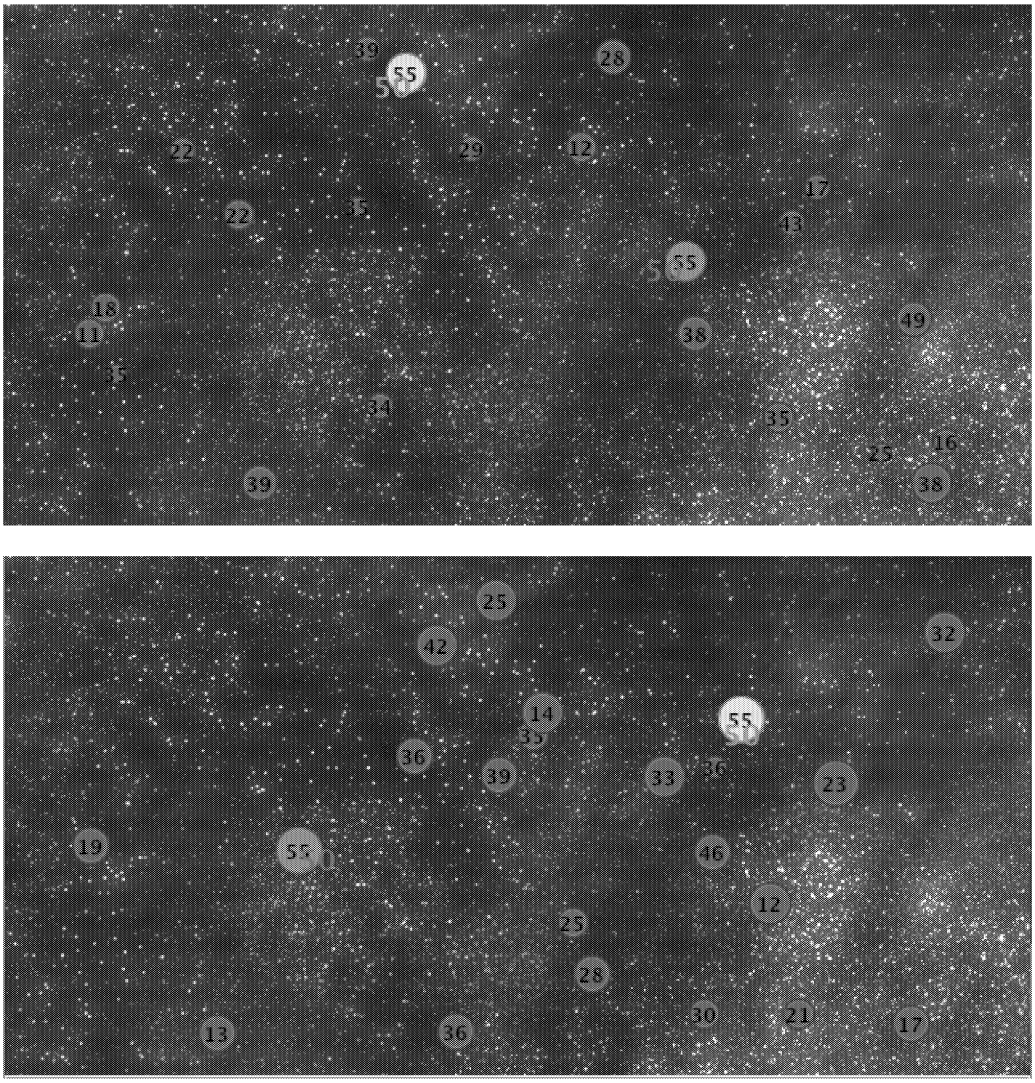


Figure 5: Balanced maps for Planet Wars.

To evaluate the fitness of every individual the algorithm makes a genotype-to-phenotype transformation, scaling the values of the individual. Positions are scaled between 10 and 50, the number of ships is rounded to an integer and scaled between 10 and 50 while the growth rate is just rounded to an integer. Once these parameters has been normalized, the algorithm writes the map to a file, using these parameters to generate the planets. Then, the algorithm runs a game that take place on the recently generated map between two players. These two players are instances of the same bot, this way the player's ability does not affect the measurement of how balanced has been this game. Once the game has finished, the algorithm gathers the total number of ships (S1 and S2 respectively) and planets (NP1 and NP2 respectively) owned by both players and compute the fitness function (1) with these values.

$$f = \frac{(NP1 * NP2)(S1 + S2)}{|S1 - S2| + 1} \tag{1}$$

We wanted to obtain balanced maps, that is, maps where a player doesn't get any advantage over the other players. This is the reason why we have the difference between the two players' number of ships on the denominator of the fitness function, because we wanted at least score difference as possible between the two players (is a balanced map). Keeping in mind the same objective, we sum these number of ships on the numerator to promote maps where the players last enough to build big fleets. This sum is multiplied by another multiplication (number of planets of both players) to penalize those maps where a player wins over another or both players remain static until the end of turns.

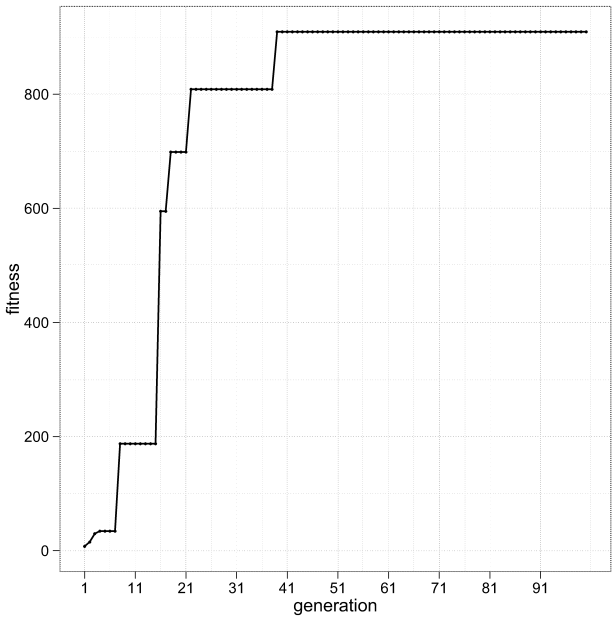


Figure 4: Evolution of the fitness.

RESULTS

We have run this algorithm several times, and we have obtained many fully playable and balanced maps. After making these experiments we have noticed that the evolution of the fitness is not constantly growing, that is, sometimes the fitness remains unchanged during a certain amount of generations. This fitness doesn't change because this genetic algorithm has elitism as the selection method (the best individual survives and is included in the next population). Another observation is that the planets of the generated maps are much separated from each other. Maps of this kind should have a high fitness value because it takes a long time (number of turns) to reach the enemy, so the number of ships for each player increases without battles decreasing it. These neutral planets have different sizes and these sizes don't appear to follow any trend or be restricted to any range (besides the value range for the parameters of the genetic algorithm's individuals). Moreover, the position of these neutral planets are different in every map and they don't follow any trend as well.

CONCLUSION AND FUTURE WORK

In this paper we have introduced a simple procedural map generator for a RTS game that is capable of generating balanced maps for two player games in an acceptable execution time. An example of a map generated by this algorithm is shown in figure 5. Despite this algorithm generates fully playable maps, there are several improvements that could be made to this generator. For example, the generator uses a simple genetic algorithm which can be tuned more exhaustively to obtain a better performance (changing the breeding pipeline or/and researching more optimal breeding operators). The maps generated by this algorithm are not symmetrical and some planets should be overlapped, so the map generation function could be improved avoiding overlapped planets and forcing these to be symmetrical. When evolving the maps, the fitness function is obtained from only one game (execution) with the same two players. A better fitness function should obtain its data from several games with different players playing each of these games. This way the map would not been balanced only for a kind of player, but a group of them. Although this is a simple map generator for a simple RTS game, it can be easily scaled to work with more complex games and situations.

ACKNOWLEDGEMENTS

This work is partially supported by Spanish MICINN under project ANYSELF (TIN2011-28627-C04-01), and by Junta de Andalucía under project P10-TIC-6083 (DNEMESIS).

REFERENCES

Blizzard Entertainment 1998, *Starcraft*, Blizzard Entertainment.

Frade, M., de Vega, F. & Cotta, C. 2010*a*, Evolution of artificial terrains for video games based on accessibility, *in* C. Di Chio, S. Cagnoni, C. Cotta, M. Ebner, A. Ekrt, A. Esparcia-Alcazar, C.-K. Goh, J. Merelo, F. Neri, M. Preu, J. Togelius & G. Yannakakis, eds, 'Applications of Evolutionary Computation', Vol. 6024 of *Lecture Notes in Computer Science*, Springer Berlin / Heidelberg, pp. 90–99.

Frade, M., de Vega, F. F. & Cotta, C. 2008, Modelling video games' landscapes by means of genetic terrain programming - a new approach for improving users' experience, *in* M. Giacobini et al., eds, 'Applications of Evolutionary Computing', Vol. 4974 of *Lecture Notes in Computer Science*, Springer, pp. 485–490.

Frade, M., de Vega, F. F. & Cotta, C. 2009, 'Breeding terrains with genetic terrain programming: The evolution of terrain generators', *International Journal of Computer Games Technology* **2009**.

Frade, M., de Vega, F. F. & Cotta, C. 2010*b*, Evolution of artificial terrains for video games based on obstacles edge length, *in* 'IEEE Congress on Evolutionary Computation', IEEE, pp. 1–8.

Gearbox Software 2009, *Borderlands*, 2K Games.

Mahlmann, T., Togelius, J. & Yannakakis, G. N. 2012, Spicing up map generation, *in* C. D. Chio et al., eds, 'EvoApplications', Vol. 7248 of *Lecture Notes in Computer Science*, Springer, pp. 224–233.

Maxis 2008, *Spore*, Electronic Arts.

Mojang 2011, *Minecraft*, Mojang.

Togelius, J., De Nardi, R. & Lucas, S. 2007, Towards automatic personalised content creation for racing games, *in* 'Computational Intelligence and Games, 2007. CIG 2007. IEEE Symposium on', pp. 252–259.

Togelius, J., Preuss, M., Beume, N., Wessing, S., Hagelback, J. & Yannakakis, G. 2010, Multiobjective exploration of the starcraft map space, *in* 'Computational Intelligence and Games (CIG), 2010 IEEE Symposium on', pp. 265 –272.

Togelius, J., Preuss, M. & Yannakakis, G. N. 2010, Towards multiobjective procedural map generation, *in* 'Proceedings of the 2010 Workshop on Procedural Content Generation in Games', pp. 3:1–3:8.

Togelius, J., Yannakakis, G. N., Stanley, K. O. & Browne, C. 2011, 'Search-based procedural content generation: A taxonomy and survey', *IEEE Transactions on Computational Intelligence and AI in Games* **3**(3), 172–186.

White, D. 2012, 'Software review: the ecj toolkit', *Genetic Programming and Evolvable Machines* **13**, 65–67. 10.1007/s10710-011-9148-z.

BEHAVIOURAL AI

SELF-ORGANIZING SQUAD AND CROWD FORMATION FOR EMERGENCY EVACUATIONS IN SERIOUS GAMES

César García-García, Victor Larios-Rosillo and Hervé Luga
CUCEA-Universidad de Guadalajara
Periférico Norte 799, Guadalajara
Mexico
cesar.garcia@cucea.udg.mx

KEYWORDS
AI, Serious Gaming, Behaviour Modeling

ABSTRACT

This paper presents a novel approach to organize agents into groups or squads for evacuation simulation. Previous work containing personality profiles is used to simulate social interaction and leadership roles, as well as comparing this approach to centralized, partly-autonomous and randomized formations in terms of survivability of squad members during a simulated evacuation.

INTRODUCTION

One of the first steps in crowd simulation is grouping or clustering the agents into a crowd, this could be achieved by using self-organizing agents that autonomously create and join groups inside the crowds. In this specific context we will be addressing the problem of creating evacuation units in such a manner that the chances of survival of individual agents is increased by the configuration of the evacuation units.

PROBLEM DESCRIPTION

An evacuation is by itself one of the most chaotic social phenomena, but it is not pure chaos. There are underlying mechanisms pulling the strings of the crowd: patterns emerge, lanes begin to form, leaders rise up to the task and groups organize around them. Identifying a leader to follow and freely associating into groups according to individual personality is a key phenomena in evacuation simulation.
Several approaches to self-organization exist, including Centralization, where unit composition is explicitly declared; Reactivity (Parunak 1997), where agents lack a representation of themselves and follow only stimulus-response rules; Cooperative Information (Hoile et al. 2002), where the concept of environment and resources is present; Adaptive Multi-Agent System (AMAS) (Gleizes et al. 1999) where agent behaviour is assumed to be always cooperative; and Role-based Multi-Agent Systems (Nair et al. 2002, Serugendo et al. 2005), where agents have roles and create organizational structures. The AMAS approach (Gleizes et al. 1999) is particularly relevant to the problem of creating self-organizing evacuation squads due to the nature of its meta-rules: 1) information must be unambiguous, 2) information has to be useful and 3) agent actions lead to improving the wellbeing of the group.

APPROACH

We propose a new method for self-organized unit formation that takes into account the personality of each unit member. Our approach includes agent negotiation aimed to: Identify individuals that could be able to lead an evacuation unit and Join a leader-based unit that improves individual survivability.

Measuring leadership

Leader-referenced formations have been used to maintain agent formations in simulated battlefields (Balch and Arkin 1999). In this case, leadership is discretionally assigned beforehand. (Murgatroyd et al. 2011) uses a hierarchical agent structure to simulate a medieval army marching into the battle of Manzikert. Leadership lies in the Emperor agent and is assigned beforehand. Previous works have used personality traits to explain perceived leadership profiles (Durupinar et al. 2008, Karnes et al. 1984) and to create a leadership skill inventory (Edmunds 1998) using specific 16PF Cattell (1957) personality factors such as Reasoning (B), Liveliness (F), Rule-Consciousness (G), Openness to Change (Q1) and Self-Reliance (Q2). When translated to the 16PF-5 (Cattell et al. 1993) personality factors, we represent the same relationships as in (1) where l stands for leadership and is a function of Extraversion (Ex), Anxiety (An), Tough-Mindedness (TM), Independence (I) and Self-Control (SC). The highest the leadership score l, the better fit is an agent to lead others.

$$l = Ex - An + TM + I + SC \quad (1)$$

We originally considered including training as part of the leadership score l, much as like in the work of

(Pelechano and Badler 2006) where leadership is measured as a better knowledge of the premised to be evacuated, in the end we opted against it because we consider the aspects of knowing a place and the ability to lead to be independent.

Measuring survivability

Our previous work (García-García et al. 2012) shows that personality is not a significant factor to survivability during an emergency evacuation. From this fact we are assuming that the only other factors are awareness and training, both represented in our behaviour model simply as training. We will assume individual survivability to be equivalent to training (2) and unit survivability as the mean training for the unit (3).

$$S_{member} = training \tag{2}$$

$$S_{unit} = \frac{\sum_{n=1}^{N} n_{training}}{n} \tag{3}$$

Leadership negotiation behaviour

As we want to move from the predefined unit leaders, we need to provide our agents with a mechanism to identify and "elect" leaders according to the previously defined leadership score (1). At first, each agent announces to the environment its aspiration to become a unit leader. Each potential leader then compares itself to each other and decides to decline or stay based on a probability function described in (4) where t is the agent training; t' is the other agent's training; TM stands for tough-mindedness; In stands for independence; and An stands for anxiety.

$$P_{(declination)} = 0.5 - (t - t') - TM - In + An \tag{4}$$

It is understood that an anxious agent ($An+$) is more likely to decline, and a more self-reliant ($TM+$, $In+$) agent is less likely to decline, where a clear difference in training is the key factor for leadership declination. This process repeats itself until only a set number of potential leaders remain within the environment. If, for any reason, communication with a leader is lost the unit becomes acephalous: each member then has the choice of joining an existing unit (declining in favor of the unit's leader) or competing to form a new unit.

Unit joining behaviour

Once the best leaders have been identified each agent has to individually choose which leader to follow, this joining their unit. This is done by applying the k-medoid clustering algorithm called Partitioning Around Medoids (Theodoridis and Koutroumbas 2006) using the previously defined leaders as the medoids, with a 5-dimension euclidean distance metric based on the personality profile of each agent vs. each leader.

EXPERIMENT AND RESULTS

We begin by generating a population of 100 untrained civilian agents normally distributed around a previously obtained personality profile (García-García et al. 2012). We proceed to calculate the leadership score for each agent and select the top five as unit leaders.
We then proceed with the following unit formation experiments:

- Unit formation by randomly assigning members to five predefined leaders.
- Round-robin assignment of members to five predefined leaders and their units.

Next, we use the same civilian profiles running the leadership negotiation behaviour, where agents "vote" for the best leaders until a set number of votes have been cast and the most voted agents are selected as unit leaders. This allows for an unlimited negotiated selection leaders and round-robin assignment of members to their units.
Figure 1 illustrates the resulting organization around selected leaders.

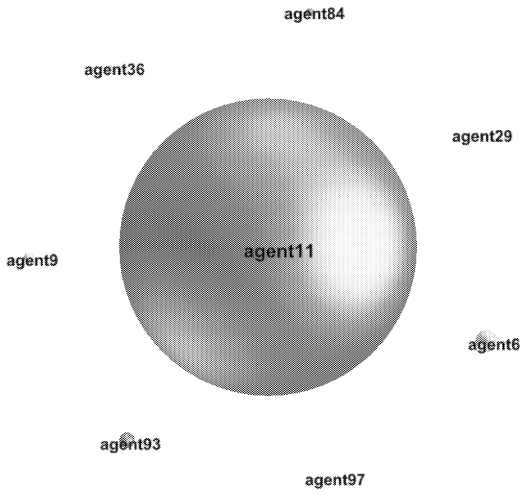


Figure 1: Agents organized around their selected leader by personality-based affinity.

As shown in Figure 2, the autonomous selection of leaders and units shows an improvement over the purely randomized and hand-picked leader approaches.

CONCLUSIONS

Our results show that awareness and training are effectively a key to individual survivability during emer-

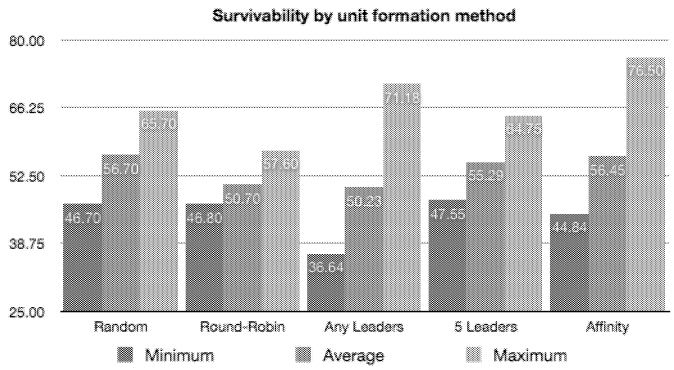


Figure 2: Comparison between all presented methods

gency evacuations. The generated leader identification and unit formation behaviours will contribute to have more realistic crowds where individuals associate and gravitate each other based on their personality and not only in simulated physical forces. As our proposed measurement of leadership is not related to training in any way, there are scenarios in which a disruptive character is better fit to lead and thus selected as a unit leader. In the real world, it is highly advisable to identify civilians whose personality profile suggest a leadership role and inviting these individuals to get involved in the self-defense efforts to become trained civilians.

ACKNOWLEDGEMENTS

This project is funded by CONACYT and COECYT-JAL grants. The DVRMedia2 framework is a cooperative effort between Universidad de Guadalajara and Institut de Recherche en Informatique de Toulouse (IRIT) trough Le Programme de Coopération Post-Gradués (PCP) Franco-Mexicain. Special thanks to Intel Guadalajara, to the IBM Smarter Cities Exploration Center and to Unidad Estatal de Protección Civil y Bomberos Jalisco for providing data, training and access to their facilities.

REFERENCES

Balch T. and Arkin R.C., 1999. *Behavior-based Formation Control for Multi-robot Teams. IEEE Transactions on Robotics and Automation.*

Cattell R., 1957. *Personality and motivation structure and measurement.* World Book.

Cattell R.B.; Cattell A.; and Cattell H., 1993. *16PF Fifth Edition Questionnaire.* Institute for Personality and Ability Testing, Champaign, IL, 5th ed.

Durupinar F.; Allbeck J.; Pelechano N.; and Badler N.I., 2008. *Creating Crowd Variation with OCEAN Personality Model.* In L. Padgham; Parkes; Müller; and Parsons (Eds.), *Proc. of 7th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2008).* International Foundation for Autonomous Agents and Multiagent Systems, Estoril, Portugal, 12–16.

Edmunds A.L., 1998. *Content, concurrent and construct validity of the leadership skills inventory. Roeper Review.*

García-García C.; Larios-Rosillo V.; and Luga H., 2012. *Intelligent Computer Graphics 2012*, Springer, *Studies in Computational Intelligence*, vol. 441, chap. Agent behaviour modeling using personality profile characterization for emergency evacuation serious games.

Gleizes M.P.; Camps V.; and Glize P., 1999. *A THEORY OF EMERGENT COMPUTATION BASED ON COOPERATIVE SELF- ORGANIZATION FOR ADAPTIVE ARTIFICIAL SYSTEMS.* In *4th European Congress of Systems Science.* Valencia, Spain.

Hoile C.; Wang F.; Bonsma E.; and Marrow P., 2002. *Core Specification and Experiments in DIET: A Decentralised Ecosystem-inspired Mobile Agent System.* In *AAMAS 2002.* ACM, ACM, Bologna, Italy.

Karnes F.A.; Chauvin J.C.; and Trant T.J., 1984. *Leadership profiles as determined by the HSPQ of students identified as intellectually gifted. Roeper Review*, 7, no. 1, 48–48.

Murgatroyd P.; Craenen B.; Theodoropoulos G.; Gaffney V.; and Haldon J., 2011. *Medieval Military Logistics: An Agent-based Simulation of a Byzantine Army on the March.* In *Computational and Mathematical Organization Theory.* 1–19.

Nair R.; Tambe M.; Marsella S.; and Pynadath D.V., 2002. *Model for Team Formation for Reformation in Multiagent Systems.* In *AAAI Technical Report WS-02-04.* AAAI.

Parunak H.V.D., 1997. *Go to the Ant: Engineering Principles from Natural Multi-Agent Systems.* In *Annals of Operations Research.* vol. 75, 69–101.

Pelechano N. and Badler N.I., 2006. *Modeling Crowd and Trained Leader Behavior during Building Evacuation. IEEE Computer Graphics and Applications.*

Serugendo G.D.M.; Gleizes M.P.; and Karageorgos A., 2005. *Self-organization in multi-agent systems. The Knowledge Engineering Review*, 20, no. 2, 165–189.

Theodoridis S. and Koutroumbas K., 2006. *Pattern recognition.* 3rd. ed.

CHAMELEON: A LEARNING VIRTUAL BOT FOR BELIEVABLE BEHAVIORS IN VIDEO GAME

Fabien Tencé[1,2], Laurent Gaubert[1], Pierre De Loor[1] and Cédric Buche[1]
[1] UEB – ENIB – LAB-STICC
[2] Virtualys
Brest – France
{tence,gaubert,deloor,buche}@enib.fr

KEYWORDS
Bayesian inference, believability, behavioral simulation, experiments.

ABSTRACT

The believability of a virtual world can be increased by improving the behavior of the characters in it. Considering literature, we choose a model developed by Le Hy to generate the behaviors by imitation. The model uses probability distributions to find which decision to choose depending on the sensors. Then actions are chosen depending on the sensors and the decision. The core idea of the model is promising but we propose to enhance the expressiveness of the model and the associated learning algorithm. We hope the model will be able to generate more believable behaviors and learn them with minimal *a priori* knowledge. We first revamp the organization of the sensors and motors by semantic refinement and add a focus mechanism in order to improve the believability. To achieve believability, we integrate an algorithm to learn the topology of the environment. Then, we revamp the learning algorithm to be able to learn much more parameters and with greater precision at the cost of its time of convergence.

INTRODUCTION

Presence is one of the main goals of virtual worlds. It consists in making the users of those environments feel like they are in the simulation. To do so, there must be many rich interactions between the user and the virtual world. One option is to populate the simulation with entities which exhibit a *believable behavior* (Bates 1994). The main difficulty is that in virtual worlds, some entities may have unpredictable behaviors, like for instance users' avatars. It may lead to unexpected situations which artificial entities may not be able to handle correctly, exhibiting unadapted behaviors. Regarding this kind of problems, it is necessary for the entities to be able to dynamically evolve. We propose to provide our entities with *imitation learning* abilities in order to adopt a real player's behavior.

Video game companies want the players to be immersed in the simulation. To achieve this, they create rich and complex virtual worlds. Thanks to these kind of work, researchers can avoid some technical difficulties (rendering, physics, networking, etc.) by using such games and then focus on the entities to be studied (Mac Namee 2004). Furthermore, and as a feed-back effect, since video games are made for human users and often popular, they offer a real challenge for the entities to be believable.

In this article we first give an overview of the kind of models which drive entities' behaviors in video games. Then we focus on a model from Le Hy, which seems to fit our need for both believability and imitation learning. Then we propose some modifications in order to make the virtual character more believable and capable of learning almost all the parameters of the model.To conclude, we give explanations about how we would like to evaluate the believability of our model.

LITERATURE

In the video game industry, most of the games are scripted to have a storyline, models must be flexible and readable enough to be adjusted by game designers. Animation is also important so models have to handle low level details. As a consequence, very simple models are still used such as finite state machines (FSMs). They are easy to read and give a good control on how the character will behave. The main drawbacks are that it is hard to code complex behaviors, difficult to maintain and they need a lot of time to be parametrized correctly.

An alternative to FSM are behavior trees. They give the same control over the character's behavior but it is a lot easier to code and maintain. However, they are still not widely used is the video game industry and, as FSM, they lack of expressiveness for complex behaviors.

Bayesian-based approaches

Recently, some Bayesian-based approaches have been used to control characters in video games. The model described in (Gorman et al. 2006*a*) tries to apply to a video game a Bayesian model of imitation previously developed in (Rao et al. 2004). The believability of this model has been studied in (Gorman et al. 2006*b*). The model, however, does not seem to offer easy generalization. But it shows that a Bayesian approach fits to our need for believability and compatibility with imitation learning.

A specific Bayesian-based model (Le Hy et al. 2004) has been developed for characters in video games. The advantage of this model is that it is quite easy to modify so that the character acts as wanted. It is also possible to learn the parameters of the model by imitation. The believability of the behaviors has not been carefully evaluated but some preliminary tests show that it may be comparable to models from industry. This last model will be our base for future work because it is quite close to FSM which give good results. It has also more expressiveness, partly because it uses probabilities, and has an imitation learning algorithm. We believe that choosing a model, more complex than what is used in industry but less complex than cognitive or multi-agents model, is a good mean to generate believable behaviors. Indeed, using that approach, it is possible to generate complex behaviors, still being able to understand and modify the internal parameters to achieve the best believability.

Le Hy's model

In Le Hy's model, the agent has sensors named $S = (S_0, .., S_n)$. They give information on internal and environment's state like for instance the character's inventory and the position of another character. Agent's movements are driven by motors named $M = (M_0, ..., M_p)$ which can be rotation, jump commands and so on. Both sensors and motors take discrete values. In order to simulate the character's behavior, the notion of decision has been introduced, the associated variable is named D and may have different values like searching for an object or fleeing.

The value of D^t, where t is the time, is chosen according to the value of the sensors, following the probability distribution $P(D^t|S)$, and to the previous decision, following the probability distribution $P(D^t|D^{t-1})$. Thus the value of D is chosen following the probability distribution $P(D^t|S^tD^{t-1})$, the decision model, computed using the two previous distributions. As S is the conjunction of n variables, Le Hy introduces the notion of *inverse programming* to reduce the complexity: $P(D^t|S^t)$ is computed using $P(S_i^t|D^t)$ (and not $P(D^t|S_i^t)$!) as they are assumed to be independent, which is a strong assumption.

Once the value of D^t is chosen randomly following the distribution $P(D^t|S^tD^{t-1})$, the model must decide which motor command should be activated. The value of each motor command is chosen following the distribution $P(M_i^t|S^tD^t)$, the motor model. Again, to reduce the complexity, Le Hy introduce the notion of *fusion by enhanced coherence*. Each command is computed separately then they are combined using the formula $P(M_i^t|S^tD^t) = \frac{1}{Z}\prod_j P(M_i^t|S_j^tD^t)$ where $1/Z$ is a normalization factor.

Thus the model, which can be categorized as an input-output hidden Markov model, is composed of three types of parameters whose relations are summarized in figure 1:

- $P(D^t|D^{t-1})$
- $P(S_i^t|D^t)$
- $P(M_i^t|S_j^tD^t)$

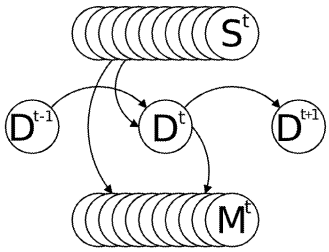


Figure 1: Summary of the influences between model's variables (Le Hy et al. 2004).

Those parameters can be specified by hand or learned by imitation. Results seems to be better in term of believability and performance with learned parameters. The imitation is done by observation of the virtual representation of the player, his avatar also named the demonstrator. By monitoring at each time step the values for S and M for this demonstrator, it is possible to update the value of the parameters. The learning algorithm developed by Le Hy is based on (Florez-Larrahondo 2005) but only updates the decision model, the parameters $P(D^t|D^{t-1})$ and $P(S_i^t|D^t)$.

This algorithm, a modified version of the incremental Baum-Welch, updates at each time step the parameters with the following formula:

$$P_n(D^t|D^{t-1}) = \frac{1}{Z}\Big(P_{n-1}(D^t|D^{t-1}) + \Delta P_{n-1}(D^t|D^{t-1})\Big)$$

$$P_n(S^t|D^t) = \frac{1}{Z}\Big(P_{n-1}(S^t|D^t) + \Delta P_{n-1}(S^t|D^t)\Big)$$

$1/Z$ is a normalization factor. The Δ are computed using the motor model, $P(M_i^t|S_j^t)$, and the actual values of the decision model, $P_{n-1}(D^t|D^{t-1})$ and $P_{n-1}(S^t|D^t)$.

Limits

As believability is based on the feeling of an observer, we have to examine the behaviors produced by the model to see its real advantages and drawbacks. Based on those observations, we can propose some modifications to compensate potential flaws.

The main flaw of agents using Le Hy's model is the way they move in the environment. We noticed that its movements were not very smooth and gave an overall feeling of non-humanness. Moreover, it often chose motor commands which seem not to satisfy any goal. How the agent act is very important because it gives the first impression to observers.

On top of the movements, the paths the agent uses to go from one point of the environment to another do not look like the ones a player would take. This problem does not comes from the model itself but from the representation it uses for the environment. Indeed, the agent uses navigation points placed by the designers of the environment which may not represent well how players prefer to use the environment.

The implementation of Le Hy's model has not enough sensors to exhibit the whole range of behavior a layer would exhibit. Le Hy's model is flexible enough for new sensors to be added. However increasing the number of perception make the learning more difficult because it increases the parameters. A compromise should be found to give enough information to the agent without adding unnecessary complexity to the model.

CHAMELEON: LE HY'S MODEL ENHANCEMENTS

Despite the problems raised, the implementation of Le Hy's model showed that the decision sequencing using Bayesian programming is quite efficient and simple to settle. The concept of decision makes the behavior easy to adjust by modifying the probabilities value. This is why we decided to follows Le Hy's general idea. First, a decision is chosen knowing the previous one and some information about the environment. Then actions are done depending on the decision and the environment. However some limitations were raised during our different experiments so we modified the way the decision and the actions are chosen and how the sensors and actions are linked to the decision in several ways.

Improvements On Le Hy's Decision Model

The first change we made is to apply a semantic refinement on sensors, splitting them in two types: high level ones (random variables H_i) and low level ones (random variables L_j). As decisions represent general behaviors (attacking, fleeing, etc.) the agent does not need a very accurate information about the environment to make its choice. However, in order to accomplish any action according to the chosen decision, the agent needs much more accurate values (to aim or to run avoiding walls for instance). This should increase the amount of available information for the agent without increasing too much the parameters.

The second change is to regroup actions into three types: motion, interaction and reflexive actions. A motion action (random variable M) gathers all the motor commands needed for the agent to be able to move in the environment. In our case it is a combination of five motors: pitch, yaw, run, lateral movement and jump. The difference with Le Hy's model is that we do not assume that all the motor commands are independent. As a consequence, the agent has a better control of how it moves at the cost of increasing the number of parameters in the model. An interaction action (random variable I) regroups all the motor commands needed for the agent to interact with other players or objects in the environment. In our case there is only one motor: shooting or not. The last kind of action is the reflexive action (random variable R): it models any action that the agent applies to itself. In our case there is only one motor: changing the current weapon. The main difference between reflexive actions and the other actions is that they depend on the current decision and the high level sensors whereas the other actions depend on the current decision and the low level sensors. This models the fact that the agent does not need very accurate information about its surroundings in order to achieve reflexive actions.

The third and last change to the model is to replace the mechanisms to reduce the complexity resulting from the number of possible values for sensors. The first mechanism used by Le Hy is the inverse programming where $P(D^t|D^{t-1}S^t)$ is computed using $P(S_i^t|D^t)$, assuming that all sensors are independent knowing the decision. This assumption may be wrong depending on the chosen sensors, and moreover, the more sensors used the higher the chances the assumption may be wrong. The second mechanism is the fusion by enhanced coherence. This technique suffers some simple problems: it makes use of probability distributions, but handle them in total opposition with their natural properties. The main aim of this technique is, in the end, to consider a weighted sum of probability distributions, which is easily and rig-

orously achieved with the sum of random variables over a random index. Therefore, we propose to use instead a mechanism where the agent focus on one high level sensor and one low level sensor. This focus mechanism use two distributions $P(G^t|H^t)$ and $P(J^t|D^tL^t)$ where the random variables G and J gives respectively the index of the high level sensor the agent focus on and the index of the low level sensor the agent focus on. H^t and L^t are respectively the conjunction of all the high and low level sensors. As a result, we must simplify the expression of the two distribution because they may take far too many values to be tractable. We propose to express the distributions as follows:

$$\mathbb{P}\left(G^t = i \,\middle|\, H^t\right) = \frac{\theta_i\left(H_i^t\right)}{\sum_n \theta_n\left(H_n^t\right)} \quad (1)$$

$$\mathbb{P}\left(J^t = j \,\middle|\, D^t, L^t, K^t\right) = \frac{\lambda\left(D^t, L_j^t\right)}{\sum_m \lambda\left(D^t, L_m^t\right)} \quad (2)$$

The higher the values of θ and λ, the more likely the agent will focus on the associated sensor. This greatly reduces the number of parameters still giving the agent a mechanism to focus only on one sensor.

The model works in the following way (see figure 2):

- Pick an index i of a high level sensor, using (1)
- Pick a decision using $P(D^t|D^{t-1}H_i)$
- Pick an index j of a low level sensor, using (2)
- Pick a motion action using $P(M^t|D^tL_j)$
- Pick an interaction action using $P(I^t|D^tL_j)$
- Pick a reflexive action using $P(R^t|D^tH_i)$

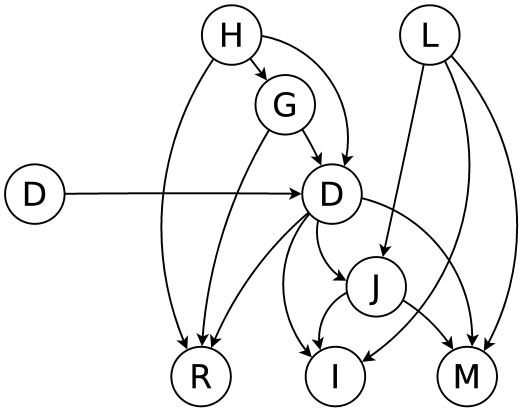


Figure 2: Summary of the relation between the random variable of the model.

Learning the Environment

To be able to learn the parameters of the model, the learning algorithm first needs sensors and motors which represent best how players interact with the game. While most of them are defined by hand, creating a representation of the environment is a quite tedious task. As a consequence, we looked for techniques to learn by imitation such a representation. Few works had been done on this subject, but an interesting technique (Thurau et al. 2004) tries to learn the movement in a virtual world by imitation. It uses a algorithm named neural gas to learn navigation points from the positions of a player's avatar. This algorithm has the advantage of being on-line: it learns at each observation of the player. Therefore, the neural gas can be used and learned at the same time, the character evolving during the game.

As this technique seems promising, we tried to apply a Growing Neural Gas (GNG) on the observed positions. The GNG (Fritzke 1995) is a graph model which is able to achieve incremental learning. Each node has position (x,y,z) in the environment and has a cumulated error which measures how well the node represents its surroundings. Each edge links two nodes and has an age which gives the time it was last activated. This algorithm needs to be omniscient, because the position of the imitated player (the demonstrator) has to be known at any time. The principle of the GNG is to modify its graph, adding or removing nodes and edges and changing the nodes' position for each input of the demonstrator's position. For each input the closest and the second closest nodes are picked. An edge is created between those nodes and the closest node's error is increased. Then the closest nodes and its neighbours are attracted toward the input. All the closest node's edges' age is increased by 1 and too old edges are deleted.
We trained 2 GNG on 2 different maps. The first one is a simple map, called Training Day, it is small and flat which is interesting to visualize the data in 2 dimensions. The second one, called Mixer, is much bigger and complex with stairs, elevators and slopes which is interesting to see if the GNG behaves well in a real three dimensional environment. The results are shown in figure 3 (a) for the simple map and in figure 3 (b) for the complex map.

In order to study the quality of the learned topology, we first chose to compare the GNG's nodes with the navigation point placed manually by the map creators. Of course, we do not want the GNG to fit exactly those points but it gives a first evaluation of the learned representation. In our case we know those navigation points but our goal is that they become not longer necessary for a character which evolves in a new environment. Figure 4 shows both the navigation points and the GNG's

nodes. As we can see, the two representations look alike which indicates that the model is very efficient in learning the shape of the map. However, there are zones where the GNG's nodes are more concentrated than the navigation points and other where they are less concentrated. We cannot tell now if it is a good behavior or not as we should evaluate an agent using this representation to see if it navigates well. Even in the less concentrated zones, the nodes are always close enough to be seen from one to another, so it should not be a problem.

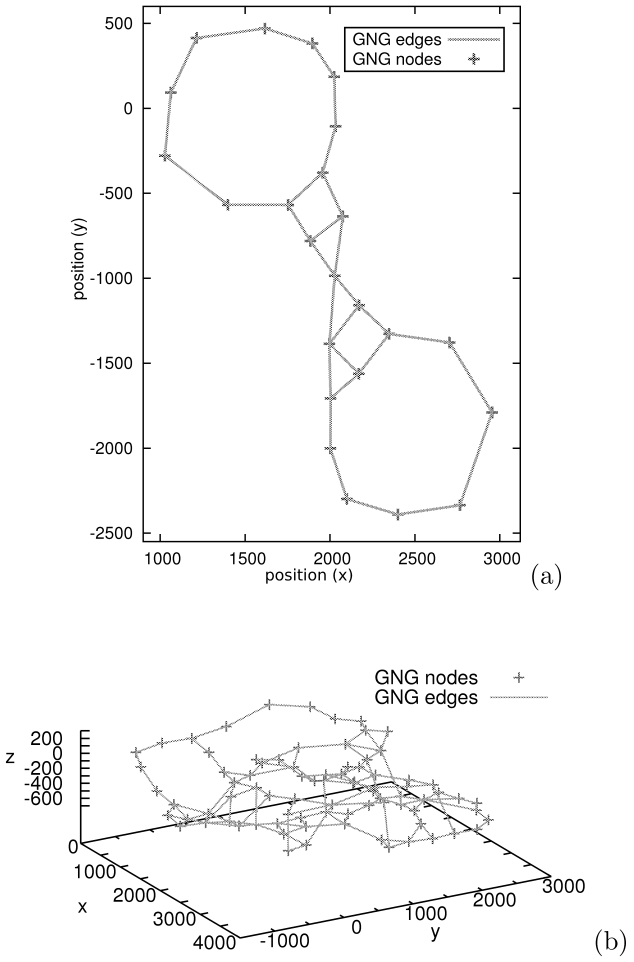


Figure 3: Result of a growing neural gas learned from a player for a simple map, top view (a) and for a complex map (b).

As the attraction applied to the nodes for each input is constant, the GNG is not converging to a stable state. This is a desired behavior, allowing the GNG to adapt to a variation in the use of the map: if the teacher suddenly uses a part of the map which he/she has not yet explored, the GNG will be able to learn this new part even if the GNG has been learning for a long time.

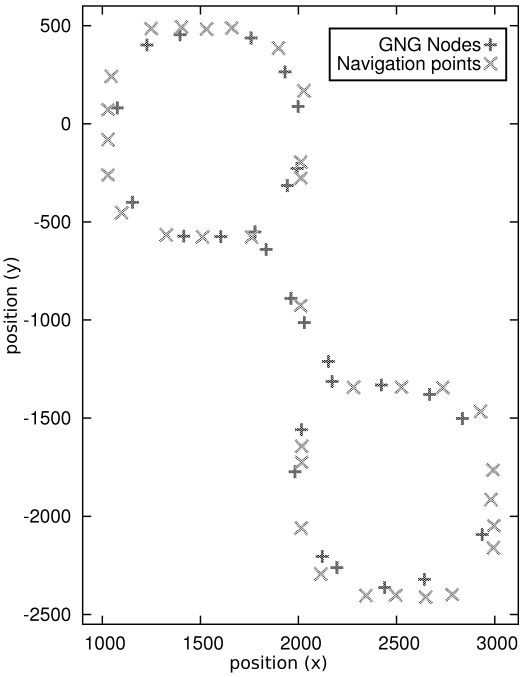


Figure 4: Comparison of nodes learned by the growing neural gas with the navigation points placed manually by the game developers.

Learning the Parameters Of The Decision Model

In order to learn the parameters of the model, Le Hy uses a modified version of Florez-Larrahondo's incremental Baum-Welch algorithm. We prefer not to use this algorithm because it has to approximate very roughly the probabilities computed by the backward procedure. In our case such probabilities give the chances of taking a certain decision at t knowing what the demonstrator did at $t + 1...T$. This information should not be lost (for instance, if the demonstrator is looking for something specific, the learning algorithm cannot know what it is picked up). As a consequence, it seems wiser to learn on a whole sequence of observations (from time 0 to time T) instead of using an incremental version. We choose to begin the sequence when the demonstrator's avatar appears in the environment and end the sequence when the avatar dies.

We also want to extend the learning to all the distributions and not only the ones used to pick a decision. In order to do that, we apply a EM algorithm to our model to optimize the parameters. For each sequence of observations of a demonstrator, our algorithm gives a local maximum of the likelihood function. The quality of this local maximum depends on the initialization of the distributions and some modification we can do to the parameters during each maximization step. At the time we write this article, we use a random initialization and a technique to make the distributions less uniform

using the formula $a = \frac{1}{Z}a^n$, a being a probability, $\frac{1}{Z}$ a normalization factor and $n > 1$.

θ and λ are the only parameters that are not learned. Our first attempts to find solutions that optimize the likelihood function using a gradient method did not give good results. On the contrary, specifying them by hand and making the model learn on demonstrators gives quite good results, the agent being able to move in the world and shoot at enemies (although an accurate shooting procedure is hard to learn and it may be enhanced by specific heuristics).

CONCLUSION

Virtual worlds, like for example video games, need believable characters for users to feel in the environment. For virtual characters, actual solutions in research focus on intelligence with cognitive or multi-agents approach. In the industry, the goal is mainly believability but the models often generate too simple behaviors.

Le Hy's model seems to fill the gap between the two approaches, focusing on believability but trying to produce quite complex behaviors. To have a first look on the results of the model we implemented it. The result is very flexible, behaviors can be easily specified. However agents are not as believable as what is done in the industry. This difference can come from the assumption done in the model and because characters designers in the industry spend a lot more time to specify the behavior rules.

We first propose some modifications to Le Hy's model by rearranging the sensors and the motors and by adding a focusing mechanism. We hope these changes will enable the model to produce more complex behaviors. We then introduce the growing neural gas to learn the topography of the environment. Finally we apply an EM algorithm to learn how find a decision and to carry it out.

The learning algorithm still need some enhancements to be complete. θ and λ are not learned yet and techniques to find maxima close to the global maximum are still to be found. Then, the next step is to evaluate our work by gathering a pool of player to let the agent learn from their behavior. When the learning is done, we will try to assess the believability of our agent. Believability is very difficult to evaluate because it is subjective. We will do a test based on (Turing 1950; Gorman et al. 2006*b*), using humans as judges.

REFERENCES

Bates J 1994 The Role of Emotion in Believable Agents *Communications of the ACM* **37**(7), 122–125.

Florez-Larrahondo G 2005 Incremental learning of discrete hidden Markov models PhD thesis Mississippi State University.

Fritzke B 1995 A growing neural gas network learns topologies *in* 'Advances in Neural Information Processing Systems 7' MIT Press pp. 625–632.

Gorman B, Thurau C, Bauckhage C and Humphrys M 2006*a* Bayesian imitation of human behavior in interactive computer games *in* 'Proceedings of the 18th International Conference on Pattern Recognition (ICPR'06)-Volume 01' IEEE Computer Society Washington, DC, USA pp. 1244–1247.

Gorman B, Thurau C, Bauckhage C and Humphrys M 2006*b* Believability testing and bayesian imitation in interactive computer games *in* 'From Animals to Animats 9' Vol. 4095 Springer pp. 655–666.

Le Hy R, Arrigoni A, Bessiere P and Lebeltel O 2004 Teaching bayesian behaviours to video game characters *Robotics and Autonomous Systems* **47**, 177–185.

Mac Namee B 2004 Proactive Persistent Agents: Using Situational Intelligence to Create Support characters in Character-Centric Computer Games PhD thesis.

Rao R, Shon A and Meltzoff A 2004 A bayesian model of imitation in infants and robots *in* C. L Nehaniv and K Dautenhahn, eds, 'Imitation and Social Learning in Robots, Humans, and Animals' Cambridge University Press pp. 217–248.

Thurau C, Bauckhage C and Sagerer G 2004 Learning human-like movement behavior for computer games *in* 'Proceedings of the 8th International Conference on the Simulation of Adaptive Behavior (SAB'04)'.

Turing A 1950 Computing machinery and intelligence *Mind* **59**(236), 433–460.

BIOGRAPHIES

FABIEN TENCÉ received his Ph.D. in computer science in 2011 concerning "Probabilistic Behaviour Model and Imitation Learning Algorithm for Believable Characters in Video Games". Now, he is working at Virtualys.

LAURENT GAUBERT is an associate professor of mathematics at the Brest National Engineering School (ENIB), which belongs to the European University of Britain (UEB). He is a member of the LAB-STICC IHSEV team, member of the European Center for Virtual Reality (CERV). His research concerns the simulation of adaptive behaviors, the synchronization of periodic coupled systems, the study of links between individuals and populations in abstract models and applied models, such as data produced by DNA chips.

PIERRE DE LOOR is a professor at the Lab-STICC laboratory (UMR-CNRS). He is the head of the IHSEV team since 2012 and the head of the CERV. He wrote his doctoral thesis in Automatic, Signal and Software engineering (1996) and his Accreditation to Direct Research (HDR) in Computer Science (2006). He's interrested on the link beetween artificial intelligence, cognitive science and virtual reality. He's also interrested on phenomenology, epistemology and links between art and science.

CÉDRIC BUCHE is an associate professor at the ENIB/UEB Member of the IHSEV team of the LAB-STICC, member of the CERV. He wrote an Accreditation to Direct Research (HDR) concerning the simulation of adaptive behaviors (2011). He is the leader of the CHAMELEON project.

HALL-OF-FAME COMPETITIVE COEVOLUTIONARY ALGORITHMS FOR OPTIMIZING OPPONENT STRATEGIES IN A NEW GAME

Mariela Nogueira[1], Juan M. Gálvez, Carlos Cotta, Antonio J. Fernández-Leiva
[1]University of Computers Science in Cuba, University of Málaga in Spain
Email: [1]mnogueira@uci.cu, {ccottap|afdez}@lcc.uma.es

KEYWORDS
Coevolution, RTS game, Genetic Algorithm, Artificial Intelligence

ABSTRACT

This paper describes the application of competitive coevolution as a mechanism of self learning in a two-player real time strategy (RTS) game. The paper presents this (war) RTS game, developed by the authors as an open-source tool, and describes its (built-in) coevolutionary engine developed to find winning strategies. This engine applies a competitive coevolutionary algorithm that uses the concept of *Hall-of-Fame* to establish a long-term memory that is employed in the evaluation process. An empirical analysis of the performance of two different versions of this coevolutionary algorithm is conducted in the context of the RTS game. Moreover, the paper also shows, by an example, the potential of this coevolutionary engine as a prediction tool by inferring the initial conditions (i.e. army configuration) under which a battle has been executed when we know the final result.

INTRODUCTION

Artificial Intelligence (AI) implementation represents a challenge in game development, and a deficient game AI surely decreases the satisfaction of the player. Game AI has been traditionally coded manually which causes well-known problems such as loss of reality, sensation of artificial stupidity, and predictable behaviors, among others; to overcome them a number of advanced AI techniques have recently been proposed in the literature, and coevolution, a biologically inspired technique based on the interaction between different species, represents one of the most interesting techniques.

Coevolution has been shown to be successful on a number of applications but it also has a number of drawbacks Ficici and Bucci (2007), Watson and Pollack (2001). In consequence several alternatives have already been proposed to alleviate the inherent problems of the process; for instance, the integrative techniques such as the so-called Pareto Coevolution De Jong and Pollack (2004), Jong (2007), de Jong (2004), Ficici and Pollack (2000) that proffers the integration between coevolution with Evolutionary Multi-Objective Optimization; also the application of evolutionary game theory to the study of coevolutionary algorithms Ficici and Pollack (2000), Ficici (2004), Wiegand et al. (2002). However the primary remedy to cope with the pathologies of coevolution consists of proposing new forms of evaluating individuals during coevolution Rosin and Belew (1995), and memorization of a number of successful solutions to guide the search is one of the most employed. Following this idea, Rosin and Belew (1997) already proposed the use of a *Hall-of-Fame* based mechanism as an archive method and, since then, there have been similar proposals such as those described in de Jong (2004), Jaskowski and Krawiec (2010), Jong (2007), Yang et al. (2009).

Coevolutionary systems are usually based on two kinds of interactions: one in which different species interact symbiotically (i.e. the cooperative approach) and other in which species compete among them (i.e. the competitive approach). In the cooperation, an individual is decomposed in different components that evolve simultaneously and the fitness depends on the interaction between these components; in the competition, an individual competes with other individuals for the fitness value and, if necessary, will increase its fitness at the expense of its counterparts, decreasing their fitness, as happens in a competitive game Johnson et al. (2004). This latter approach resembles an arms race in which the improvement of some individuals causes the improvement in others, and viceversa. Moreover, coevolutionary arms races can in principle lead to the discovery of complex, original behaviors that can make the game dramatically more interesting and challenging Dziuk and Miikkulainen (2011). Several experiments have shown significants results in the application of coevolutionary models in competitive environments to study the emergence of strategies in simple and complex games Angeline and Pollack (1993), Sims (1994), Johnson et al. (2004), Avery et al. (2008a), Smith et al. (2010).

This paper deals with the application of competitive coevolution (CC) as a self learning mechanism in RTS games. As a first contribution, the paper presents *robotWars*, a RTS war game available as open-source, that has been developed by the authors of this paper to be used as a tool to do research on CC methods. As a second contribution, the paper describes the evolutionary engine built-in inside *robotWars*; this engine consists of a competitive coevolutionary algorithm that uses in the evaluation process a Hall-of-Fame that acts as a long-term memory mechanism; from this engine, we devise two different variants that are used to automatically produce game strategies to govern the behavior of the army in the game that can also beat its opponent counterpart. Finally, we show (through an example) that the coevolutionary en-

gine built-in into *robotWars* can also be used as a predictive model to guess what has happened in already finished battles and whose conditions were not known at the time.

GAME DESCRIPTION

This section is devoted to *robotWars*[1], our arena for cooperative evolution; *robotWars* is a test environment that allows two virtual players (i.e. game AIs) to compete in a 3 dimensional scenario of a RTS war game, and thus it is not a standard game itself in the sense that no human players intervene interactively during the game; however it is a perfect scenario to test the goodness and efficacy of (possibly hand-coded) strategies to control the game AI and where the human player can influence the game by setting its initial conditions. To provide a good 3D simulation, the game was programmed with Ogre3D.

In *robotWars*, two different armies (each of them controlled by a virtual player - i.e. a game AI) fight in a map (i.e. the scenario) that contains multiple obstacles and has limited dimensions. Each army consists of a number of different units (including one general), in the initial configuration both armies have identical number of units, and the army that first wipes out the rival general is considered the winner. The game is executed in turns and in each turn one army is allowed to send, separately, one order to each of its constituent units (i.e. each unit can received a different order to be executed).

Virtual players are internally coded as a 4 dimension matrix where the first dimension has 7 different values corresponding to the *type of unit*(i.e. general, infantry, chariot, air force, antiaircraft, artillery, cavalry), the second dimension to *objective closeness* (i.e. a binary value: 0 if the unit is closer to the enemy general than to its mate general, and 1 otherwise), the third dimension to *numeric advantage* (i.e. are there, in a nearby space[2], more mate units than rival units? A binary answer:yes/no), and the fourth dimension to *health* (i.e. an amount that indicates the health level of the army as high, medium or low, respect to the numbers of living units). Each position of the matrix acts as a gen and stores one of the following 5 actions: *attack*[3], *advance*, *recede*, *crowd together*, or *no operation*.

The whole matrix represents a strategy that controls, deterministically, the behavior of an army during the game. For a specific type of unit there are 12 possible different states (i.e. $2 \times 2 \times 3$, all the possible value combinations considering the last three dimensions of the matrix), and basically, in a specific turn of the game each unit of the army will execute the action stored in the state in which the unit perceives that it is. Note that all the units (irrespective of their type) are managed by the same controller, and in any instant of the game, the team behavior will be the result of summing up all the action taken by each of its constituent units. Note however that this does not mean all the units execute the same action because the action to be executed by a unit will depend on its particular situation in the match and its specific category.

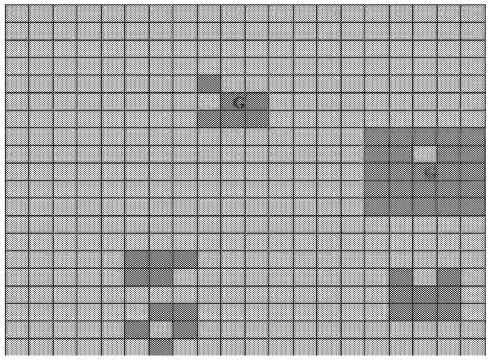


Figure 1: A simple map without obstacles

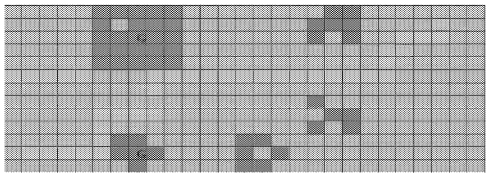


Figure 2: A map with obstacles

robotWars includes two interesting tools: the *Battle generator*, that allows a human to generate different scenarios for the game by changing the topologies and structures of the armies; two types of army formation are initially allowed: *testudo or tortoise* formation, in which the units are placed as a homogeneous block and the general is positioned in its center, and *band or guerrilla* formation, in which the units are dispersed in the battle scenario; Figure 1 shows a very simple scenario with an army initially positioned in tortoise formation (red) and the other in guerrilla formation (blue); also the green cell represents a walkable area, and the letter G points out the general position. Figure 2 shows a map with obstacles which are symbolized by gray cells. The other tool, the *Battle simulator*, receives as entry two different game AIs and a map (that includes not only the orographic features of the terrain but also the topologies and structure of the two armies) over which a game will be run and provides a deterministic simulation of a war between the two armies. There are a number of configurable parameters such as maximum number of iterations or initial army formation for example. Figure 3 shows a screenshot of a battle in the game.

HALL-OF-FAME BASED COMPETITIVE COEVOLUTIONARY ALGORITHM (HofCC)

Our objective is to apply competitive coevolution techniques to automate the generation of victorious strategies for the game. According to the strategies encoding explain in the

[1] `http://www.lcc.uma.es/~afdez/robotWars`

[2] Each unit u placed in a position (x_u, y_u, z_u) in the map has its own *visual range* that embraces any position (x, y, z) that is placed to a maximum distance τ of its own position; this means that each unit only receives information that is perceived inside its visual range.

[3] This action starts the combat against the nearest enemy.

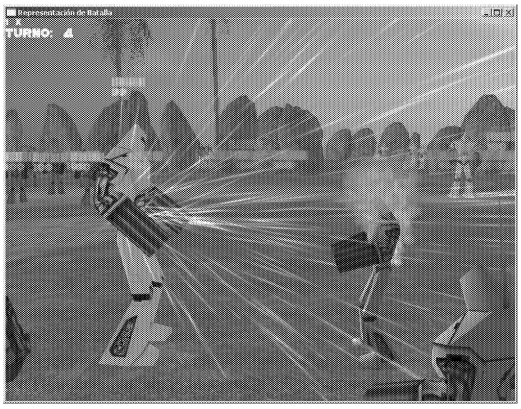


Figure 3: Screenshot of *robotWars* game

previous section, the search space is $5^{7\times2\times2\times3} = 5^{84}$, which is really huge, and precisely this forms the basis for bases the employment of metaheuristic techniques.

Our RTS game incorporates a competitive coevolutionary algorithm that uses a long-term memory to keep the winning strategies found in each coevolutionary step; more specifically, each army (i.e. player) maintains its own Hall-of-Fame (HoF) Rosin and Belew (1995) in which its own winning strategies (with respect to the set of winning strategies of its opponent) found in each coevolutionay step will be saved. More recent studies continue exploring in the use of HoF Lichocki et al. (2008), Avery et al. (2008b) as a long-term memory mechanism, and although each author can propose their own implementation, the original idea of the method remains. Algorithm 1 shows schema of the said algorithm with our implementation of HoF. A strategy is considered *winning* if it achieves a certain score (see below) when it deals with each of the strategies belonging to the set of winning strategies of its opponent (i.e. the rival Hall-of-Fame). The initial objective is to find a winning strategy of player 1 with respect to player 2 (i.e. the initial opponent) so that the HoF of player 2 is initially loaded with some strategies (randomly or manually initialized: line 2). Then a standard evolutionary process tries to find a strategy for player 1 that can be considered as victorious (lines 7-13). A strategy is considered winning if its fitness value is above a certain threshold value ϕ (line 14) that enables the tuning of the selective pressure of the search process by considering higher/lower quality strategies; in case of success (line 14), this strategy is added to the HoF of player 1 (line 16) and the process is initiated again but with the players' roles interchanged (line 17); otherwise (i.e. no winning strategy is found) the search process is restarted again. If after a number of coevolutionary steps no winning strategy is found the search is considered to be stagnated and the coevolution finishes (see while condition in line 4). At the end of the whole process we obtain as a result two sets of winning strategies associated respectively to each of the players.

With respect to the evaluation of candidates for a specific player p (where $p \in \{player\ 1, player\ 2\}$), the fitness of a specific strategy is computed by addressing it with each of the (winning) strategies in the Hall-of-Fame of its opponent player (note that the Hall-of-Fame acts as a long-term memory by keeping all the winners found previously and all of them are also used in the evaluation process). Given a specific strategy s its fitness is computed as follows:

$$fitness(s) = \frac{\sum_{j=1}^{k}(p_j^s + extras_s(j)}{k} \quad (1)$$

where $k \in \mathbb{N}$ is the cardinality of the opponent HoF (i.e. number of opponent winning strategies found so far), $p_j^s \in \Re$ returns ϕ points if strategy s beats strategy h_j belonging to the opponent HoF (i.e. victorious case), $\frac{\phi}{2}$ in case of a draw, and 0 if h_j wins to strategy s; Also:

$$extras_s(j) = c - nturn_{sj} + c * \Delta health_{sj} + c * \Delta Alive_{sj} \quad (2)$$

where $c \in \mathbb{N}$ is a constant, $nTurn_{sj} \in \mathbb{N}$ is the number of turns spent on the game to achieve a victory of s over its opponent h_j (0 in case of draw or defeat), $\Delta health_{sj} \in \mathbb{N}$ is the difference between the final health of the army (i.e. sum of the health of all its living units) controlled by strategy s at the end of the match and the corresponding health of the enemy army, and $\Delta Alive_{sj}$ is its equivalent with respect to the number of living units at the end of the combat. This fitness definition was formulated based on our game experience, and it values the victory above any other result.

Algorithm 1: HofCC()

```
1  nCoev ← 0; A ← player_1; B ← player_2; φ ← thresholdvalue;
2  HoF_A ← ∅; HoF_B ← InitialOpponent();
3  pop ← Evaluate(HOF_B); // Evaluate initial population
4  while nCoev < MaxCoevolutions ∧ NOT(timeout) do
5      pop ←RandomSolutions(); // pop randomly initialized
6      i ← 0;
7      while (i < MaxGenerations) ∧ (fitness(best(pop)) < φ) do
8          parents ←Select (pop);
9          childs ← Recombine (parents, p_X);
10         childs ← Mutate (childs, p_M);
11         pop ← Replace(childs);
12         pop ← Evaluate(HOF_B);
13     end while
14     if fitness(best(pop)) ≥ φ then //winner found!
15         nCoevolutions ← 0; // start new search
16         HoF_A ← HoF_A ∪ {best(pop)}
17         temp ← A; A ← B; B ← temp; // interchange players'
               roles
18     else
19         nCoev ← nCoev + 1; // continue search
20     end if
21 end while
```

EXPERIMENTS AND ANALYSIS

For the experiments we have considered two maps (with and without obstacles respectively), 4 possible combinations of the two types of formation (i.e. tortoise vs. guerrilla, tortoise vs. tortoise, guerrilla vs. tortoise, and guerrilla vs. guerrilla), and two possible initial predefined strategies, *offensive* or *random*, to be charged initially in the HoF of player 2

-see Line 2 in Algorithm 1). The *offensive* strategy was preprogrammed and basically executes the action of attacking when the unit perceives that it is in an advantageous state; otherwise, it executes the action of going back (i.e. retreating). And *random* strategy selects a random action (from the five predefined actions) for each cell of the action matrix. In summary, 16 distinct battle scenarios were considered.
In addition, two variants of the HofCC algorithm were used: the first version of the algorithm (version A) uses the maximum size for the matrix of strategies as described in Section and thus reserves memory for storing the behavior (i.e. the action to be done) of all types of military units under all types of possible situations (i.e. states), even though not all types of units form part of the army; the consequence is that genetic operators can be acting on genes that will not have an influence on the fitness evaluation. The second version (version B) is an optimized version that considers, for the genetic search, only those units that are actually in the army, so the genetic operators act on the 100% of the useful genes that might influence the results, and surely expedites the search process (this is important to alleviate in certain form the computation cost associated to both coevolution and the Hall-of-Fame based evaluation process).
We only show the graphs for four battles, the other results are collected and analyzed in a table at the end of this section. Note that we use the colors red and blue to distinguish the armies.

CONFIGURATION OF THE EXPERIMENTS

For the experiments we have used a steady-sate genetic algorithm (GA - see Lines 7-13 in Algorithm 1) with the aim of finding a winning strategy with respect to a set of strategies (stored in the HoF of the opponent) that were considered winning in previous stages of the coevolutionary algorithm; this GA employed binary tournament for selection, Bernoulli crossover, it-flip mutation, elitist replacement, $MaxCoevolution = 15$, $Maxgenerations = 300$, population size was set to 100, and standard values for crossover and mutation probabilities were used ($p_X = .9$ and $p_M = 1/nb$ respectively where nb is the number of genes); and $\phi = 2000$ so that a strategy that defeats all the strategies in the HoF of its opponent is surely considered as victorious, although others can also be. The choice of these values is due to a previous analysis of the mean fitness reached by individuals in the competitions. We also set the constant c in Equation (2) to 200, a representative value of the range of scores that were obtained from the battle simulator after executing a number of games.
Note that a strategy is wining only within the proposed battle scenario and that we do not expect to find a global optimal strategy for a generic scenario. Our analysis has been guided by the following indicators:

- *Best fitness in each coevolution*: displays the best strategies and possible stagnation produced in the coevolutionary steps.
- *Number of generations used in each coevolution*: represents the number of iterations that are required to find a winning strategy. It helps to identify whether the problem's difficulty increases as best solutions are obtained, or if it remains stable.
- *Equality percentage of genes*: it allows us to know whether the new strategies adopted by the search algorithm suffer a continuous readjustment and massive variation of their genes, or if instead they are the result of changes produced on small sets of genes in the strategies found in previous coevolutionary steps.

In what follows, due to limitations of space, we only show the results obtained in 4 scenarios and will focus mainly on the first indicator mentioned. In any case, at the end of the section a summary of the results is provided.

RESULTS OF VERSION A

Figures 4 and 5 show the performance of the best fitness achieved by each army during the coevolution, using as initial enemy strategy the *offensive* (Fig. 4) and the *random* (Fig. 5). The battles were fought on the map shown in Figure 1. The red bar represents the best solutions found by an army (red army) and blue bars identifies the other army. The black line located on the 2000 fitness points represents the threshold at which the solutions are classified as winning strategies. When a solution exceeds this line, the next bar has a different color, which means that this army found a winning strategy, and the other army now tries to find, in the next coevolutionary step, another strategy that can beat the winners (including the current champion) encountered by the rival in the previous steps of the coevolutionary algorithm. A sequence of candidates represented by bars of equal color (where none of them exceeds the threshold that identifies a winning solution) means that the search does not progress as no winning strategy can be found.

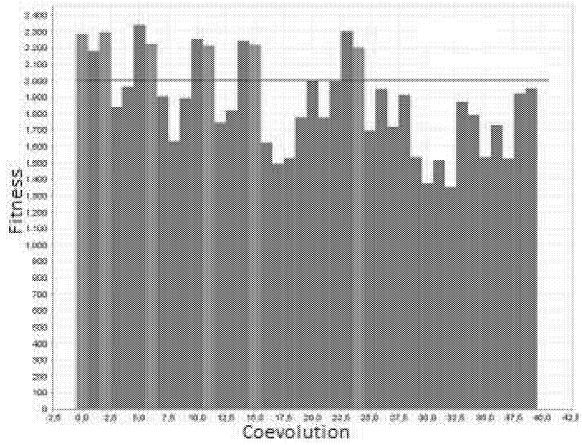


Figure 4: Version A: Offensive intial strategy; Best fitness

In these scenarios, the red army has less difficulties for finding winning strategies whereas the blue army, on the other

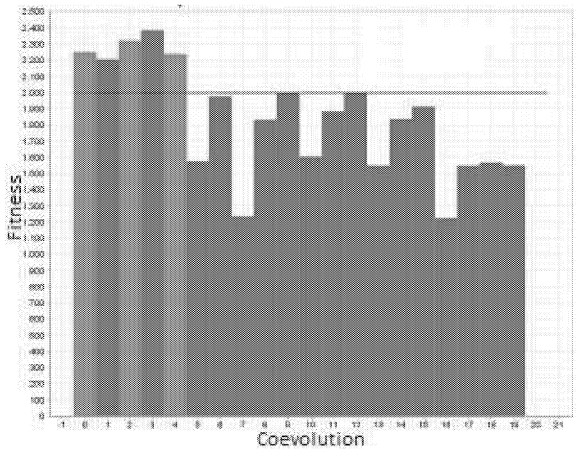


Figure 5: Version A: Random intial strategy; Best fitness

hand, has serious problems and requires an increasing number of coevolutionary steps to confront the new strategies proposed by the red army. In principle, observing the ease with which the red army finds a winning strategy, we can decide that the *tortoise* formation can provide advantages over the *guerrilla* formation, in this type of map (i.e. open and unobstructed).

Regarding the percentage of equality between the solutions, the analyzed data (graphs not shown for reasons of space) showed that solutions never exceed 30% of equality. In general, we observe that the solutions of the blue army made more drastic genetic modifications which clearly can be interpreted as an attempt to explore alternative regions in the search space, although finally this was a frustrated attempt.

Regarding the average number of game turns needed in each battle, an analysis was done for each scenario. As shown in Equation (2), the fitness function rewards those solutions which require fewer game turns for beating the opponent, and those which also generate fewer casualties in relation to the number of dead units of its rival army. The result of this analysis was that the new strategies found require fewer turns to defeat the opponent. In the two initial strategy options (*offensive* and *random*) both armies tend to minimize the number of turns used for each new strategy encountered.

We also analyzed the difference between the number of units which survive at the end of battles in each of the armies. We note that in the case of setting an *offensive* initial strategy all these differences were positive, which means that all winning solutions have more survivors than their corresponding opponent strategies. This is a logical result because all winning solutions tend to reach victory by means of the destruction of all units of the rival, although the reality is that the victory can be obtained without necessarily destroying all the opponent's units (for example, by simply destroying its general). In this way it is perfectly possible and correct to find solutions with negative differences favorable to the opponent. In fact, we have seen that despite all differences being positive, in general this difference tends to decrease in the new strategic solutions, which means that as strategies are refined it becomes more difficult to find solutions to overcome the rival without affecting the integrity of the units, forcing these solutions to sacrifice an increasingly larger number of units to achieve the desired military victory. This decrease in the difference between surviving units denotes some convergence towards more strategic results, where solutions are more balanced between them. This balance tends to make zero the difference between the number of casualties suffered by both armies during the battles.

On the other hand, for the case of setting a *random* initial strategy, we observed that both armies found strategic solutions which generated a greater number of casualties on the opponent in every coevolutionary step. And this continues being a positive outcome because it is interpreted as the finding of more and more effective strategies that achieve victory generating further damage to the opponent.

RESULTS OF VERSION B

For analyzing the results obtained with version B of the algorithm, Figures 6 and 7 show the behavior of the best fitness found in each coevolution for both types of initial enemy strategy (*offensive* and *random*), respectively. These battles took place in the same scenario that we saw previously for Figures 4 and 5.

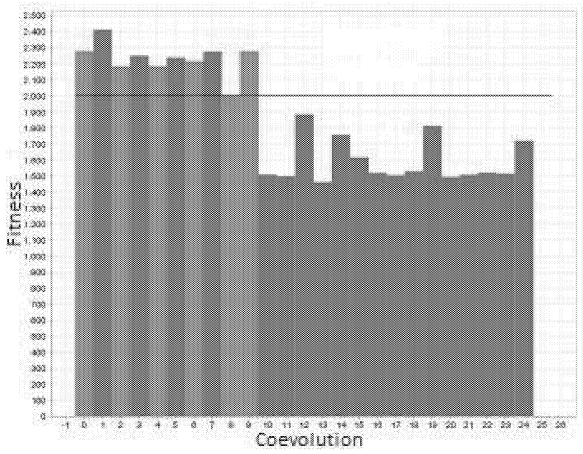


Figure 6: Version B: Offensive intial strategy; Best fitness

In conclusion we do not see any detail that shows an improvement in the succession of coevolution returned by this version of the algorithm. Specifically, there is a very similar sequence of coevolution between the results obtained by both versions (A and B), and they coincide even in the army which reached stagnation (the blue one) because it cannot overcome the opponent strategy.

In the case of equality percentage among the solutions nothing unusual was appreciated, although the percentages in the version B were lower due to the more drastic changes which were made in the genotype of the solutions found; this might even be interpreted as a greater difficulty for algorithm B to find winning solutions.

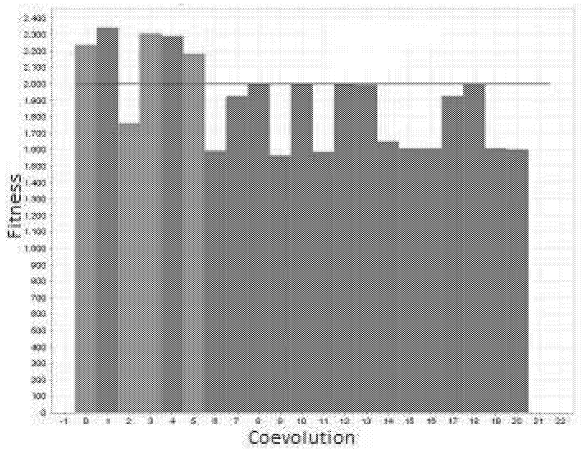


Figure 7: Version B: Random intial strategy; Best fitness

SUMMARY OF THE RESULTS FROM BOTH VERSIONS

In summary, using version B we note that the armies are able to find more refined solutions, and sometimes solutions that tip the scales of victory over the army which in version A was considered a loser are found. In general, version B produced a fitness evolution very similar to that of version A, although sometimes it adds an extra level of coevolution which provides a new solution that was not found by A. This shows that the way to model the solutions in version B where the genes are only destined to represent the behavior of military units which actually participate in the battle, makes genetic operators more accurate and effective, which likely results in a greater ability to explore solutions that version A could not reach. This conclusion was somewhat expected, because the adaptation of the strategy matrix to the context of the battle reduces the complexity of the problem, and thus reduces the number of possible solutions, discarding those solutions that really were useless, and this issue has a significant impact on the effectiveness of version B with respect to that shown by version A.

Tables 1 and 2 summarize the results obtained by both algorithms when they were executed on the 16 scenarios initially considered in the experiments (see Section). Each battle is identified by the initial opponent strategy adopted (first row), the armies formation (columns 1 and 2), the map where the battle was executed (i.e. Map 1 - without obstacles -, or Map 2 - with obstacles), and the winner army (red, or blue).

Observe that the blue army was affected the most because it stagnated (i.e. did not find better solutions) in 24 out of 32 cases. If we examine the results with respect to the initial enemy strategies, we observe that the blue army failed in 14 out of 16 cases when the initial strategy is *offensive*; however, when the initial strategy is *random* the results are more than compensated. This indicates that the selection of the initial opponent has a strong influence on the results.

From an orography point of view, and considering the two battles between *tortoise* and *guerrilla* formations, the *tortoise* formation tends to dominate in Map 1, which is completely free of obstacles, whereas in Map 2, which has a more complex topology, the *guerrilla* formation is the dominant; this result forms the idea that this type of formation are more suitable for fighting in areas with more obstacles.

Table 1: Algorithm HofCC (Version B): Summary of the results

Formation		Initial Offensive		Initial Random	
Blue army	Red army	Map 1	Map2	Map 1	Map2
Tortoise	Tortoise	Red	Red	Blue	Blue
Tortoise	Guerrilla	Red	Red	Red	Blue
Guerrilla	Tortoise	Blue	Red	Blue	Red
Guerrilla	Guerrilla	Red	Red	Red	Red

Table 2: Algorithm HofCC (Version B): Summary of the results

Formation		Initial Offensive		Initial Random	
Blue army	Red army	Map 1	Map2	Map 1	Map2
Tortoise	Tortoise	Red	Red	Blue	Red
Tortoise	Guerrilla	Red	Red	Red	Red
Guerrilla	Tortoise	Blue	Red	Blue	Red
Guerrilla	Guerrilla	Red	Red	Red	Red

SIMULATION OF THE BATTLE OF CARRHAE

This section is devoted to showing, by an example, how the HofCC engine can also be used to argue (or guess) what could have happened initially to obtain a certain final result. Here we consider the Battle of Carrhae, an historical battle that took place in the year 53 B.C. near the town of Carrhae; it was an important battle between the Parthian Empire and the Roman Republic. The Parthian Spahbod Surena decisively defeated a Roman invasion force led by Marcus Licinius Crassus. It was the first of the battles between the Roman and Persian empires, and one of the most crushing defeats in Roman history. Table 3 shows some data (i.e. statistics) associated with this battle[4].

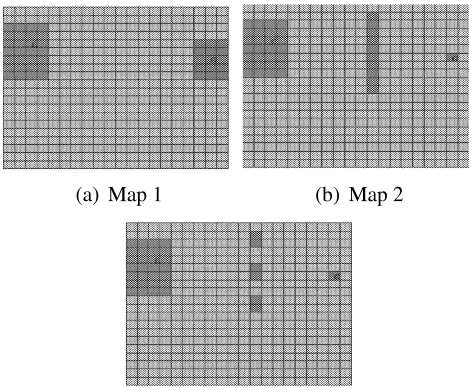
We have modeled three virtual scenarios for the battle where we considered a map free of obstacles as we assume that these should have weight in the strategic decisions. The red army represents the Roman army whereas the blue army represents the Parthian army (less numerous than the Roman infantry but more powerful). The real types of soldiers employed in the Battle (i.e. legionaries, cavalry, horse archers, etc) were adapted to our game by an intuitive mapping of the

[4] `en.wikipedia.org/wiki/Battle_of_Carrhae`, April 2012.

Table 3: Dates of The Battle of Carrhae

Roman Empire	**Parthian Empire**
Strength	**Strength**
35,000 legionaries 4,000 cavalry, 4,000 light infantry	9,000 horse archers, 1,000 cataphracts
Casualties and losses	**Casualties and losses**
20,000 dead, 10,000 captured, 10,000 escaped	$\sim$ 100

type of units built-in inside *robotWars*, and for the simulation we respected the ratio between the number of units belonging to both armies. Version B of HofCC algorithm was applied in three different scenarios for the battle (shown in Figure 8).



(a) Map 1 (b) Map 2

(c) Map 3

Figure 8: Battle of Carrhae: Scenarios for the simulation

In the first simulated battle (Map 1 in Figure 8) the romans have infantry units with displacement velocity and shot scope set by default, whereas as the Parthian army makes mainly use of cavalry units. Figures 9 and 10 show the results, in terms of best fitness value found in each evaluation, found by considering first an *offensive* initial opponent strategy, and second a *random* one, respectively. Observe in Figure 9 that the fitness evolution resembles the historical reality.
In the case of *random* strategy, the Romans find a victory in the second turn of coevolution, perhaps as a consequence of an imperfection in the strategic defensive formation of the Parthian cavalry that has its general completely unprotected. After this, the blue army takes total control of the battle and the Romans are not able to find a winning strategy, and here, the fitness evolution resembles the historical reality.
In the second map we reduced the number of Parthian units (from 14 to 10) given as result a ratio of 3 to 1 (in favor to the Romans); even in these circumstances the Parthians won the battle as the fitness evolution was quite similar to the results shown previously. In the simulation of this game, the

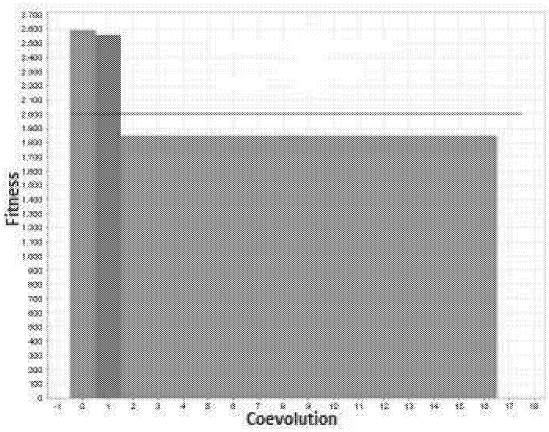


Figure 9: Best fitness with *ofensive* initial strategy

Parthian army executed approaches towards the Roman units followed by quick attacks carried out from a certain distance to avoid a direct confrontation with the Roman infantry units. This scenario represents with certain accuracy the historical events that really happened.
In the third scenario, we imposed a ratio of 5 to 1 by reducing by even more the number of Parthian units. However, in this case, the Parthian cavalry was insufficient for beating the Roman army that easily got the victory.

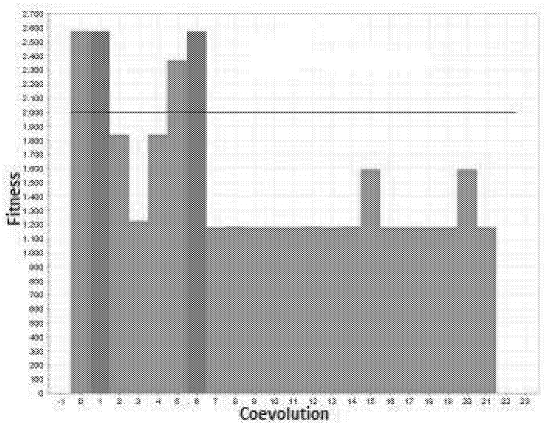


Figure 10: Best fitness with *random* initial strategy

CONCLUSIONS AND FUTURE WORK

This paper has dealt with the employment of competitive coevolution (CC) in RTS games. We have first presented a new RTS game called *robotWars* that can be used as a platform for researching with CC. We have also described a coevolutionary algorithm that has been implemented in the game described and that has experimentally shown promising results for searching winning strategies in this game.
We have also shown, through an example, the ability of the coevolutionary algorithm to reason about historical events according to known results. In the future we plan to investigate if this ability can be extrapolated to show a predic-

tive capacity, even in other areas outside the game arena that consider complex scenarios (with emergent properties for instance).

We have also identified some weaknesses on which we must work to improve our coevolutionary model. For this, let us begin by exploring the use of the Hall-of-Fame, which has the great advantage that it very easy to implement, also ensures an evaluation mechanism that takes into account previous champions (i.e. a memory mechanism), and favors the transitivity between the strategies of the hall. However, these benefits can be dulled by the efficiency of the algorithm which is negatively affected when the size of the hall grows.

ACKNOWLEDGEMENTS

This work is partially supported by Spanish MICINN under project ANYSELF (TIN2011-28627-C04-01), and by Junta de Andalucía under project P10-TIC-6083 (DNEMESIS).

REFERENCES

Angeline P. and Pollack J., 1993. *Competitive environments evolve better solutions for complex tasks*. In *Proceedings of the Fifth International Conference on Genetic Algorithms*. San Mateo, California, 264–270.

Avery P.; Greenwood G.; and Michalewicz Z., 2008a. *Coevolving strategic intelligence*. In *Evolutionary Computation. CEC 2008.(IEEE World Congress on Computational Intelligence). IEEE Congress on*. IEEE, 3523–3530.

Avery P. et al., 2008b. *Coevolving a computer player for resource allocation games: using the game of Tempo as a test space.* Ph.D. thesis, School of Computer Science University of Adelaide.

de Jong E., 2004. *Towards a bounded Pareto-Coevolution archive*. In *Evolutionary Computation, 2004. CEC2004. Congress on*. IEEE, vol. 2, 2341–2348.

De Jong E.D. and Pollack J.B., 2004. *Ideal Evaluation from Coevolution*. *Evol Comput*, 12, no. 2, 159–192. ISSN 1063-6560.

Dziuk A. and Miikkulainen R., 2011. *Creating intelligent agents through shaping of coevolution*. In *Evolutionary Computation (CEC), 2011 IEEE Congress on*. IEEE, 1077–1083.

Ficici S., 2004. *Solution concepts in coevolutionary algorithms*. Ph.D. thesis, Brandeis University.

Ficici S. and Pollack J., 2000. *A game-theoretic approach to the simple coevolutionary algorithm*. In *Parallel Problem Solving from Nature PPSN VI*. 467–476.

Ficici S.G. and Bucci A., 2007. *Advanced tutorial on coevolution*. In *Proceedings of the 2007 GECCO conference companion on Genetic and evolutionary computation*. ACM, New York, NY, USA. ISBN 978-1-59593-698-1, 3172–3204.

Jaskowski W. and Krawiec K., 2010. *Coordinate System Archive for Coevolution*. In *Evolutionary Computation (CEC), 2010 IEEE Congress on*. IEEE, 1–10.

Johnson R.; Melich M.; Michalewicz Z.; and Schmidt M., 2004. *Coevolutionary tempo game*. In *Evolutionary Computation. CEC'04. Congress on*. vol. 2, 1610–1617.

Jong E., 2007. *A monotonic archive for pareto-coevolution*. *Evolutionary Computation*, 15, no. 1, 61–93.

Lichocki P. et al., 2008. *Evolving players for a real-time strategy game using gene expression programming*. Master's thesis, Poznan Universtity of Technology.

Rosin C. and Belew R., 1995. *Methods for competitive coevolution: Finding opponents worth beating*. In *Proceedings of the Sixth International Conference on Genetic Algorithms*. San Francisco, CA, 373–380.

Rosin C. and Belew R., 1997. *New methods for competitive coevolution*. *Evolutionary Computation*, 5, no. 1, 1–29.

Sims K., 1994. *Evolving 3D morphology and behavior by competition*. *Artificial life*, 1, no. 4, 353–372.

Smith G.; Avery P.; Houmanfar R.; and Louis S., 2010. *Using co-evolved RTS opponents to teach spatial tactics*. In *Computational Intelligence and Games (CIG), 2010 IEEE Symposium on*. Citeseer, 146–153.

Watson R. and Pollack J., 2001. *Coevolutionary dynamics in a minimal substrate*. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2001)*. Morgan Kaufmann, 702–709.

Wiegand R.; Liles W.; and De Jong K., 2002. *Analyzing cooperative coevolution with evolutionary game theory*. In *Evolutionary Computation. CEC'02. Proceedings of the 2002 Congress on*. vol. 2, 1600–1605.

Yang L.; Huang H.; and Yang X., 2009. *A Simple Coevolution Archive Based on Bidirectional Dimension Extraction*. In *Artificial Intelligence and Computational Intelligence, 2009. AICI'09. International Conference on*. IEEE, vol. 1, 596–600.

Best- First Search with Genetic Algorithm for Space Optimization in Pathfinding Problems

Ulysses O. Santos and
Alex F. V. Machado
Departamento de Computação,
Instituto Federal de Educação,
Ciência e Tecnologia do Sudeste de
Minas Gerais,
Rio Pomba, MG - Brasil
Email: ulyssesdvlp@gmail.com and
alexcataguases@hotmail.com

Esteban W. G. Clua

Instituto de Computação,
Universidade Federal Fluminense,
Niterói, RJ – Brasil
Email: esteban@ic.uff.br

Keywords: Pathfinding, Best-First-Search, Genetic Algorithm, optimization, electronic games.

ABSTRACT

This paper presents a novel method to optimize the process of finding paths using a model based on Genetic Algorithms and Best-First-Search for real time systems, such as video games and virtual reality environments. The proposed solution uses obstacle pattern detection based at online training system to guarantee the memory economy. The architecture named Patterned based Pathfinding with Genetic Algorithm (PPGA) uses a learning technique in order to create an agent adapted to the environment that is able to optimize the search for paths even in the presence of obstacles. We demonstrate that the PPGA architecture performs better than classic A* and Best-First-Search algorithms in patterned environment.

INTRODUCTION

The search of the paths between two points is a fundamental problem for the area of electronic games, but it is also relevant for other areas (Stodola and Mazal 2010). In this field the most used methods are A * (Russell and Norvig) and Best-First Search (Russell and Norvig). Due to the demands of a real-time processing and hardware limitations, these traditional methods can cause problems in performance. Therefore many optimizations are studied for the same (Dechter and Pearl 1985; Leigh et al. 2007; Korf 1985).

The traditional algorithm A* uses f(n)=g(n)+h(n) heuristic, where g(n) is the cost to reach the node n, h(n) is the estimated cost for reach the goal node from the node n. This calculation can be performed by Manhattan distance. From this algorithm we can find the optimal solution. As can be seen in Fig. 1, the nodes accessed by search are stored in the open and closed lists, marked by "O" and "C" respectively, and for each one, a value of “f” is calculated. During the entire processing of the algorithm the lists are maintained in memory. By applying these methods in multi-agents systems such as RPGs and strategy games in real time, the memory usage can be very high.

The algorithm IDA * (Korf 1985) is designed for the efficient use of memory. It is based on depth-first search using the same heuristic as algorithm A*, ensuring that finds the shortest path and decreasing the number of nodes expanded impacting the amount of memory required by the search. Although requiring less memory than the A*, it takes longer to find the path.

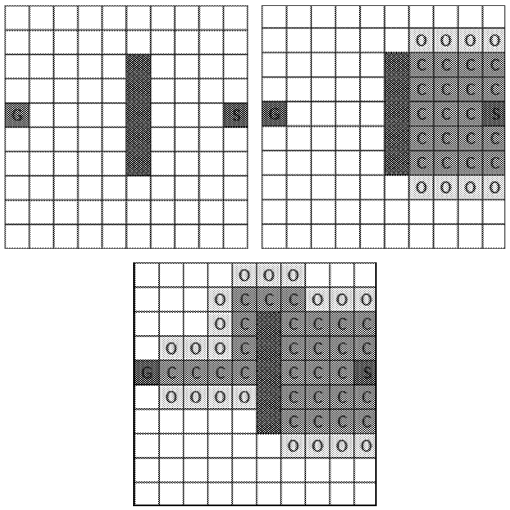


Fig.1. A-Star route sample.

The Best-First-Search algorithm works similarly to A*, but the heuristic adopted is simple and greedy (f (n) = h (n)), tending to expand only the nodes that are close to the goal one, thus the amount of memory required is less than the A* algorithm (as shown in Fig. 2) as a result the solution will not necessarily optimal.

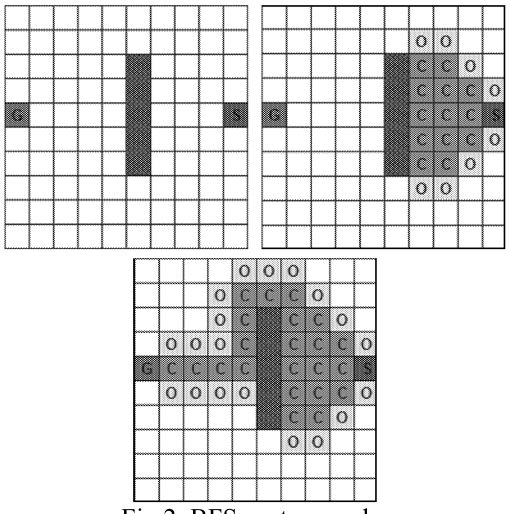


Fig.2. BFS route sample.

This paper proposes a promising algorithm, which the same as IDA*, optimizes the expense of A* memory, but as the Best-First, without guaranteeing the definition of the optimal path. For this, we designed an architecture based in Genetic Algorithms (Mitchell 1996) and in the Best-First Search to optimize the search for paths with a reduction in the number of nodes expanded, and obtaining a path that is not always optimal.

In our previous work (Machado et al. 2011) we proposed an approach that aims to reduce the number of nodes accessed using a module with GA with a local search to move through the environment. Compared with the traditional A* in patterned environments with obstacles, our previous method showed superior performance, but in environments without any kind of pattern obstacles, the performance was far below, not being suitable to many practical cases. Unlike the RTP-GA, our new approach ensures a minimal loss compared to the traditional Best-First in maps without pattern, maintaining a satisfactory standard advantage in environments with obstacles.

We show in (Machado et al. 2011) that the decrease in the number of access nodes is proportional to the processing gain. As investigated by (Korf 1985) and (Russell and Norvig) this relationship is not always proportional. Thus, to avoid this error, in this work we suggest only space optimization (memory).

This papers is organized as following: chapter 2 presents a survey about related works of pathfinding, followed by the PPGA algorithm specifications and features on chapter 3. At chapter 4 we present our tests and comparative analysis between the PPGA, BFS and A* algorithm. Finally, at chapter 5, we present our conclusion and future work.

RELATED WORK

Searching for patterns in maps in order to optimize the pathfinding algorithms is a well-known technique. (Demyenand and Buro 2006) shows that the Triangulation Reduction A* (TRA*) method has many benefits, including accurate representation of polygonal environments, reduction of the search space and refinement of the optimal path. Although this algorithm uses real-time learning, it has severe constraints regarding the environment. To use the TRA* we must provide a description of polygonal obstacles in the environment, which can be easily applied to maps with barriers and holes but it is not suitable for labyrinths and mazes.

The use of meta-heuristics for adaptation to the environment is a topic addressed by several authors. (Leigh et al. 2007) proposed the GA-Manufactured Maneuvering Algorithms (GAMMAs) that was able of finding paths more than 1000% faster than the traditional A*. In presented architecture the GA was able to adapt to the obstacles (called risk zones), approach the human-made path and define the best way to evaluate the path. However GAMMA depends on a prior offline training, that consists in exploring some maps in order to extract some patterns before the optimization process.

(Burchardt and Salomon 2006) implemented a Genetic Algorithm to search for paths for robot routing. The algorithm runs until the agent achieves its goal. There is a constant update process on the environment. The fitness function is based on the length of the path with penalization on collisions. The gene encoding follows the pattern:

Length	X_1	Y_1	X_2	Y_2	X_3	Y_3

The path is encoded as equidistant distances between the starting and ending point, allowing the occurrence of a deviation. However, this approach only works when a nearly direct route exists.

(Bulitko and Lustrek 2006) call as "lookahead" the incomplete search method, which is similar to the min-max based algorithms used in two-person games. It conducted a limited depth search, expanding the space centered on the agent, and heuristically evaluates the distance between the agent and the destination. However, due to the “lookahead” pathology (Bulitko and Lustrek 2006), determining the optimal depth for the search procedure is not an easy task.

As in (Demyenand and Buro 2006) and (Burchardt and Salomon 2006), this work investigates partial solutions in the map, trying to minimize the problems defined by (Bulitko and Lustrek 2006) when using this incomplete search method. We intend to hold a learning process through an online training (as opposed to Leigh et al.), based on a codification of the route, such as presented in (Burchardt and Salomon 2006).

Therefore we decide to use Genetic Algorithms for solving the pathfinding problem combining two approaches of the literature, presented in (Leigh et al. 2007) and in (Burchardt and Salomon 2006). The difference between our model and the one presented in (Leigh et al. 2007) is that we use an online training instead of an offline one. Our approach also improves the model presented in (Burchardt and Salomon 2006) since we define a chromosome able to adapt to any environment that present obstacle patterns.

We generate possible sub-paths (or "lookahead") to be randomly applied in a map, evaluate the sub-paths in the map, and then adapt the sub-paths in order optimize the amount of memory necessary to generate the path. In Fig. 3 we show an example of obstacle avoidance:

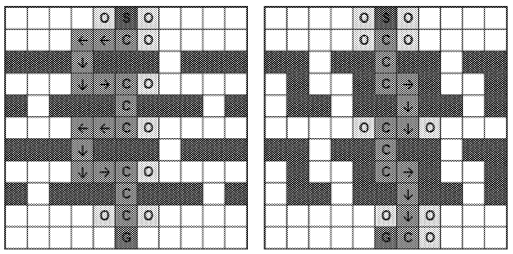


Fig 3: Two examples of obstacles adapting with Patterned based Pathfinding with Genetic Algorithm. The arrows represent the pattern detected by the GA, treated in this way sub-paths, which do not require evaluation of the neighbors.

PPGA

Patterned Based Pathfinding with Genetic Algorithm (PPGA) uses the classical Best-First-Seach with its greedy heuristic to find the path. When the criterion for the use of GA is reached, the fittest individual elected by the same will be used in the environment.

The Genetic Algorithm iterates over a population of solutions that are called individuals or chromosomes. On each generation (algorithm iteration), new individuals are generated by crossover and mutation process and only the most suitable individuals survive for the next generation. The individual fitness is calculated by a fitness function, or objective function. Thus, in each generation, only the individuals with the best values of objective function are selected to be in the next generation.

It is important to note that, in the presented architecture, the quality of individuals (paths) is measured not only by the

The GA use a chromosome modeling based on vectors of movements. Its evaluation function uses a heuristic based on Best-First Search to evaluate the best individual. Some restrictions have been adopted to avoid the generation of invalid individuals, for better use of the GA. These restrictions are used in any environment. The individual chosen as the fittest by GA, which is a sub-valid path, is projected into the environment and its nodes are stored on the closed list of the Best-First without the need to expand them. Thus we obtain a reduction in the number of checked nodes.
As the main processing of PPGA achieve gains at the presented pattern environments, in order to guarantee the solution it was necessary to develop a module for identifying when a maze has this feature. In those cases, the GA module is not used.In the following subsections the procedures for this case will be detailed.

Chromosome Representation

As one of the goals of the algorithm is to detect obstacle patterns in the map, the proposed Genetic Algorithm uses a model based on four movement vectors, two horizontal and two vertical. This model represents a sub-path, which can consist of up to four straight lines, two for each direction, which guarantees the boundary of obstacles. As the maze gets narrow, the distance of the vectors (lines length) tends to decrease, and vice-versa. If the obstacles can be overcome by contouring them, these vectors tend to find this pattern. Each of these four vectors has a distance and a direction, as expressed in Tab 1:

Type	Interval	Description
Float	0.1 to 1	Distance of Movement 1 (DM1)
Float	0.1 to 1	Distance of Movement 2 (DM2)
Float	0.1 to 1	Distance of Movement 3 (DM3)
Float	0.1 to 1	Distance of Movement 4 (DM4)
Integer	-1 to 1	Movement Direction 1 (M1)
Integer	-1 to 1	Movement Direction 2 (M2)
Integer	-1 to 1	Movement Direction 3 (M3)
Integer	-1 to 1	Movement Direction 4 (M4)

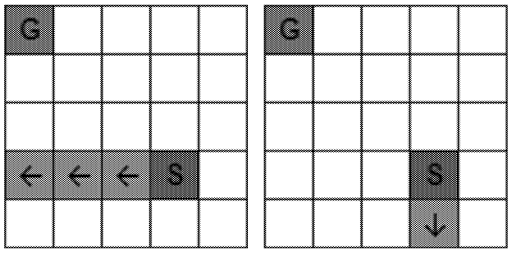
Tab. 1 – Chromosome encoding

The interval where the movement distances may vary represent a percentage of the width (in the case of horizontal vectors) and height (in the case of vertical vectors) of the map. The directions are given by:

- M1 and M3: 0 no movement, -1 down (subtracts the y-axis) and +1 up (increase in y-axis).
- M2 and M4: 0 no movement, -1 to the left (subtracts the x-axis) and +1 to the right (increase in x-axis).

Movement Instances

In order to show the types of changes that may happen, we will consider, with no loss of generality, a 5x5 dimension map. Also let "S" be the initial point of the agent and "G" be the final point (exit). The path under consideration will be represented by arrows in the map. The following situations may be represented in our chromosomes (Fig. 4):



(a) (b)

Fig. 4 – Straight movement.

The Fig. 4 shows a visual representation of the chromosomes exposed in Tab. 2.

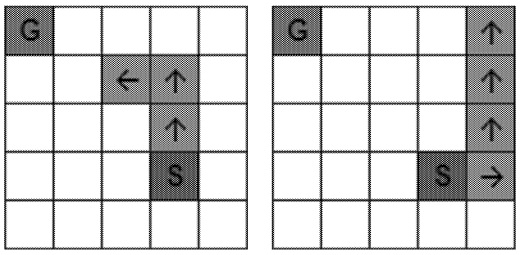
DM1	DM2	DM3	DM4	M1	M2	M3	M4
0.2	0.6	0.4	0.1	0	-1	0	0
0.2	0.9	0.7	0.1	-1	0	0	0

Tab. 2 – Values of straight movement

Tab. 2, line 1, presents an individual in which only the M2 field has a valid value. Thus it represents a single movement to the left on the map (due to the negative value). The distance traveled is a fraction of the map size. As DM2 (relative to M2) has value 0.6 and the map has width 5, the total distance traveled in this movement is given by 0.6 x 5 = 3 squares, as shown in Fig. 4 (a) . The chromosome exposed in Tab. 2 (b) , line 2, also shows a movement with a single direction, presented in Fig. 4 (b). The Manhattan distance value of both chromosomes, i.e. the number of squares between the resulting point of the movement and the maze exit, are 3 and 7, respectively. The objective functions of both chromosomes are shown:

FA(a) = (30 – 3 – 3 * 0,1) * 1 => 26,7
FA(b) = (30 – 7 – 1 * 0,1) * 1 => 22,9

Since the first chromosome of Tab. 2 has a higher FA it is considered better than the second one.
In the next set of examples we will evaluate a movement with a single deviation (Fig. 5) represented by the chromosomes of Tab. 3:



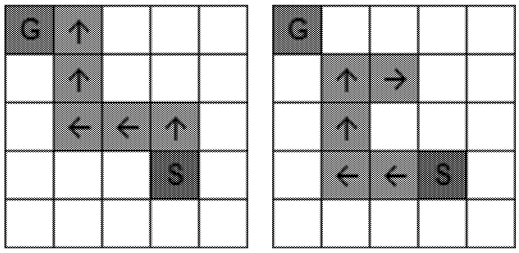
(a) (b)

Fig. 5 – Movement with deviation.

DM1	DM2	DM3	DM4	M1	M2	M3	M4
0.4	0.2	0.5	0.9	1	-1	0	0
0.6	0.2	0.6	0.1	0	1	1	0

Tab. 3 – Values of deviation movement.

The first chromosome has two nonzero directions: M1 vertically and M2 horizontally. Therefore we will have two line segments forming a path with deviation. A similar situation occurs in the second chromosome. Both chromosomes are graphically exposed in Fig. 5.
Tab. 4 presents chromosomes with two deviations, shown in Fig. 6.



(a) (b)

Fig. 6 – Movement with two deviations.

DM1	DM2	DM3	DM4	M1	M2	M3	M4
0.2	0.4	0.4	0.9	1	-1	1	0
0.6	0.4	0.6	0.2	0	-1	1	1

Tab. 4 – Values of the two deviations movement

Fitness Function

The objective function (FA) is used to check each candidate path. It takes into consideration The Manhattan Distance heuristic (MH) and Total Movement (MT).
The MH is represented by the function:

$$MH = | Gx - (Sx + (M2 * DM2 + M4 * DM4) * Width) | + | Gy - (Sy + (M1 * DM1 + M3 * DM3) * Height) |$$

Where Sx and Sy represent the position of the current node and Gx and Gy goal node positions.The higher value of MH is the worse situation, since it means the agent that is far from the final target.
The MT is calculated by the function:

$$MT = (M1*DM1 + M3 * DM3) * Height + (M2* DM2 + M4 * DM4) * Width$$

The weight of MT is smaller than the one of MH to serve as a tie-breaker, and for eliminate cycling paths.

This objective function must be maximized, meaning that the higher the value the better the candidate. To ensure this feature, we adopted a roof value (upper limit) given by:

$$Roof = (Width + Height) * 3$$

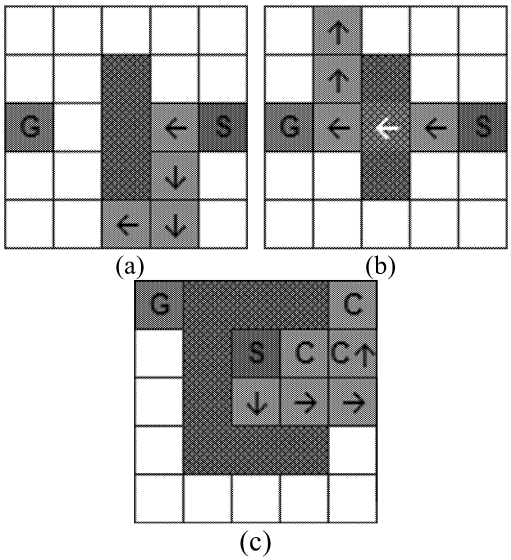
This value represents the sum of the maximum of four movement vectors plus the maximum value of the Manhattan distance. If a path is not valid, the convergence ensures that the result is the lowest possible (in this case, zero). The objective function is given by:

$$FA = (Roof - MH - MT * 0,1)$$

Restrictions, Validations and Adaptions

In order to guarantee better results for the sub-paths we defined some fixed and dynamic restrictions. The first restrictions deny the generation of sub-paths without movements, i.e., with all directions from M1 through M4 with null values. These also avoid the generation of paths where their segments are superimposed, which occurs when M2 is null and M1 and M3 have opposite values and when M3 is null and M2 and M4 have opposite values.
In the second case, related to the dynamic restrictions, there is dependence about the output of the map environment and it the direction of the last collision. For instance, if there is an obstacle at the top neighbor of the last visited point, there will not be a generation of a sub-path for the upper way.
Once generated, the chromosomes are ranked according to their fitness values being sorted in descending order. Each chromosome is subjected to a field test, starting at the first, in this test it will be classified as valid or invalid. Its value is invalid if the MH is greater than the value of "f" of the current node, and if the last node that composes the sub-path chromosome generated is a node and it is already presents on the closed list of the best-first search module, Fig. 7 (c), in this case the chromosome is removed from the population.,

When classified as valid, it moves to the next step of the test, that consists in evaluating if any node that composes the sub-path is an obstacle. If true the chromosome is adapted, Fig. 7 (a) and (b), and the next generation of chromosomes will be evaluated. If false, it is used in the environment and will be maintained for the next generation.

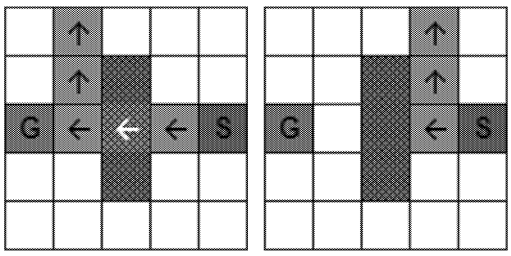


(a) (b)

(c)

Fig. 7. Validation of the paths: (a) is a valid sub-path; (b) is a sub-path to be adapted; (c) is invalid due its last node is at the closed list

The adaptation process was implemented with the purpose of inducing the chromosomes evolution of the population by improving the performance of GA. After this process the individual adapted receives a new fitness value and it will be relocated in the generation, according to this value. If this is chosen, it does not need to go through the second stage of field testing and will be used immediately in the environment.



(a) (b)

Fig. 8. Adaptation of the sub-path

The situation shown in Fig. 8 - (a) represents a sub-path that passes through obstacles. Its codification is listed in Tab. 5.

DM1	DM2	DM3	DM4	M1	M2	M3	M4

In Fig. 8 - (b) the adapted sub-path is replaced by this codification (Tab. 6).

DM1	DM2	DM3	DM4	M1	M2	M3	M4
0.4	0.2	0.4	0.9	0	-1	1	0

Tab. 6 – Codification of the Fig. 8 – (b) Sub-Path

Identification of Environment without Pattern

To avoid an excessive usage of processing in maps that do not have patterns of obstacles, an adopted measurement was the destruction of the GA module once its event is identified.
We defined two metrics for evaluation: the percentage of the distance traveled (D), calculated by the actual distance to the destination ($MD_{current}$) divided by the total distance from the beginning point of the search to the destination (MD_{total}) using the heuristic calculation of Manhattan, as can be seen in Equation 1, and the success rate (S), calculated by dividing the total number of times the reused "sub-path" of the previous generation (C_{first}) divided by the total number of times it failed (U_{first}), shown in Equation 2.
In order to identify if the map is locally patterned (result of the equation is Pattern=1) or not (Pattern=0), the limits of the success rate and the percentage of the distance have been set through a predetermined calibration, being respectively 0.4 and 0.5 as stated in Equation 3.
In case that the pre-determined percentage of the distance between the start and the end points and the success rate is higher than the value stated previously, the GA will continue to be used. If this rate is a low value, the module GA should be disabled, then the algorithm considers that there is not a pattern of obstacles in the region in which the search occurs.

$$D = \frac{MD_{current}}{MD_{total}} \quad \text{Equation - 1}$$

$$S = \frac{\sum C_{first}}{\sum U_{first}} \quad \text{Equation - 2}$$

$$Pattern = \begin{cases} 0, & D < 0{,}5 \ \ and \ \ S < 0{,}4 \\ 1 & \end{cases} \quad \text{Equation - 3}$$

PPGA Architecture

As the PPGA was developed based at a modification of the BFS, we introduced a module that uses the Genetic Algorithm in order to calculate the sub-paths that are used along the processing. Therefore, each time that the module is called, the generated sub-paths will inherit information of the past generations, making possible a considerable gain of performance. The event for calling the GA module happens when none of the values of the open node list of the BFS is better than its actual node. Fig. 9 shows the proposed architecture.

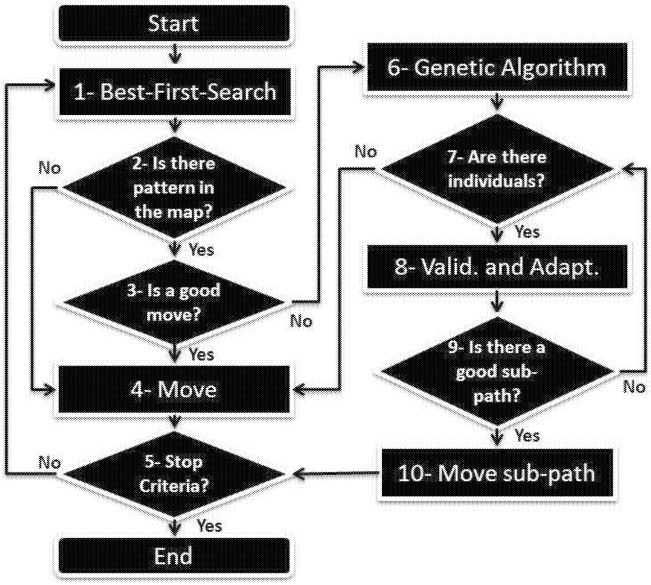


Fig. 9. Patterned based Pathfinding with Genetic Algorithm architecture

1. Best-First-Search – For each BFS iteration list, the actual node, previously added to the closed list, is expanded and the open list node with the lowest value of F is selected (greedy algorithm)
2. Is there a pattern? – In the genetic algorithm module there is a counting variable that defines the proportion of times that the sub-paths were used by the tentatives of usage. If this proportion is low, we define that there is no pattern for the environment and from there on the AG module will not be used anymore.
3. Is a good move? – This verify if the selected node corresponds to a good movement, i.e., if its value is lower than the actual node.
4. Move – adds the selected node to the closed list.
5. Stop Criteria – If the actual node consist on the destiny node, the algorithm ends, other way it continues for the next iteration.
6. Genetic Algorithm – In this module it is obtained only one generation, following the determined restrictions by the following steps:
   a. Generate a initial population of 4 individuals;
   b. Generate the first off-spring by a crossover of the 4 first individuals. The established rate was 50%;
   c. The second off-spring is generated by the mutation of the initial population. The established rate for this stage was 25%;
   d. It is calculated the fitness value for the 12 individuals. The element that was used in the last generation receives the largest value;
7. Are there individuals? – It is verified if there are still chromosomes to be evaluated at the population. In case none of the individuals are valid, the algorithm proceeds maintaining the current node still selected.
8. Validation and adaptation – This stage helps to guarantee the generation and adaptation of good individuals coming from the AG.
9. Is there a good sub-path? – The generated individuals go by this test, one at each iteration. If it is not a good element, this will be taken off from the list of population.
10. Move sub-path – The element will be used as a sub-path composed by the nodes that will be added to the closed list. The actual node will become the last node of the sub-path. This element will be inserted to the next

Fig. 10 illustrates part of this architecture. In (a) the BFS does not have a neighborhood that is better than its current state, so it will be call the GA module. In (b) the valid sub-path is projected. In (c) and (d) the BFS ends the search.

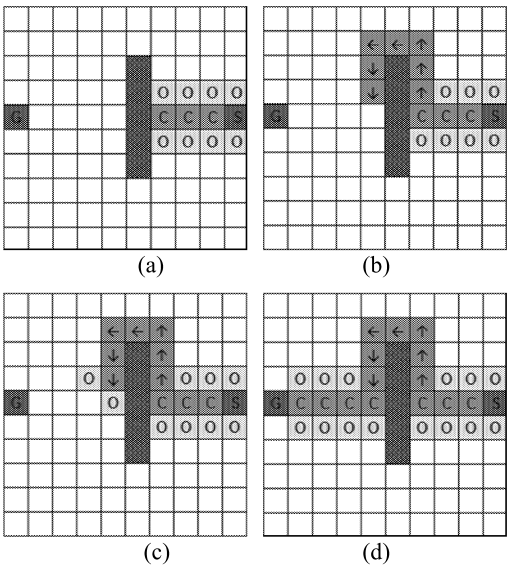


Fig. 10 – Part of the stages of execution of the PPGA among a maze after a training step.

PPGA Algorithm

The next two algorithms represent, respectively, the main module and function of the genetic algorithm:

```
PPGA()

01. OPEN_LIST ← ⍉
02. CLOSED_LIST ← ⍉
03. CURRENT_NODE ← START
04. SUCCESS ← 0
05. NOT_SUCCESS ← 0
06. SUCCESS_RATE ← 0
07. DISTANCE_RATE ← 0
08. BEST_INDIVIDUALS ← ⍉
09. while CURRENT_NODE is not GOAL do
10.     add(CLOSED_LIST,CURRENT_NODE)
11.     for earch neighbor from CURRENT_NODE do
12.           if neighbor is not on CLOSED_LIST
          and is not on OPEN_LIST then
13.              distance ←
manhattan(neighbor,GOAL)
14.           setCost(neighbor,distance)
15.           setParent(neighbor,CURRENT_NODE)
16.           add(OPEN_LIST,neighbor)
17.           end
18.     end
19.     NODE ← removes the lowest cost item of
the OPEN_LIST
20.     SUCCESS ← SUCCESS/ NOT_SUCCESS
21.     DISTANCE_RATE ←

  manhattan(START,GOAL)/manhattan(CURRENT_NODE,G
OAL)
22.     if
MapWithPattern(SUCCESS_RATE,DISTANCE_RATE)
equals to 1 then
23.        if NODE's cost is higher than
CURRENT_NODE's cost then
24.           BEST_INDIVIDUALS ←
              MODULE_GA(BEST_INDIVIDUALS,
CURRENT_NODE, GOAL)
25.           if BEST_INDIVIDUALS not ⍉ then
26.              LIST_NODES ←
27.              convert BEST_INDIVIDUALS to
list of nodes
28.              add nodes from LIST_NODES in
```

```
30.              SUCCESS++
31.           else
32.              NOT_SUCCESS ++
33.           end
34.        end
35.
36.     end
37.     CURRENT_NODE ← NODE
38.
39. end
```

```
MODULE_GA(BEST_INDIVIDUALS, CURRENT_NODE, GOAL)

01. GENERATION ← ⍉
02.  generateInitialPopulation(GENERATION,
BEST_INDIVIDUALS,CURRENT_NODE)
03. crossover(GENERATION,CURRENT_NODE)
04.  mutation(GENERATION,CURRENT_NODE)
05.
calculateFitnessValueAndSort(GENERATION,GOAL)
06.  for earch individual from GENERATION do
07.     if evaluationInEnvironment(individual)
then
08.        return individual
09.     else
10.        new_individual = adaptate(individual)
11.        add new_individual in the GENERATION
12.     end
13.  end
14.  return ⍉
```

The main module of the algorithm is very similar to the classic Best-First Search, it uses Open and Closed lists, represented by OPEN_LIST and CLOSED_LIST respectively, and an additional structure called BEST_INDIVIDUALS, which stores the best individual obtained by the GA module (MODULE_GA). The variables SUCCESS, NOT_SUCCESS, SUCCESS_RATE and DISTANCE_RATE are used to determine if the MODULE_GA will be used.

Like Best-First Search, at each iteration a node presents in OPEN_LIST is chosen according to their cost (line 19). If the MODULE_GA is enabled (line 22) before the chosen node is expanded and added to CLOSED_LIST must evaluate whether its cost is greater than the cost from the current node (line 23), if the iteration continues false. If it is true, the function MODULE_GA is called (line 24). This function returns an individual which is converted to a list of linked nodes, beginning from CURRENT_NODE, forming a sub-path. These nodes are added to CLOSED_LIST and the last becomes the NODE (lines 26-29).

The GA module uses the genetic algorithm to generate a sub-path better every generation through genetic inheritance contained in individuals (BEST_INDIVIDUALS). Another important parameter is the CURRENT_NODE, that is used to generate the restrictions that must be obeyed in all phases of GA (lines 02-05), since the parameter GOAL is used to compute the fitness value (line 06).

After individuals are generated and their fitness values calculated, must be stored and sorted in GENERATION. Each individual goes through an evaluation process where the environment determines whether it is valid or not. This evaluation verifies whether the sub-path does not pass over any obstacle map, if true this is returned. If the individual crossed through some obstacle, should be adapted and returns to GENERATION so it can be used if all individuals are invalid.

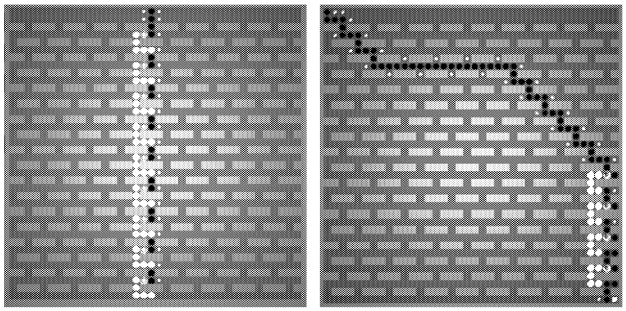
In some cases MODULE_GA returns nothing, it may occur more frequently if the environment is not formed by obstacles which follow a pattern, from the moment it starts happening more frequently this is disabled (line 22).

EXPERIMENTS

Since one of the goals of this work is to prepare a test bed for optimization algorithms for the pathfinding problem, suitable programming language and development environment were chosen. Another factor considered in the choice of language and development environment was the possibility of exporting the developed systems to mobile devices such as tablets and mobile phones where the optimized use of the resources is highly recommended.

Technology

The development technologies used were Unity 3D [2] through the C # language.
Unity3D 3.x is a game development engine which has the main components needed for designing games developing rapidly, such as the physics engine, collision systems, sound system and a high level programming language based on C, among others.
Unity 3D has an interface for programming in Mono. It incorporates key .NET components, including a compiler for the C# programming language and a complete suite of class libraries.
We executed this algorithm in a Tablet with Android Operational System 2.1 éclair, as exposed in Fig. 11. We note that in (a) a pattern was found at the beginning. As predicted, in (b) the algorithm spent some time to activate the module GA since the value of F for all nodes of the open list is smaller than the current node in most of the time.



(a) (b)

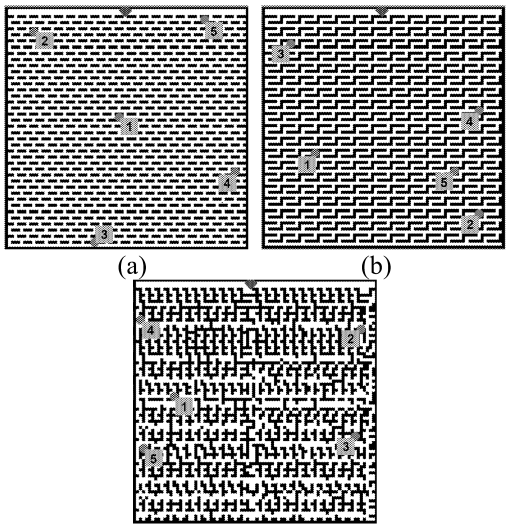
Fig. 11 – PPGA simulation in a tablet. Black circles represent the closed list, small white circles the open list and large white is the sub-path generated by GA".

Tests

We conducted three tests to compare the costs of memory between A*, BFS and PPGA algorithms. Each test was made in a different map, with constant dimensions of 80 by 80 tiles. In all, to be deterministic (and therefore not generate solutions with different memory costs in an environment without variation), the A* and BFS was performed only once. In contrast the average was calculated for 20 runs of PPGA in each test case. In order to characterize the principle of a dynamic environment, we defined five destinations in each map. Once a target was reached, the open and closed lists of all algorithms were reseted, simulating changes in the environment.

The maps types were:

- With patterns (Fig. 12 (a)) - Aiming to assess the advantages of the adjustment proposed by PPGA;
- Mixed Map (Fig. 12 (b)) - In order to evaluate a balanced case.



(a) (b)

(c)

Fig. 12 – Map patterns

Results and Analysis

The results show that the agents managed to adapt themselves to the surrounding environments even in the presence of obstacles.
One objective of the tests was to check if the agents are able to find a path to the maze exit. In 100% of the cases the agents achieved the goal.
The result of the experiments can be checked in Ta. 7, in which the values represent the number of access nodes.

	Map		
Algorithm	**With Patterns**	**Mixed**	**Without Patterns**
A-Star	16016	27692	18468
Best-First	2048	3088	5796
PPGA	1725	3142	6252

Tab. 7 - Tests Results

In Fig. 14 it is possible to check the comparative projection of BFS and PPGA. We do not show the A* in this comparison because its high values would undermine the view of the difference.

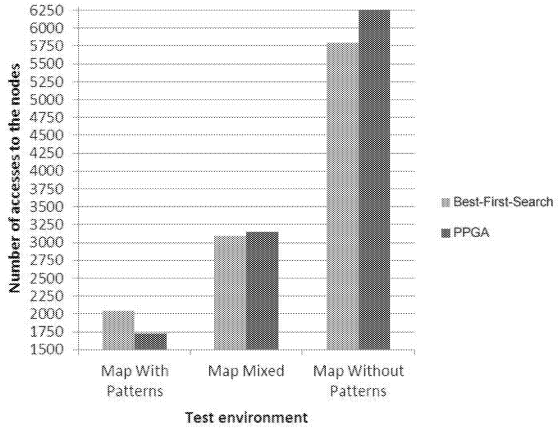


Fig. 14 – Map Patterns Comparison.

The purpose of this study is optimize the search for paths between two points through the reduction of nodes accessed or expanded, but without the guarantee of the shortest path.
The main result of this research was to optimize 16% of the expense of processing the PPGA compared to the Best-First

In the other two tests the results were balanced (losing by 2% on the mixed map) or slightly worse (losing by 8% on the map without patterns).
In the test without patterns, the proportional loss of processing is due to the fact that in this case the PPGA algorithm tends to generate sub-paths more frequently, because there is no genetic advantage of the previous chromosomes which leads to an overspending of memory. This result could be worse if wasn't the presence of the pattern identification module inserted into the algorithm.
The results were in accordance with the expectation, since the PPGA has a better performance on maps with patterns of obstacles due to the adjustment provided by the Genetic Algorithm.

CONCLUSION AND FUTURE WORK

This work defines a new method for searching the path between two points in 2D maps. The PPGA uses the classic BFS with a Genetic Algorithm in order to find good solutions in a short period of time.
The proposed GA used "lookahead" based modeling to the chromosome that can be adapted to any kind of map that contains obstacle patterns. Also the "lookahead" pathology is avoided by the iterative nature of the algorithm.
For this work, dynamic environment is defined as one that can suffer alteration in the position of the obstacles or target at any time. For this, our experiments were conducted on large maps with more than one goal. The knowledge gained in the initial executions ensures the adaptation of the agent to new situations with lower latency learning, in practical situations.
Experiments proved that the PPGA showed slightly better than the Best-First Search being compared on maps of three variations of patterns of obstacles. When there are patterns in this distribution, the proposed algorithm achieved its best performance with an average gain of 16% when compared with the BFS. Therefore we consider this architecture promising.
As future work, we will seek improvements in this architecture to optimize the search process, such as varying the number of segments of lines contained in the chromosome and improving the function of identifying patterns in the map.

ACKNOWLEDGEMENT

The authors acknowledge the support of the Instituto Federal de Educação, Ciência e Tecnologia do Sudeste de Minas Gerais and Laboratório de Multimídia Interativa IF Sudeste - MG. Besides financial support provided by Coordenação de Aperfeiçoamento de Pessoal de NÌvel Superior - CAPES - Brasil, through Programa de Educação Tutorial do Ministério da Educação (PET-MEC).

REFERENCES

Dechter, R., and Pearl, J., *"Generalized Best-First Search Strategies and the Optimality of A*."* Journal of the Association for Computing Machinery , Vol. 32, No. 3, July 1985, pp. 505-536.

Leigh, R., Louis, S. J. and Miles, C. *"Using a Genetic Algorithm to Explore A*-like Pathfinding Algorithms"*. In: IEEE Congress on Computational Intelligence and Games. CIG – 2007.

Demyenand, D. and Buro, M. *"Efficient Triangulation-Based Pathfinding"*. In: AAAI'06 Proceedings of the 21st national Conference on Artificial intelligence. 2006.

Burchardt, H. and Salomon, R. *"Implementation of Path Planning using Genetic Algorithms on Mobile Robots"*. IEEE Congress on

Bulitko, V., Lustrek, M. *"Lookahead pathology in real-time path-finding"*. In Proceedings of the National Conference on Artificial Intelligence. AAAI 2006.

Hart, P.E., Nillson, N.J. and Raphael, B. *"A formal basis for the heuristic determination of minimum cost paths"*. IEEE Transactions on Systems Science and Cybernetics, 1968.

Bjornsson, Y., Enzenberger, M., Holte, R.C. and Schaeffer, J. *"Fringe search: Beating A* at pathfinding on gamemaps"*. IEEE Computational Intelligence in Games, 2005.

Stodola, P. and Mazal, J. *"Optimal location and motion of autonomous unmanned ground vehicles"*. In World Scientific and Engineering Academy and Society. WSEAS 2010.

Mitchell, M.; *"An introduction to genetic algorithms"*. The MIT Press, United States; 1996.

Korf, R. *"Depth-first iterative deepening: An optimal admissible tree search"*. Artificial Intelligence, (1985).

Stuart J. Russell and Peter Norvig, *"Artificial Intelligence: A Modern Approach"* , Second Edition.

Machado, A. F. V. ; Clua, E. W. ; Goncalves, R. ; Valverde, I. P. ; Santos, U. O. ; Neves, T. ; Ochi, L. S. *"Real Time Pathfinding with Genetic Algorithm"*. In: SBGames, 2011, Salvador, BA. Simpósio Brasileiro de Jogos e Entretenimento Digital (SBGames), 2011.

WEB REFERENCES

Unity Technologies: Unity 3D User Manual. At www.unity3d.com/support/documentation/.

BIOGRAPHIES

ULYSSES O. SANTOS studied computer science at the IFSUDESTE-MG. He is a research assistant at the group of the Programa de Educação Tutorial of Ministério da Educação (PET-MEC).

ALEX F. V. MACHADO has PhD in Computer Science at the Universidade Federal Fluminense. He is professor of Bachelor of Computer Science at IFSUDESTE-MG. and Coordinator of the Laboratory for Interactive Multimedia IFSUDESTE-MG, of the Programa de Educação Tutorial of Ministério da Educação (PET-MEC).

ESTEBAN W. G. CLUA is Professor at Universidade Federal Fluminense and general coordinator of the UFF Medialab. He is PhD in Informatics from PUC-Rio. He is one of the founders of SBGames, SBC, Director of the IGDA Rio Academy, responsible for research and academy in the area of digital entertainment in the country and a pioneer in the field of scientific research in games and digital entertainment.

AUTHOR LISTING

AUTHOR LISTING

Alvarado A. **31**

Batista I.A..... **5**
Bird J..... **48**
Brown G. **34**
Buche C. **64**

Carvalho S.P..... **5**
Chowdhury M.I..... **41**
Cielniak G. **17/48**
Clua E.W.G..... **5/79**
Cotta C..... **25/53/71**

De Loor P..... **64**
Dickinson P. **17**

Eliëns A..... **8**

Feltwell T..... **17/48**
Fernández-Leiva A.J. ... **53/71**

Gálvez J.M. **71**
García-García C..... **61**
Gaubert L. **64**

Hernández B..... **31**

Katchabaw M. **41**
King D..... **34**

Lara-Cabrera R..... **53**
Larios-Rosillo V. **61**
Luga H. **61**

Machado A.F.V. **5/79**
Merelo J.J. **25**
Mora A.M. **25**

Nogueira M. **71**

Padovani R.R. **5**

Santiago M.C. **5**
Santos U.O. **79**
Soares B.G. **5**

Tencé F. **64**