11TH INTERNATIONAL CONFERENCE

ON

INTELLIGENT GAMES AND SIMULATION

GAME-ON® 2010

EDITED BY

Aladdin Ayesh

NOVEMBER 17-19, 2010

Holiday Inn

Leicester

UNITED KINGDOM

A Publication of EUROSIS-ETI

11TH International Conference
on
Intelligent Games and Simulation

LEICESTER, UNITED KINGDOM

NOVEMBER 17 - 19, 2010

Organised by

ETI

Sponsored by

EUROSIS

Co-Sponsored by

Binary Illusions

Delft University of Technology

De Montfort University

Ghent University

Larian Studios

University of Skövde

Hosted by

Holiday Inn

Leicester, United Kingdom

EXECUTIVE EDITOR

PHILIPPE GERIL
(BELGIUM)

EDITORS

General Conference Chair

Dr. Aladdin Ayesh
De Montfort University, Faculty of Technology, Leicester, United Kingdom

General Program Chair

Professor Dimitrios Rigas
De Montfort University, Faculty of Technology, Leicester, United Kingdom

INTERNATIONAL PROGRAMME COMMITTEE

Game Development Methodology

Track Chair: Licinio Roque, University of Coimbra, Coimbra, Portugal
Óscar Mealha, University of Aveiro, Portugal
Jari Multisilta, University of Tampere, Finland
Esteban Clua, Universidade Federal Fluminense, Brasil

Physics and Simulation

Graphics Simulation and Techniques
Ian Marshall, Coventry University, Coventry, United Kingdom
Marco Roccetti, University of Bologna, Bologna, Italy

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

Rendering Techniques
Joern Loviscach, Hochschule Bremen, Bremen, Germany
Frank Puig, University of Informatics Sciences, Havana, Cuba

Artificial Intelligence

Artificial Intelligence and Simulation Tools for Game Design
Maria Arinbjarnar, University of York, York, United Kingdom
Stephane Assadourian, UBISOFT, Montreal, Canada
Jonathan Attfield, Supermassive Games, Guildford, United Kingdom
Cameron Browne, United Kingdom
Michael Buro, University of Alberta, Edmonton, Canada
Patrick Dickinson, Lincoln University, Lincoln, United Kingdom
Antonio J. Fernandez, Universidad de Malaga, Malaga, Spain
Marek Grzes, University of Waterloo, Canada
Joseph Kehoe, Institute of Technology Carlow, Carlow, Ireland
David King, University of Abertay, Dundee, United Kingdom
Sandy Louchart. Herriot-Watt University, Edinburgh, United Kingdom
David Moffat, Glasgow Caledonian University, Glasgow, United Kingdom
Gregory Paull, The MOVES Institute, Naval Postgraduate School, Monterey, USA
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
Olli Leino, IT-University of Copenhagen, Copenhagen, Denmark
Sean Pickersgill, University of South Australia, Adelaide, Australia
Richard Wages, Nomads Lab, Koln, Germany

Storytelling and Natural Language Processing

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

Modelling of Virtual Worlds

Rafael Bidarra, Delft University of Technology, Delft, The Netherlands

Online Gaming and Security Issues in Online Gaming

Marco Furini, University of Modena and Reggio Emiliano, Modena, Italy

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

Games Analysis

Clive Chandler, Staffordshire University, Stoke-on-Trent, United Kingdom

Wargaming Aerospace Simulations, Board Games etc....

Roberto Beauclair, Institute for Pure and Applied Maths., Rio de Janeiro, Brazil
Richard Ferdig, University of Florida, Gainesville, USA
Henry Lowood, Stanford University Libraries, Stanford, USA
Tony Manninen, Ludocraft Ltd., Oulu, Finland
Jaap van den Herik, Tilburg University, Tilburg, The Netherlands

Games for training

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

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

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

Games Interfaces - Playing outside the Box

Games Console Design

Chris Joslin, Carleton University, Ottawa, Canada

Mobile Gaming

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

Perceptual User Interfaces for Games

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

GAME ON®
2010

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: 978-9077381-58-8
EAN: 978-9077381-58-8

Preface

Game technologies have been evolving over the last couple of years and coming into force into the main stream of computing. As a result, games development started to influence system analysis and design, development methodologies and software architectures. However, games were the playground of Artificial Intelligence researchers prior to their transformation into the lucrative market they currently fulfill with dedicated technologies.

Artificial intelligence research has had a fascination with games from board strategy games such as chess, puzzle games, paradoxes to simulation games. Simulation games in particular provided testbeds for many AI methodologies and theories such as AI planning, intelligent agents, and cognitive architectures. In fact, one may claim that AI simulation gave the birth to the elaborate video games of today. Nonetheless, these same games are providing a driving force to rediscover and in some cases rework AI techniques in both contexts of computational intelligence and cognitive sciences. For example, emotions and perception are two important aspects of believable characters that often use agent technologies as their underlying architecture with other elements of AI such as planning, heuristic search and machine learning.

Game technologies influence does not stop at AI and simulations. It currently extends to main stream computing as mentioned. This in fact gave birth to a new field of game research that is serious gaming. Serious gaming research focuses on the application of game technologies to serious applications which impact on two important areas of software development: methodologies and user interfaces.

This year's GameOn conference provides a meeting place to researchers from all disciplines of gaming research where AI researchers, game designers, software architects, computer graphics and simulation systems developers can interact and exchange ideas.

Dr. Aladdin Ayesh
GameOn 2010 General Chair

CONTENTS

CONTENTS

GAME DESIGN

SCIENTIFIC PROGRAMME

GAME METHODOLOGY

AN EVALUATION OF DIFFICULTY HEURISTICS IN GAME DESIGN USING A SIMULATED PLAYER

Fergal Costello
Department of Information Technology
National University of Ireland, Galway, Ireland
Email: f.costello2@nuigalway.ie

Colm O'Riordan
Department of Information Technology
National University of Ireland, Galway, Ireland
Email: colm.oriordan@nuigalway.ie

KEYWORDS

Artificial Intelligence, Game Methodology

ABSTRACT

In this paper we present our research in the area of authoring tools development. The over arching goal of this work is to provide mechanisms that can be used to measure the design characteristics of games. We aim to develop a framework that can offer designers ongoing informative feedback on the difficulty of a level as they build prototypes, thus allowing them to gauge their designs. We outline a number of measures, which we hypothesise can be useful in predicting the difficulty of a level. In this work we present details of our exemplar game, and the reasons for choosing it as our test bed. We incorporate a naive strategy for simulating a player's performance in any given instance of this game. In order to verify the usefulness of our measures, we undertake an extensive set of experiments using this simulated player. We present the results illustrating the relationship between our measures and the performance of the simulated player.

INTRODUCTION

Though still in its infancy, the technological strides that have been made in the computer games domain over the last number of years has been phenomenal. The current depth and breadth of three-dimensional worlds has allowed game designers to create more elaborate game play experiences, but its has also brought about added complexity in development time. With the industry showing no signs of slowing down, it is up to designers to create more immersive environments, while also managing the game-play experience to make sure that the users find the worlds fun and compelling.
In an industry in which the budget for AAA game titles can run up into tens to hundreds of millions of dollars, rapid prototyping is more important now than ever. The current practice for creating of game worlds follows a four stage iterative approach of: conceptualise, plan, execute and refine (Castillo and Novak, 2008). Using this approach an idea is conceived, expounded on paper, built using a digital content creation (DCC) tool and then refined. This refinement involves play testing to work out what parts of the level are fun to play, and which are not, which mechanics work and which ones don't. In this testing, users' experiences may be monitored and recorded to give the designers an insight into the quality of world. Based on this feedback, the refinement may be unobtrusive and small parts amended, or, in the worst case scenario, the entire level needs to be completely redesigned. Changes can fall under many categories and may include, among others, rewriting agent behaviours, redesigning the flow through the level, modifying placement of objects, changing script based events, altering shading and lighting and other aesthetic features. Any of these costs incurred can represent a large expense for game developers, especially if they occur later in the development processes (as is true for any software engineering task).

The effort of this research is to investigate ways in which work of designers can be supplemented by automated testing of their designs using techniques driven by artificial intelligence. We are exploring ways by which such techniques can be incorporated into a design tool, thus allowing for decisions to made about a game mechanic or level. as their are being built, from early prototypic stages. We see this as a possible way of alleviating some of the issues surrounding the development processes of contemporary games and not as a way of replacing designers. This paper illustrates work we have undertaken in defining a set of measures for quantifying difficulty and outlines a set of experiments we have performed in showing correlation between some of those measures and a simulated player's experience in the designs.

The remainder of this paper is laid out as follows. The next section outlines the related work in the field. Then we introduce the abstract game we utilise to test our hypotheses. From here we discuss a set of heuristics for this game, which we believe capture its difficulty. Next we outline the experimental set up and results, which discuss the correlation between our measures and the outcomes. Finally, we conclude and discuss future work of this research.

RELATED WORK

One of the fields of research related to this work is that of Dynamic Difficulty Adjustment, which modulates in-game systems to respond to a particular player's abilities over the course of a game. Hunicke presents a system, termed Hamlet, which maps the state of a game world to a set of adjustment actions to intervene on behalf of the player (Hunicke, 2005). Generally speaking, this approach adjusts the game mechanics such as the amount of damage the player can inflict on enemies, and the amount enemies can inflict on players, enemy accuracy, spawn locations etc. Adjustment of these values serves as a way to adapt to a player's ability. Similar research has been undertaking in the notion of dynamic scripting. In his work, Spronck developed an unsupervised online learning technique, where scripts of rules are extracted from a rule-base as NPC opponents are instantiated. The probability that a rule is selected for use in a script is proportional to how well it performed previously (Spronck et al., 2004). This coupled with his work on difficulty scaling (Spronck et al., 2006), means that a game can be adapted automatically, to modify the challenge posed to a player, by further enhancing the probabilities by which rules are selected.

Yannakakis and Hallam present work that focuses on the contributions made by an opponent's behaviour to the entertainment value of a game (Yannakakis and Hallam, 2006). They argue that because "interest" and "enjoyment" aren't explicitly definable, there is no evidence that an opponent's behaviour (in a learning context) may be fun to play against. With this in mind, they developed a set of neural networks to model player satisfaction (interest) and they investigate how qualitative factors such as challenge and curiosity contribute to entertaining experiences.

Some researchers have focussed their effort on guiding the automated approaches of procedural content generation to incorporate design considerations, thus allowing for factors such as difficulty and an interest based utility to be intrinsically managed. In her work, Smith discusses a way of generating two-dimensional platformer levels by first generating a rhythm for the level and then blocking out these rhythm segments with geometrical grammar according to a set of physical constraints (Smith et al., 2009). Similarly, Togelius presents an approach that can generate tracks for driving games using evolutionary computation. They first use player modelling to capture a player's driving style and then use this data to guide the evolution of a set of tracks that the player may enjoy (Togelius et al., 2007).

ABSTRACT GAME

As previously mentioned we use an abstract game on which to test our hypotheses. Before discussing measures of difficulty, we first outline this game on which these measures will be derived and evaluated. This game is an instantiation of the patroller-intruder game, which has received much attention in recent years as a way of developing patrolling strategies for mobile robots (Basilico et al., 2009; Amigoni et al., 2010). We believe that this type of game is a good abstraction of many contemporary computer games, where one set of agents outnumber and oppose a single agent in the environment. The latter is analogous to a human player playing a computer game trying to overcome a set of non-player characters (the patrollers) in order to progress from one encounter to the next. A more explicit mapping, however, would be to commercial games that incorporate stealth as their primary gameplay mechanic. As this game stems from a game theoretic background, the description of the environment in the literature generally involves a grid of cells, with agents moving from cell to cell in discretised intervals. In our instantiation, however, we have adopted a more conventional game environment implementation with continuous update cycles, and agents driven by rigid body dynamics and steering behaviours. The goal state of each round of the game is based on the following: if the intruder attains the goal, it is marked as a win, if they are perceived by the patrollers, it is marked as a loss, and if neither of these states is achieved within a given time interval, it is marked as a time out. A screen capture of this game is presented in Figure 1. We believe that a simulation environment further enhances our findings and lends weight to our hypotheses as opposed to using a purely abstract framework, which would be harder to evaluate and validate in a computer games context. It is important to note however, that the heuristics we use to calculate the difficulty were derived from the abstract game.

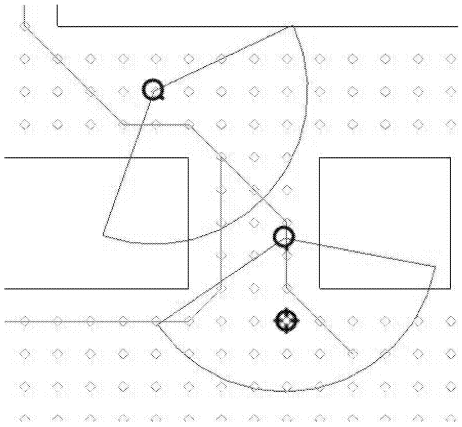


Figure 1: Abstract game, with two agents with their fields of view visible, following paths assigned to them, navigating around simple geometrical shapes. The small diamonds are nodes on the navigation graph and the cross hair acts as the target.

MEASURES OF DIFFICULTY

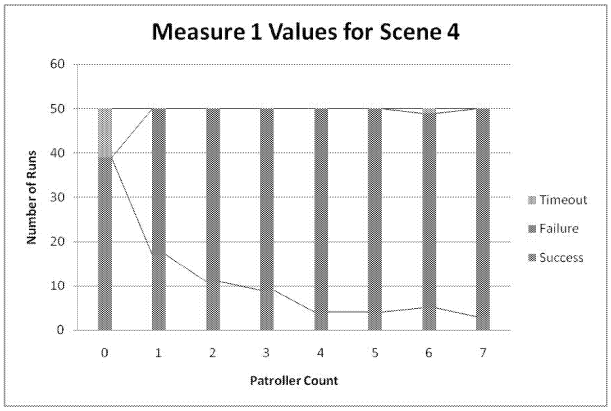
In previous work we posited that a set of heuristics could be used to measure the difficulty of an abstract game level (Costello and O'Riordan, 2009). We developed these heuristics from intuition, and this section outlines how we have incorporated these into our current framework. Although we discussed a number of measures in the previous work, we only tackle two of the measures in this paper, namely:

1. The number of patrollers in the environment
2. An approximation of the area of the scene that is covered by the patrollers

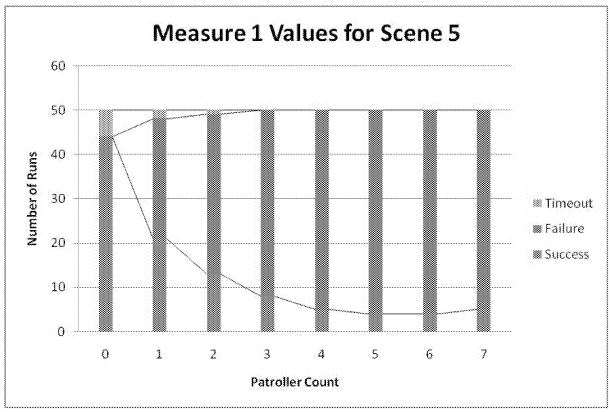
The first of these is merely a record of the number of patrollers in the environment. The second is more complex in nature, as it estimates how much of the scene is perceived by the patrollers with respect to the underlying navigation graph used by the agents. This can be formally expressed as:

$$Measure_2 = \sum_{i=0}^{N} \frac{Length_i \times Perc_i}{Area} \qquad (1)$$

where N is the number of patrollers in the scene, *Length* and *Perc* are the length of the patrol path and the perception ability respectively of patroller i and *Area* is the area of the scene.
The following sections explore correlations between our heuristics and the results of our experiments involving a simulated player.



(a) Scene 4



(b) Scene 5

Figure 2: The number of intruder wins, losses and time outs for Scenes 4 and 5 organised by the number of patrollers in the scene.

EXPERIMENTAL SETUP

The experimental setup is as follows: we create five different scenes (level layouts). For each scene, we create a large number of instantiations. We create 50 runs with zero patrollers, 50 runs with one patroller, and so on, up to and including seven patrollers. This resulted in 400 scenarios per scene. For each of these, the target is randomly placed in the world and the paths of the patrollers are randomly assigned. At the end of each round we record the values of the two heuristic measures, described in the previous section, calculated for that instantiation. We also record the outcome of the round noted as a win for the intruder, a loss for the intruder or a time out. The time interval chosen for each round was three minutes, as we felt this was adequate for the size of the levels we were using (1600x1200 pixels). Finally, for this experiment set we utilised a naive simulated player, who executes a random walk through the environment in order to reach the goal. It has no notion of exploration, nor of self preservation from the patrollers. Although it may be limited in its expected performance, we feel that this is an adequate base case for the simulated players we hope to implement in future work. The next section presents and discusses the results of these experiments.

RESULTS

Measure 1 - Number of Patrollers

We first describe the results pertaining to the simpler of the two measures — the number of patrollers present. We hypothesise that as we increase the number of patrollers, the difficulty of the environment should in turn increase. The graphs presented in Figure 2 show the number of successes, failures and time outs for a subset of the scenes run as part of the experiments — Scene 4 and Scene 5. The data shows that with zero patrollers in the scene there are zero failed outcomes for either scene. What is interesting however is that Scene 4 has an unusually high number of time outs compared to all other scenes at this patroller level. This may be the result of the layout of the scene and suggests that there are other measures of difficulty in effect not accounted for, such as the geometrical layout of the level. As is presented in

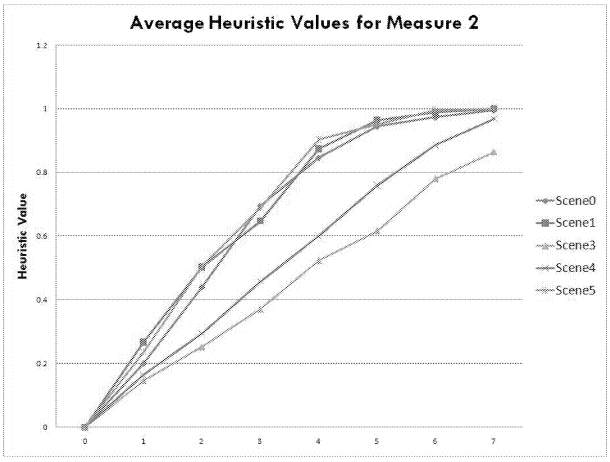


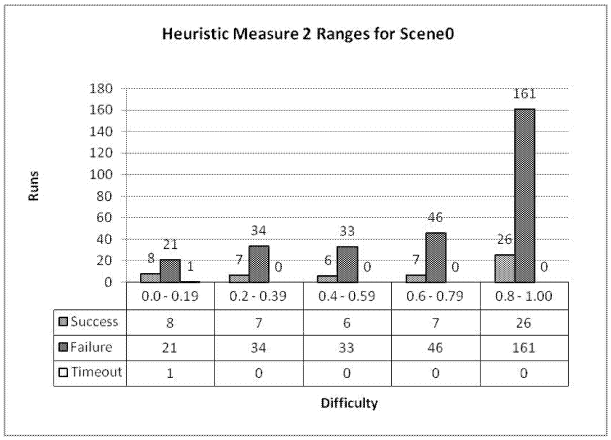
Figure 3: Estimated perception ability plotted against patroller count averaged across each scene.

Figures 2a and 2b, and indeed across all data sets, we see that as we add just one patroller, the level of difficulty (the number of losses) increases. As the patroller count increases, the number of intruder successes decreases. For the later values of patroller count (5, 6 and 7), there is little difference in the number of successes with the intruder failing in nearly all cases.

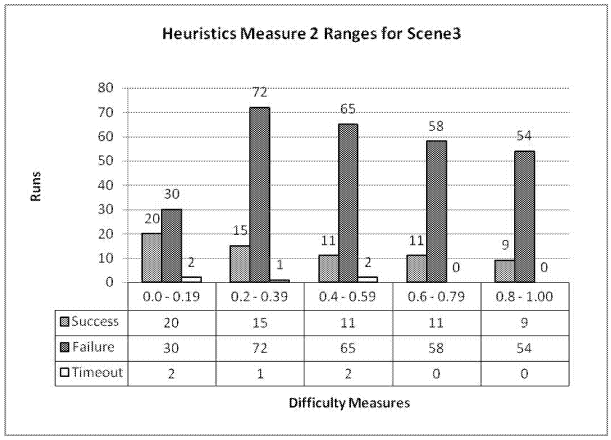
Measure 2 - Estimated Perception Ability of Patrollers

More interesting features are visible from the results obtained when calculating heuristic 2, an estimate of the probability of perception of an intruder. In presenting Figure 3, we show measure 2 averaged across each scene at different patroller count levels. As expected, as the number of patrollers increase, so too did their estimated ability to perceive the intruder. It is interesting to note, however, that we observed variances in the rate of growth across the different environments. Scenes 0, 1 and 5 have a linear growth up until about 4 patrollers and then they taper off as they approach one.

Scene 3 and 4 behave differently, in that they approach one at a constant linear rate. We note that this may be with respect to the geometrical layout of the scenes themselves. These two were more open environments, thus shorter paths are calculated on average and hence the probability of detecting an intruder may not be as high as compared to the other scenes, which have more geometry, around which an agent may have to navigate. This seems to highlight that geometrical features of an environment also contribute to the difficulty, and this is something we may look at in future work. We also clustered results per scene based on difficulty ranges. Again, a subset of these results are presented in Figure 4. Figure 4a represents the results pertaining to Scene 0. As expected, for high levels of difficulty (0.8-1.0),



(a) Scene 0



(b) Scene 3

Figure 4: Measure 2 values for scenes across all patroller counts into heuristic value ranges

there is the highest levels of failures. For lower levels of difficulty, the simulated player performs better, winning a higher ratio of games. It is worth noting that this scene has a high level of occluding geometry, and most rounds were classified with the highest level of difficulty according to our heuristic. Figure 4b reflects the results for Scene 3. For the easiest level (0.0-0.2) the simulated player performs well, winning 40% of its games. As the difficulty increases, the player performs less effectively. From the results of that scene we see a more even distribution of measure 2 values across all difficulty ranges. Again, this captures some of the nuances of the actual physical layout of the scene and its more open nature.

One feature of these results is that we have been able to show that across different scenes we can highlight a useful guide of varying difficulty levels on a per scene basis. Some environments will be most difficult with 4 patrollers, and adding a 5^{th} may not add any extra complexity not already captured; whereas other scenes may require more patrollers to achieve maximum difficult levels.

Finally, in Figure 4, we present results detailing the level of failure and success of the intruder across the different

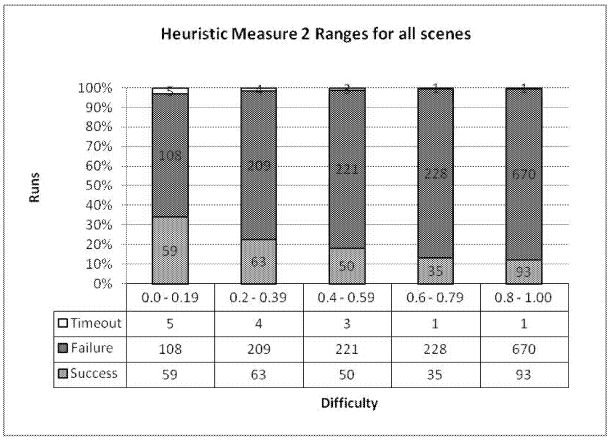


Figure 5: Clustering across all scenes and all patroller counts for measure 2, expressed as a percentage of the number of rounds within each interval

scenes for the difficulty level as recorded using measure 2. This highlights that overall, measure 2 is very closely related to the success of the player, such that, as the value for the measure increase, so to do the number of intruder failures.

CONCLUSIONS

In this paper we discussed our research effort into evaluating a set of simple heuristically guided measures that could be used to approximate the difficulty of a game world for an agent to overcome. For this task we have presented a framework based on the patroller-intruder game. We showed empirically that measures based on patroller count, path length and approximated perception abilities could be used to give a good prediction for the performance of a naive simulated player. Moreover, we highlighted that simple measures could inherently capture features pertaining to the physical layout of the environment and its influence on the difficulty of a level. In future work we wish to further explore the set of viable heuristics, including the number of paths for the intruder, the deceptive nature of paths and to learn a means to optimally combine a set of these measures to provide a strong indicator of difficulty. We would also like to tackle the experiment set again with a larger set of simulated players of varying abilities.

REFERENCES

Amigoni F.; Basilico N.; Gatti N.; Saporiti A.; and Troiani S., 2010. *Moving Game Theoretical Patrolling Strategies from Theory to Practice: An USARSim Simulation.* In *ICRA 2010, International Conference on Robotics and Automation.* IEEE Press.

Basilico N.; Gatti N.; and Rossi T., 2009. *Capturing augmented sensing capabilities and intrusion delay in patrolling-intrusion games.* In *CIG'09: Proceedings of the 5th international conference on Computational Intelligence and Games.* IEEE Press, Piscataway, NJ, USA. ISBN 978-1-4244-4814-2, 186–193.

Castillo T. and Novak J., 2008. *Game Development Essentials.* Delmar Cengage Learning, Clifton Park, N.Y.

Costello F. and O'Riordan C., 2009. *An Approach to Providing Feedback at the Design Phase in Game Authoring Tools.* In L. Breitlauch (Ed.), *Game-On 2009.*

Hunicke R., 2005. *The case for dynamic difficulty adjustment in games.* In *ACE '05: Proceedings of the 2005 ACM SIGCHI International Conference on Advances in computer entertainment technology.* ACM, New York, NY, USA. ISBN 1-59593-110-4, 429–433. doi:http://doi.acm.org/10.1145/1178477.1178573.

Smith G.; Treanor M.; Whitehead J.; and Mateas M., 2009. *Rhythm-based level generation for 2D platformers.* In *Proceedings of the 4th International Conference on Foundations of Digital Games.* ACM, New York, NY, USA, FDG '09. ISBN 978-1-60558-437-9, 175–182. doi:http://doi.acm.org/10.1145/1536513.1536548. URL `http://doi.acm.org/10.1145/1536513.1536548`.

Spronck P.; Ponsen M.J.V.; Sprinkhuizen-Kuyper I.G.; and Postma E.O., 2006. *Adaptive game AI with dynamic scripting. Machine Learning*, 63, no. 3, 217–248.

Spronck P.; Sprinkhuizen-Kuyper I.G.; and Postma E.O., 2004. *On-line Adaptation of Game Opponent AI with Dynamic Scripting. Int J Intell Games & Simulation*, 3, no. 1, 45–53.

Togelius J.; De Nardi R.; and Lucas S.M., 2007. *Towards automatic personalised content creation for racing games.* In *CIG'07: Proceedings of the 3rd international conference on Computational Intelligence and Games.* IEEE. URL `http://cogprints.org/5573/1/Togelius2007Towards.pdf`.

Yannakakis G.N. and Hallam J., 2006. *Towards Capturing and Enhancing Entertainment in Computer Games.* In G. Antoniou; G. Potamias; C. Spyropoulos; and D. Plexousakis (Eds.), *SETN.* Springer, *Lecture Notes in Computer Science*, vol. 3955. ISBN 3-540-34117-X, 432–442.

ACKNOWLEDGEMENTS

The authors would like to acknowledge the support of the Irish Research Council for Science, Engineering and Technology (IRCSET) for their assistance through the Embark Initiative.

A GOAP ARCHITECTURE FOR EMERGENCY EVACUATIONS IN SERIOUS GAMES

César García-García, Laura Torres-López, Victor Larios-Rosillo and Hervé Luga
CUCEA-Postgraduate and Research Department
Universidad de Guadalajara
Periférico Norte 799, Guadalajara
Mexico
cesar.garciagarcia@acm.org

KEYWORDS
Serious Games, Behavioral Animation, Crowd Simulation, Distributed Virtual Environments, First Responders

ABSTRACT

This work presents the current development of a Goal-Oriented Action Planning Architecture to be used in the DVRMedia2 Framework AI Engine. The purpose of this research is to model complex behaviors to be used for massive simulations, with thousands of virtual characters acting in a congruent manner in emergency situations, where first responders and civilian population could be senzibilized and trained.

INTRODUCTION AND CONTEXT

When modeling non-player character behavior for any game, one way to obtain realism is the use of action planning schemes (Funge et al. 1999, Hoang et al. 2005, Orkin 2006, Panzoli et al. 2008) that incorporate perception of the environment and goal selection.
In this context, we are dealing with a serious game where emergency evacuations can be simulated to sensibilize and train the general civilian population as well as the first responders.

Behavior modeling

To effectively model crowd behavior, it is necessary to first model individual behavior that is coherent with the simulated virtual environment (Thalmann et al. 2000). In order to behave in a believable way these artificial actors must act according to their surrounding environment, being able to react to environmental changes and also to the actions of human-operated avatars in the virtual world (Ulicny and Thalmann 2001).
Reactive behaviors, such as adapting to the changes in the environment, are not enough to create realistic autonomous actors. Creating realistic behaviors also demands the representation and manipulation of the perceived information (Shao and Terzopoulos 2005). This is what we call *cognitive behaviors* and once integrated to the reactive ones, we can truly speak of complex behavior.
Once the individual behaviors are realistic, a multi-agent system could be used to replicate the individual into crowds, this however, creates an homogeneous crowd behavior that might or not appear realistic.

Psychological Factors

Psychological or *human* factors in behavior that have been considered for crowd simulations are part of the artificial actors' *internal state* which provides certain degree of individuality and also reflects differences in perception and emotion, such as fear (Pelechano and Badler 2006), hunger (Shao and Terzopoulos 2005), tiredness (Edward et al. 2009), etc.
These variations provide the actors with realistic behaviors under emergency conditions, enabling the simulation of panic situations where real people are unable to act due to fear. Also, certain behaviors are tied to specific internal states, limiting their execution to predefined levels of said state. In this manner we can simulate an stampede when fear overcomes the crowd.

CONCEPTUAL BRAIN MODEL

The conceptual model for the artificial brain shown in Figure 1 includes the following modules:

Perception Module

The perception module contains the information that is available to the artificial actor about his environment through his senses. This includes what the actor can "see" –such as the place where he is and threats present– and what he can "hear" –such as other actors in the same area.
To effectively store this information it is divided into three different substructures: region information contained in the database (Torres-Lopez and Larios-Rosillo 2009); Voronoi neighbors (Martinez-Vargas et al. 2009); and perceived threats (Pelechano and Badler 2006).

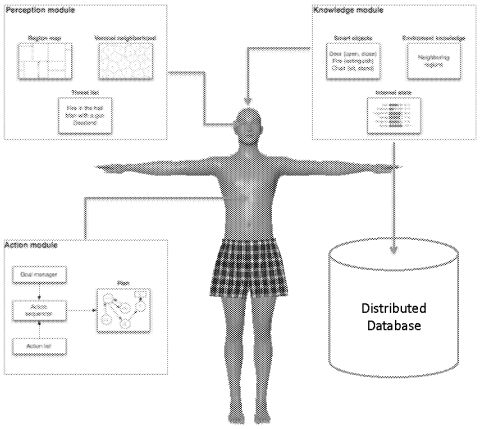


Figure 1: Brain Architecture

The perception module represents what is available to the senses all the time and as such it gets updated every simulation cycle with fresh information. In later parts of this work the perception module will also handle virtual actor communication, such as sharing threats with other actors and acquiring knowledge from them.

Knowledge Module

The knowledge module contains the information about the world that has already been internalized by the actor, as well as his own *internal state*. This also includes a set of *smart objects* of which the actor is aware and the regions to which the actor can move from the present one.
Smart objects announce themselves to the environment, as well as the actions associated with them. Those actions are then added to the individual actor's *action lists* and are available to the Action Module for their execution.
Knowledge of adjacent regions is also stored as knowledge, as it is not always available to the actor's senses, so he must "remember" where he has been as well as the places he can move to.

Internal State
The internal state deserves a more detailed view as it is, along with the individual action lists, the basis of individual actor behavior. The internal state represents the status of the actor as a series of values that influence its behavior.
Certain specific goals and actions will only be included in plans when the actor's internal state reaches certain thresholds, for example, a character will panic when his *fear* has reached a certain value.
There is also a relationship or hierarchy in which the internal state's values are to be recalculated, as proposed by (Edward et al. 2009). This relationship, called *behavioral network*, also reflects what internal state values influence other parts of the artificial brain.

Action Module

The action module handles the decision-making processes of the virtual actor. This includes assigned or desired goals which the actor must accomplish; a list of actions available to accomplish the goal and an action sequencer or *planner* that combines actions to fulfill goals. Once a valid action sequence or *plan* has been created, it is stored for its step-to-step execution.
A *goal* is any condition that the actor wants to satisfy (Orkin 2003), or a specific state of the virtual world, which can be composed as a set of conditions, that the actor needs to reach. For example, the `SecureArea` goal implies that the virtual actors must eliminate any threats to the best of their abilities, that is, using only the actions available in their action lists.
The action lists, as per (Orkin 2004), represent the different actions available to a Fireman, a Trained Civilian and a –implicitly untrained– Civilian. When the `SecureArea` goal is fed to the Action Sequencer each different actor will create a different plan containing their available actions:

- Fireman: The fireman will include his `AttackFire` action to repeatedly attack the fire until it is put out or another goal takes precedence.
- Trained Civilian: The trained civilian will include his `AttackFireOnce` action to use an extinguisher on the fire, then evacuate orderly.
- Untrained Civilian: The untrained civilian, having no available actions to deal with the fire, will use his `Run` action to secure himself, as he is unable to secure the area.

In the specific case of the Fireman, his action plan might first need to find a fire extinguisher, go to it, grab it and then come back to the fire and use the actions made available by the extinguisher smart object to attack the fire and satisfy his `SecureArea` goal.

CURRENT WORK

We are currently using the JMonkey Engine (Powell 2003) to develop a serious game where we can simulate evacuations of massive structures such as stadiums, shopping malls, government offices, industrial complexes, etc. While JME is a good choice for our graphical needs, an additional communications layer, based in the JADE (Bellifemine et al. 2003) is also being implemented. This layer will also allow for the distribution of the goal-management and plan generation aspects of the conceptual brain model.
Our results show (??) that even when running 2500 agents, the graphical performance did not suffer from quality loss, and that the real limit was the Virtual Machine's inability to launch additional threads.

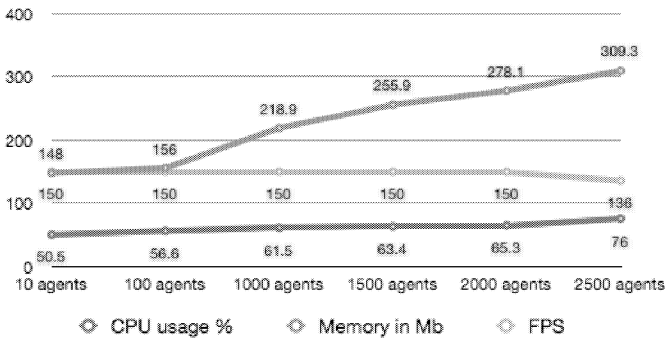


Figure 2: CPU, Memory and FPS performance during simulation

CONCLUSION AND FUTURE WORK

We find the current results promising and will continue to develop in this direction, adding modules as part of the agent's behaviors as they are developed and using the obtained values to optimize the maximum number or agents that should be running per peer in a large scale simulation.
The generated behavior model and AI engine could be used, as part of the DVRMedia2 framework, to simulate evacuation of massive structures such as stadiums, shopping malls or education facilities in the region holding over 1000 personnel.

ACKNOWLEDGEMENTS

This project is funded by CONACYT and COECYT-JAL grants through Universidad de Guadalajara. The DVRMedia2 framework is a cooperative effort between Universidad de Guadalajara and Institut de Recherche Informatique de Toulouse (IRIT), France.
Special thanks to Intel Guadalajara for providing the required high-performance computing support and to Unidad Estatal de Protección Civil y Bomberos Jalisco for providing required training and access to facilities.

REFERENCES

Bellifemine F.; Caire G.; Poggi A.; and Rimassa G., 2003. *Java Agent DEvelopment Framework: A White Paper.*

Edward L.; Lourdeaux D.; and Barthes J.P., 2009. *Cognitive Modeling of Virtual Autonomous Intelligent Agents Integrating Human Factors. IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology*, 353–356.

Funge J.; Tu X.; and Terzopoulos D., 1999. *Cognitive Modeling: Knowledge, Reasoning and Planning for Intelligent Characters. International Conference on Computer Graphics and Interactive Techniques.*

Hoang H.; Lee-Urban S.; and Muñoz-avila H., 2005. *Hierarchical plan representations for encoding strategic Game AI.* In *In Proc. Artificial Intelligence and Interactive Digital Entertainment Conference (AIIDE-05.* AAAI Press.

Martinez-Vargas M.P.; Larios-Rosillo V.; and Torguet P., 2009. *DVRMedia2 P2P Networking Strategies to Support Massive Online Multiuser Virtual Environments.* Internal report, Universidad de Guadalajara. URL `http://dvrmedia.wikispaces.com/file/view/4WITmmartinez.pdf`.

Orkin J., 2003. *AI Game Programming Wisdom 2.*

Orkin J., 2004. *Symbolic Representation of Game World State: Toward Real-Time Planning in Games. AAAI Workshop.*

Orkin J., 2006. *Three States and a Plan: The A.I. of F.E.A.R. Game Developers Conference.*

Panzoli D.; Luga H.; and Duthen Y., 2008. *A Reactive Architecture Integrating an Associative Memory for Sensory-Driven Intelligent Behavior. Intelligent Virtual Agents*, 5208, 528–529.

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

Powell M., 2003. *JMonkeyEngine.* URL `http://www.jmonkeyengine.com/home/`.

Shao W. and Terzopoulos D., 2005. *Autonomous pedestrians.* In *SCA '05: Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation.* ACM, New York, NY, USA. ISBN 1-7695-2270-X, 19–28. doi:http://doi.acm.org/10.1145/1073368.1073371.

Thalmann D.; Musse S.R.; and Kallmann M., 2000. *From Individual Human Agents to Crowds. Informatique*, 1, 6–11.

Torres-Lopez L. and Larios-Rosillo V., 2009. *DVRMedia2 P2P Database System to manage coherence in massive multiuser online games and virtual environments.* Internal report, Universidad de Guadalajara. URL `http://dvrmedia.wikispaces.com/file/view/LauraTorres_v3.pdf`.

Ulicny B. and Thalmann D., 2001. *Crowd simulation for interactive virtual environments and VR training systems. Europgraphics Workshop.*

A CONSTRAINED GROWTH METHOD FOR PROCEDURAL FLOOR PLAN GENERATION

Ricardo Lopes[1], Tim Tutenel[1], Ruben M. Smelik[2], Klaas Jan de Kraker[2], and Rafael Bidarra[1]

[1]Computer Graphics Group, Delft University of Technology, Delft, the Netherlands
[2]Modelling, Simulation and Gaming Department, TNO, The Hague, the Netherlands

KEYWORDS

Virtual worlds, procedural modeling, floor plan generation

ABSTRACT

Modern games often feature highly detailed urban environments. However, buildings are typically represented only by their façade, because of the excessive costs it would entail to manually model all interiors. Although automated procedural techniques for building interiors exist, their application is to date limited. One of the reasons for this is because designers have too little control over the resulting room topology. Also, generated floor plans do not always adhere to the necessary consistency constraints, such as reachability and connectivity. In this paper, we propose a novel and flexible technique for generating procedural floor plans subject to user-defined constraints. We demonstrate its versatility, by showing generated floor plans for different classes of buildings, some of which consist of several connected floors. It is concluded that this method results in plausible floor plans, over which a designer has control by defining functional constraints. Furthermore, the method is efficient and easy to implement and integrate into the larger context of procedural modeling of urban environments.

INTRODUCTION

Game worlds increasingly often feature highly detailed open worlds. Notable recent examples include Assassin's Creed, GTA IV and Oblivion, where players can explore beautiful cities, of which each building has been modeled by hand. Manual modeling of these city models involves an enormous amount of effort, putting a huge burden on the budget for game development companies. In particular, this is the reason that, in games, city buildings either have no interior (they consist only of a façade) or a fixed set of interiors is repeated all over the city. Urban environments are therefore one of many areas where automated procedural generation methods can excel, by providing a limitless amount of varied content for a fraction of the cost.

The procedural generation of a city entails a number of ingredients, each with its specific procedures and generation techniques: district topology (city center, suburbia), road network (ring road, streets), division of open space into parcels (building lots), building façades, floor plans (room layouts) and interior furniture (chairs, tables). In this paper, we focus on an important if somewhat neglected ingredient: floor plans. We propose a novel and flexible technique for generating procedural floor plans subject to user-defined constraints. Using these constraints, game designers control both the topological layout of the building (*e.g.* which room comes next to which other room) and the areas of the room. Furthermore, the method guarantees reachability of all rooms and over multiple floors.

The remainder of this paper is structured as follows. We first survey previous work on procedural modeling of urban environments, focusing on floor plan generation. Then we present our constrained growth method for procedural floor plans. We show several example building interiors generated by this technique. And last, we discuss our method advantages and limitations, and propose future extensions and potential applications.

RELATED WORK

Procedural modeling techniques have been proposed for almost every aspect of virtual worlds, ranging from landscapes to buildings. An extensive survey of procedural modeling techniques can be found in Smelik et al. (2009). Regarding buildings, *L-systems*, previously applied to plant models (Prusinkiewicz and Lindenmayer 1990), were among the first automated techniques used for building façades (Parish and Müller 2001). In recent years, more specialized rewriting systems have been presented for this purpose: the *split grammars* (Wonka et al. 2003) and *shape grammars* (Müller et al. 2006).

Here, we focus on procedural techniques for building floor plans, *i.e.* the creation of a suitable layout of the different rooms inside a building. Greuter et al. (2003) create a floor plan as a combination of 2D shapes. However, this floor plan is only used for extruding building façades.

Shape grammars, typically applied to building façades, can also create floor plans, as shown by Rau-Chaplin et al. (1996) in their *LaHave House project*. Their system

generates a library of floor plans, each of which can be customized and automatically transformed into assembly drawings. It uses shape grammars to create a *plan schema* containing basic room units. Possible groupings of individual units are recognized to define functional zones like public, private or semi-private spaces. After generating the required geometric data, such as the room unit dimensions and the location of walls, a specific function is assigned to each room. The rooms are filled with furniture, by fitting predefined layout tiles from an extensive library of individual room layouts.

Martin (2006) proposes a graph-based method, in which nodes represent the rooms and edges correspond to connections between rooms (*e.g.*, a door). This graph is generated by a user-defined grammar. Starting from the front door, public rooms are added. Each of these public rooms is assigned a specific function (dining room, living room, etc.). Subsequently, private rooms are attached to the public rooms and, finally, stick-on rooms like closets or pantries are introduced. This graph is transformed to a spatial layout, by determining a 2D position for each room. For each node, depending on the desired size of the room, a specific amount of "pressure" is applied to make it expand and fill up the remaining building space.

Hahn et al. (2006) present a method tailored for generating office buildings. Starting from the building structure, they first position all elements that span across multiple floors, *e.g.* elevator shafts, staircases. Then, the building is split up into a number of floors. On each of them a hallway subdivision is applied, adding straight hallway segments or rectangular loops. Next, the remaining regions are subdivided into rooms, for which geometry is created and appropriate objects are placed. A notable feature of this system is that each step is executed just in time: based on the player's position, floors and rooms are generated or discarded. A discarded room can be restored exactly in its previous state, by re-using the same random seed in the procedure, followed by re-applying all changes to its objects, caused by the player.

Marson and Musse (2010) introduce a room subdivision method based on *squarified treemaps*. Treemaps recursively subdivide an area into smaller areas, depending on a specific importance criterion. Squarified treemaps simultaneously try to maintain a length to width ratio of 1. Their methods input is the basic 2D shape of the building and a list of rooms, with desired dimensions and their functionality, *e.g.* social area, service area and private area. First these functionality areas are added to the treemap. These areas are again subdivided to create each room. Possible connections between room types are pre-defined; based on this, doors are placed between the rooms in the generated floor plan. Using the A* algorithm, a shortest path is determined, visiting all rooms that need a connection with the corridor. This path is transformed into a corridor, and all rooms are adjusted to make room for it.

Tutenel et al. (2009) applied a generic semantic layout solving approach to floor plan generation. Every type of room is mapped to a class in a semantic library. For each of such semantic classes, relationships can be defined. In this context, constraints typically define room-to-room adjacency, however, other constraints can be defined as well, e.g. place the kitchen next to the garden, or the garage next to the street. For each room to be placed, a rectangle of minimum size is positioned at a location where all defined relation constraints hold, and all these rooms expanded until they touch other rooms.

A general overview of the application of constraint solving in procedural generation can be found in (Tutenel et al. 2008). Particularly, several approaches define the creation of room layouts as a space planning problem. Charman (1993) gives an overview of constraint solving techniques that, although not specifically focused on space planning, can be applied to these problems. He compares several space planners and discusses the efficiency of these solvers. Due to many recent improvements on constraint solving techniques, his efficiency concerns are no longer relevant. However, the discussed planners are still suitable for layout solving. For instance, the planner he proposed (Charman 1993) works on the basis of axis-aligned 2D rectangles with variable position, orientation and dimension parameters, for which users can express geometric constraints, possibly combined with logical and numerical operators.

These constraint solving techniques create rectangular shapes, subject to dimension and adjacency constraints. However, they are typically quite complex and can still be time consuming. Moreover, many of them cannot handle irregular shapes, *e.g.* L-shaped or U-shaped rooms, which incidentally also holds for many of the other techniques discussed above. This is one of the main drawbacks of all these approaches. Our growth-based approach generates more flexible results, and does that faster than other constraint solving techniques.

FLOOR PLAN GENERATION METHOD

This section discusses a novel method for procedurally generating floor plans. The problem of generating floor plans is primarly a problem of generating the appropriate layout for rooms, *i.e.* their location and area. We believe that a proper procedural floor plan is one that follows the same principles (and, as a consequence, the appearance) of real building architecture. Our method has drawn its inspiration from real life architectural floor plans, where geometric grids are used as a canvas for hand drawing building interiors. Fig. 1 illustrates a real architectural floor plan example, where a point grid is visible. Our method uses a grid-based algorithm for placing and growing rooms within a building layout. We describe how this algorithm hierarchically creates a floor plan by generating building zones followed by its room areas, both of which are constrained by adjacency and connectivity relationships.

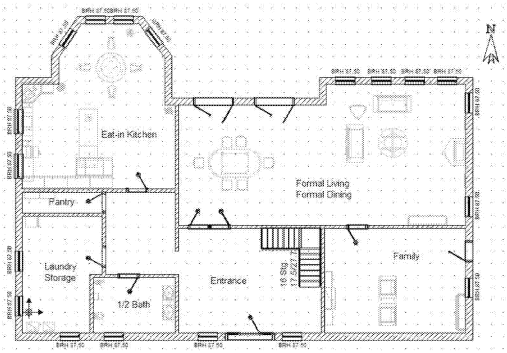


Figure 1: Example of a real architectural floor plan, using a geometric point grid

Hierarchical layout

The building layout is defined in a hierarchic manner. Certain types of rooms can be grouped, allowing the building to be subdivided in zones for these room groups, followed by subdividing these zones into sub-zones, or into specific rooms. This approach is similar to the method of Marson and Musse (2010), which was discussed above. An example of a simple hierarchic building layout is shown in Fig. 2. In this example, we subdivide a house into two zones: a public zone and a private zone. The public zone is then subdivided into a dining room, a kitchen and a living room and the private zone into a hallway, two bedrooms and a bathroom. This entails that the room generation algorithm is applied twice, once for each level in the hierarchy. Note that this hierarchy can of course have more than two levels.
Because of the hierarchic approach, we do not allow adjacency constraints to be defined between two rooms from different zones. We can however put adjacency constraints between a room and a parent zone, *e.g.* in the previous example, we can demand that the hallway from the private zone is adjacent to the public zone to allow a connection with this public zone through the hallway.

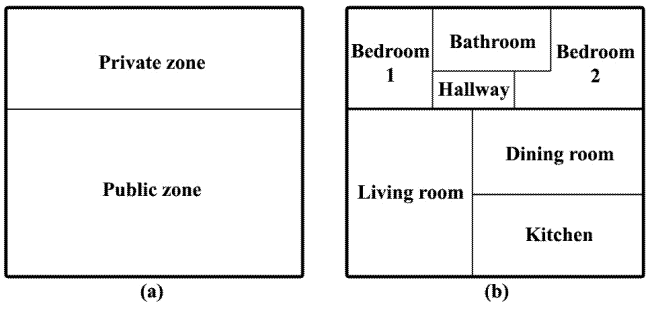


Figure 2: This is an example of a hierarchic subdivision of a building. (a) shows a building subdivided into two areas, (b) shows these areas subdivided into their required rooms.

Room placement

Our method uses a grid as the basis for the room placement and expansion process. There are two advantages in using this representation. Firstly, it maps closely to the way architects design floor plans. Secondly, it is efficient to traverse and expand in a 2D grid, which allows our floor plan generation method to execute fast. Our grid representation (a matrix) does not limit a floor to a rectangular shape. We fit non-rectangular buildings in a grid that includes space marked as outdoor.
The method starts with a building footprint polygon, which we rasterize into the 2D grid. Additional input includes:

- the grid cell dimension in meters (*e.g.* 0.5m);
- a list of room areas to generate;
- for each room area its corresponding house zone (public, private, hallway) and its preferred relative area (or size ratio);
- adjacency and connectivity constraints.

The output it produces is a floor plan, consisting in room areas, interior and exterior walls, and doors.
In the room placement phase, the initial positions for each room are determined. For this, we create a separate grid of weights of the same dimensions as the building grid. Initially, cells inside the building get a weight of one and cells outside get a weight of zero. Many of the rooms in a building are preferably placed next to an outer wall. Therefore, to apply this constraint, the weights are altered. A straightforward way to do this would be to set weights of cells not connected to the outside to zero. However, placing initial positions adjacent to a wall does not always result in plausible results, as they tend to cause less regular room shapes. Therefore, we use a different approach. Based on the size ratio of the room and the total area of the building, we can estimate a desired area of the room. Cells positioned at least a specific distance (based on this estimated area) away from the walls are assigned a weight of 1. This results in much more plausible room shapes.
This phase also deals with the defined adjacency constraints. The adjacency constraints are always defined between two rooms, *e.g.* the bathroom should be next to the bedroom, the kitchen should be next to the living room, etc. When selecting the initial position of a room, we use the adjacency constraints to determine a list of rooms it should be adjacent to. We check whether these rooms already have an initial position. If there is, we alter the weights to high values in the surroundings of the initial positions of the rooms it should be adjacent to. This typically results in valid layouts; however there is a small chance that the algorithm grows another room in between the rooms that should be adjacent. To handle this case, we reset the generation process if some of the adjacency constraints were not met.

Based on these grid weights, one cell is selected to place a room, and the weights around the selected cell are set to zero, to avoid several initial positions of different rooms to be too close to each other.

Room expansion

We use a growth-based method for determining the precise shape of rooms. Our algorithm gradually grows rooms from initial room positions until the building interior is filled. The expansion process is done by appending adjacent grid cells to rooms.
Algorithm 1 outlines the expansion of rooms in our method. It starts with a grid m containing the initial positions of each room. It then picks one *room* at a time, selected from a set of available *rooms* (`SelectRoom`), and expands the room shape to the maximum rectangular space available (`GrowRect`). This is done until no more rectangular expansions are possible. At this point, the process resets *rooms* to the initial set, but now considers expansions that lead to L-shaped rooms (`GrowLShape`). In a final step, the algorithm scans for remaining empty cells and assigns them to a nearby room (`FillGaps`).

Algorithm 1: Room expansion algorithm

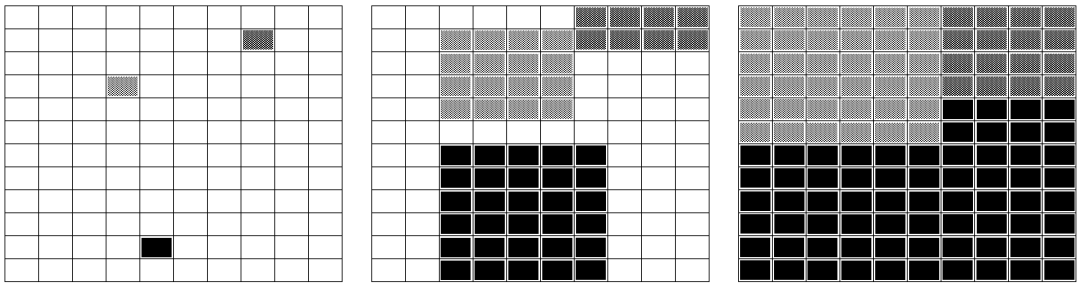
```
in  : A list of rooms to be placed l
in  : A list of room area ratios r
in  : A building grid m
out : A floor plan defined in m

rooms ← BuildRoomSet(l);
while rooms ≠ ∅ do
    room ← SelectRoom(rooms, r);
    canGrow ← GrowRect(room, m, r[room]);
    if ¬canGrow then
        rooms \ {room};
    end
end
rooms ← BuildRoomSet(l);
while rooms ≠ ∅ do
    room ← SelectRoom(rooms, r);
    canGrow ← GrowLShape(room, m);
    if ¬canGrow then
        rooms \ {room};
    end
end
if HasEmptySpaces(m) then
    FillGaps(m);
end
```

In `SelectRoom` the next room to be expanded is chosen on the basis of the defined size ratios r for each room. The chance for a room to be selected is its ratio relative to the total sum of ratios, defined in r. With this approach, variation is ensured, but the selection still respects the desired ratios of room areas.

The first phase of this algorithm is expanding rooms to rectangular shapes (`GrowRect`). In Fig. 3 we see an example of the start situation (a) and end (b) of the rectangular expansion phase for a building where rooms black, green and red have size ratios of, respectively, 8, 4 and 2. Starting with rectangular expansion ensures two characteristics of real life floor plans: (i) a higher priority is given to obtain rectangular areas and (ii) the growth is done using the maximum space available, in a linear way. For this, all empty line intervals in the grid m to which the selected *room* can expand to are considered. The maximum growth, *i.e.* the longest line interval, which leads to a rectangular area is picked (randomly, if there are more than one candidates). A room remains available for selection until it can not grow more. This happens if there are no more directions available to grow or, in the rectangular expansion case, if the room has reached its maximum size. This condition also prevents starvation for lower ratio rooms, since size ratios have no relation with the total building area. In Fig.3 (b), all rooms have reached their maximum size.
Of course, this first phase does not ensure that all available space gets assigned to a room. In the second phase, all rooms are again considered for further expansion, now allowing for non-rectangular shapes. The maximum growth line is again selected, in order to maximize efficient space use, *i.e.* to avoid narrow L-shaped edges. In this phase, the maximum size for each room is no longer considered, since the algorithm attempts to fill all the remaining empty space. Furthermore we included mechanisms for preventing U-shaped rooms. Fig. 3 (c) illustrates the result of the L-shaped growth step on the previous example. The final phase scans the grid for remaining empty space; this space is directly assigned to the room which fills most of the adjacent area.
With the described mechanism, our growth algorithm generates valid floor plans. Even with all the constraints, each room layout problem has many valid solutions. Therefore, there is quite some variation in generated floor plans. Since our grid-based algorithm is fast enough, we can collect multiple alternative solutions for each situation. We execute our growth algorithm N (parametrizable) times, obtaining N different but valid floor plans. We can either randomly select one of alternatives, or we can evaluate and rank the solutions (by *e.g.* preference for smaller hallways or the least amount of corners in room), and pick the best solution. Post-processing will convert the grid to a geometric representation of rooms, and ensures the creation of doors and windows or extrusion of walls.

Room connectivity

Our method ensures connectivity between rooms by procedurally placing inner doors. This process is more elaborate than simply making all rooms reachable from *e.g.* the entrance of the building. For plausible results,



(a) (b) (c)

Figure 3: Example for the room expansion algorithm: (a) initial room positions (b) rectangular growth (where rooms reached their maximum size) (c) L-Shape growth.

room connectivity needs to influence and be influenced by the topology of the building. We achieved this using connectivity constraints combined with a specialized connectivity algorithm.

Connectivity constraints, declared as input, state that two rooms should be directly connected. Stating that room A should be connected to room B will create a door that opens from room A to room B. These constraints influence the room layout, as they are also interpreted as adjacency constraints. After placement and expansion of rooms, we proceed to a post-processing phase, in which, among other things, our connectivity algorithm is executed. For this, the grid representation is no longer appropriate, as it is more natural to consider inner walls rather than room areas. Therefore, in this post-processing phase matching inner walls for the rooms are created first.

The connectivity algorithm places doors in these walls, influenced by the building topology and connectivity constraints. It should ensure the common connectivity principles of a building (*e.g.* hallway always leads to adjacent public rooms), while the constraints should only be used to declare specific cases. Our aim is to emulate real life architecture in its basic connectivity concepts, independent of style or type of buildings. Therefore, we use the notion of private and public rooms to decide on door placement. The goal is to create full connectivity, while respecting room privacy.

Our algorithm starts by placing doors between rooms for which connectivity was explicitly declared. Next, it connects any hallway to all of its adjacent public rooms. Unconnected private rooms are then connected, if possible, to an adjacent public room. Publics rooms with no connections are connected to an adjacent public room as well. Finally, our last step is a reachability test. We examine all rooms and if any is not reachable from the hallway, we use the adjacency relationships between rooms to find a path to the unreachable room, and create the necessary door(s).

With this algorithm, public rooms (and in particular the hallway) have priority to become focal points, in terms of connectivity to private rooms. We wanted to bring about the situation where, for example, two adjacent private rooms are connected through a common public room, instead of having a direct door between them. The hierarchical layout ensures rooms distribution in such a way that this setup is possible.

The creation of an inner door is implemented in a simple manner. First, we select a shared wall between the two rooms to connect, in which the door fits. Next, we randomly select a segment of that wall and place the door in its midpoint. Finally, doors are assigned to open towards the destination room with its hinge towards the closest corner of the wall.

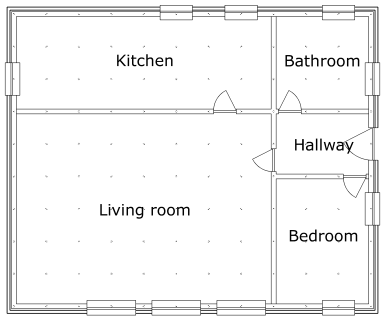


Figure 4: Example of a relatively simple floor plan including four rooms and a hallway.

RESULTS

This section provides examples of floor plans generated with our method. The first example is a relatively simple rectangular building (11 x 9 m), for which no hierarchy is defined, and which should contain four rooms (a bedroom, a bathroom, a kitchen and a living room) and a hallway. We defined a connectivity constraint between the kitchen and the living room. An example of a generated floor plan for this building is shown in Fig. 4. Notice that the connectivity constraint is satisfied, and that all rooms are reachable. Generating these kind of floor plans takes less than 100 ms.

Figure 5: Two examples of a more complex floor plan, built up of a public zone with a great room and kitchen and dining room, a private zone with a master bedroom and walk-in closet, master bathroom and a private zone with a hallway, a bathroom and two bedrooms.

In our second example, we show more complex floor plans, inspired from real floor plans of North-American style villas. The building (which is L-shaped, measuring 17 x 20 m with a top part of 7 x 9.5m and a bottom part of 17 x 10.5 m) is subdivided into three areas: one public and two private zones. There are adjacency constraints defined between the public area and the two private areas. The public area includes a great room and a room for a kitchen and a dining area. These rooms have a connectivity constraint defined. The first private area includes a master bedroom with an adjacency constraint to the public area, a master bathroom and a walk-in closet both connected to the master bedroom. The second private area includes a hallway with an adjacency constraint to the public area and three other rooms (a bathroom and two additional bedrooms), all connected to this hallway. Two floor plan examples for this building are shown in Fig. 5.

One clearly notices L-shaped areas not only for the individual rooms but also for the different areas within the floor plan. This creates more varied and less restrictive results than methods that are limited to rectangular rooms. Because of the increased complexity these floor plans take longer to generate: on average around one second per floor plan.

In our method, we included a simple mechanism for placing exterior elements, *i.e.* outer doors and windows. These wall elements are declared as requirements for each room. They are placed using the same method as for inner doors. The results show that these elements are placed correctly, if there is space available. In the future, it would be interesting to consider generating these elements automatically.

Multiple floors are also supported by our method. When generating the ground floor, a staircase (or elevator) room is generated. In the subsequent floors, this room is duplicated, fixed to the same position as the floor below. Adjacency and connectivity constraints for staircase rooms still hold as in normal rooms. Our results show that room expansion in higher floors is not affected by the initial duplication of the staircase. Rooms expand naturally around the staircase, respecting all constraints. An example of a two floor house is shown in Fig. 6.

Discussion

We discussed how our floor plan generation method is suitable for procedural building generation, since it provides fast and plausible floor plans. However, there is some room for improvement. Firstly, our connectivity algorithm identifies between which two rooms a door needs to be placed, but the actual location of the door is at random within the selected wall segment. This sometimes creates situations where the walk paths inside the building are not optimal. A smart approach for positioning these doors would be a helpful extension.

Another potential issue arises from the fact that we currently can not constrain the dimensions of a specific room. Our approach often creates L-shaped rooms, which might not be desirable in some particular cases. An example of this is a garage, which needs to have specific dimensions such that a car can be efficiently parked. An L-shaped garage makes not much sense in that respect. Constraints on the width-to-height ratio of a particular room could improve the methods ability for handling these cases.

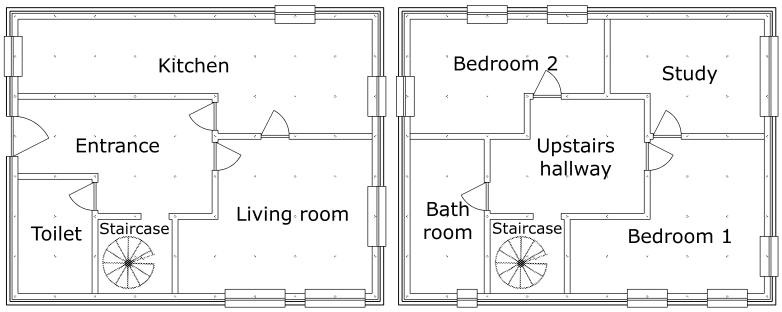


Figure 6: An example floorplan of a two floor house. Downstairs (left of the figure) an entrance, connected to a toilet, a staircase and a kitchen and living room, upstairs, the staircase is repeated and is connected to a hallway which in turn is connected to two bedrooms (one of which is connected with a study) and a bathroom.

Since we use a grid-based algorithm, buildings with arbitrary angles are not supported. However, there are straightforward extensions for such cases as well. As in the technique used by Greuter et al. (2003), a building's outer shape can be created by combining multiple geometric shapes. If all of these basic shapes would be constrained to axis-aligned polygons (with an arbitrary rotation in reference to the buildings orientation), these parts could be hierarchically subdivided into separate areas with our approach, and later combined into one complete building.

CONCLUSIONS

We can conclude that the method presented here efficiently generates valid and plausible building floor plans. We validate it in two ways. First, we visually compared floor plans generated by our method with many real-world plans of North-American houses, with which they typically matched well. However, although the method is very suitable for generating floor plans of houses, for highly regular structured spaces, such as some types of office buildings, more simple and specialized subdivision methods would be more efficient and effective. Second, we have consulted with architects and other experts on the output generated, and received valuable feedback. This has already resulted in further tuning of the method's parameters and constraints, for example on plausible adjacencies and desirable floor topologies. In the near future, we also expect this feedback to lead to the introduction of additional consistency tests.

An area which we could improve upon is the scoring mechanism for rating the set of alternative layouts generated by the method. This score could be a combination of the adjacency constraint satisfaction with a number of heuristics, *e.g.* minimum amount of internal doors, minimum amount of internal wall segments, etc.

Our implementation of the method performs well, due to its low complexity and efficient data structures. Although it is by far not optimized, it is still suitable in the context of dynamic generation of virtual worlds, where, *e.g.*, the interior of procedural buildings is generated within a small frustum around the player.

As future work, we would like to embed this method in the broader context of generating fully featured procedural cities. For this, we would have, among other things, to integrate the method with a façade generation system, resulting in complete buildings. Also, by using our semantic layout solving approach (Tutenel et al. 2009), we could fill the procedural interiors with appropriate furniture.

In our view, the constrained growth floor plan generation method is a small but significant step towards the goal of realistic and full-featured procedural cities.

REFERENCES

Philippe Charman. Solving Space Planning Problems Using Constraint Technology. In *Nato ASI Constraint Programming: Students' Presentations, TR CS 57/93, Institute of Cybernetics, Estonian Academy ofSciences, Tallinn, Estonia*, pages 80–96, 1993.

Stefan Greuter, Jeremy Parker, Nigel Stewart, and Geoff Leach. Real-time Procedural Generation of 'Pseudo Infinite' Cities. In *GRAPHITE '03: Proceedings of the 1st International Conference on Computer Graphics and Interactive Techniques in Australasia and South East Asia*, pages 87–94, New York, NY, USA, 2003. ACM. doi: http://doi.acm.org/10.1145/604471.604490.

Evan Hahn, Prosenjit Bose, and Anthony Whitehead. Persistent Realtime Building Interior Generation. In *Sandbox '06: Proc. of the ACM SIGGRAPH Symposium on Videogames*, pages 179–186, New York, NY, USA, 2006. ACM. doi: http://doi.acm.org/10.1145/1183316.1183342.

F. Marson and S.R. Musse. Automatic Generation of Floor Plans Based on Squarified Treemaps Algorithm. *IJCGT International Journal on Computers Games Technology*, 2010. Accepted for publication.

Jess Martin. Procedural House Generation: a Method for Dynamically Generating Floor Plans. Research Poster presented Symposium on Interactive 3D Graphics and Games, 2006.

Pascal Müller, Peter Wonka, Simon Haegler, Andreas Ulmer, and Luc Van Gool. Procedural Modeling of Buildings. In *SIGGRAPH '06: Proceedings of the* 33^{rd} *Annual Conference on Computer Graphics and Interactive Techniques*, pages 614–623, New York, NY, USA, 2006. ACM.

Yoav I. H. Parish and Pascal Müller. Procedural Modeling of Cities. In *SIGGRAPH '01: Proceedings of the* 28^{th} *Annual Conference on Computer Graphics and Interactive Techniques*, pages 301–308, New York, NY, USA, 2001. ACM.

P. Prusinkiewicz and Aristid Lindenmayer. *The Algorithmic Beauty of Plants*. Springer-Verlag, New York, NY, USA, 1990.

A. Rau-Chaplin, B. Mackay-Lyons, and P. Spierenburg. The LaHave House Project: Towards an Automated Architectural Design Service. In *Proceedings of the International Conference on Computer-Aided Design (CADEX)*, Hagenberg, Austria, September 1996.

Ruben M. Smelik, Klaas Jan de Kraker, Tim Tutenel, Rafael Bidarra, and Saskia A. Groenewegen. A Survey of Procedural Methods for Terrain Modelling. In *Proceedings of the CASA Workshop on 3D Advanced Media In Gaming And Simulation (3AMIGAS)*, Amsterdam, The Netherlands, June 2009.

T. Tutenel, R. Bidarra, R.M. Smelik, and Klaas Jan de Kraker. The Role of Sematics in Games and Simulations. *ACM Computers in Entertainment*, 6:1–35, 2008. doi: http://doi.acm.org/10.1145/1461999.1462009.

Tim Tutenel, Rafael Bidarra, Ruben M. Smelik, and Klaas Jan de Kraker. Rule-based Layout Solving and its Application to Procedural Interior Generation. In *Proceedings of the CASA Workshop on 3D Advanced Media In Gaming And Simulation (3AMIGAS)*, Amsterdam, The Netherlands, June 2009.

Peter Wonka, Michael Wimmer, François Sillion, and William Ribarsky. Instant Architecture. In *SIGGRAPH '03: Proceedings of the* 30^{th} *Annual Conference on Computer Graphics and Interactive Techniques*, pages 669–677, New York, NY, USA, 2003. ACM.

ACKNOWLEDGEMENTS

This work was supported in part by the Portuguese Foundation for Science and Technology under grant SFRH/BD/62463/2009. This work has also been supported by the GATE project, funded by the Netherlands Organization for Scientific Research (NWO) and the Netherlands ICT Research and Innovation Authority (ICT Regie).

BIOGRAPHY

Ricardo Lopes is a PhD student from the Delft University of Technology. His current research interests include: adaptivity in games, player modeling, interpretation mechanisms for in-game data and (on-line) procedural generation techniques. Ricardo got his master's degree in Information Systems and Computer Engineering at the Technical University of Lisbon, in Portugal.

Tim Tutenel is a PhD candidate from Delft University of Technology, participating in a research project entitled "Automatic creation of virtual worlds". His research focus is on layout solving, object semantics and object interactions. In particular, he is researching how semantics can improve procedural generation techniques. Tim got his master's degree in Computer Science at the Hasselt University in Belgium.

Ruben Smelik is a scientist at the TNO research institute in The Netherlands. He is a PhD candidate on a project entitled "Automatic creation of virtual worlds", in close cooperation with Delft University of Technology. This project aims at developing new tools and techniques for creating geo-typical virtual worlds for serious games and simulations. Ruben holds a masters degree in computer science from the University of Twente.

Klaas Jan de Kraker is a member of the scientific staff at the TNO research institute. He holds a PhD in computer science from Delft University of Technology. He has a background in computer-aided design and manufacturing, collaboration applications, software engineering (methodologies), meta-modeling and data modeling. Currently he is leading various simulation projects in the areas of simulation based performance assessment, collective mission simulation, multifunctional simulation and serious gaming.

Rafael Bidarra is associate professor Game Technology at the Faculty of Electrical Engineering, Mathematics and Computer Science of Delft University of Technology, The Netherlands. He graduated in electronics engineering at the University of Coimbra, Portugal, and received his PhD in computer science from Delft University of Technology. He leads the research line on game technology at the Computer Graphics Group. His current research interests include: procedural and semantic modeling techniques for the specification and generation of both virtual worlds and game play; serious gaming; semantics of navigation; and interpretation mechanisms for in-game data. He has published many papers in international journals, books and conference proceedings, and has served as member of several program committees.

ARTIFICIAL INTELLIGENCE

PLAYING TETRIS USING LEARNING BY IMITATION

Dapeng Zhang, Zhongjie Cai, Bernhard Nebel
Research Group on the Foundations of Artificial Intelligence
University of Freiburg
Georges-Köhler-Allee Geb. 52 Freiburg
Email: zhangd, caiz, nebel@informatik.uni-freiburg.de

KEYWORDS
Tetris, Learning by Imitation

ABSTRACT

Tetris is a stochastic and open-end board game. Several artificial players were developed to automatically play Tetris. These players perform well in single games. In this paper, we developed a platform based on an open source project for game competitions among multiple players. We develop an artificial player employed learning by imitation, which is novel in Tetris. The imitation tasks of playing Tetris were mapped to a standard data classification problem. The experiments showed that the performance of the player can be significantly improved when our player acquires similar game skills as those of the imitated human. Our player can play Tetris in diverse ways by imitating different players, and has chances to defeat the best-known artificial player in the world. The framework supports incremental learning because the artificial player can find stronger players and imitate their skills.

A. INTRODUCTION

Tetris was first invented by Alexey Pajitnov et al. in 1984, and remains one of the most popular video games today. It can be found in many game consoles and several desktop systems in PC, such as KDE and GNOME.

Tetris is a stochastic and open-end board game. A piece of block is dropped from the top of the board. The piece is randomly chosen from seven predefined ones, and it falls down step by step. The player can move and rotate the current piece to place it in a proper position. A new piece appears at the top of the board after the current one touches the ground. A fully-occupied row will be cleared and the blocks above it will automatically fall down one step. The goal of the game is to build as many such rows as possible.

Two players can compete against each other in Tetris. When one player places the current piece to clear n rows, the other player will receive an attack of $n-1$ rows, each of which contains $n-1$ empty cells. The attacks are pushed into the game board from the bottom, raising all the accumulated blocks up $n-1$ steps in the board. The player who has no more space to accommodate the next piece loses the game.

The single Tetris game was used as a test-bed in the research in artificial intelligence. Researchers developed artificial players using different approaches [1]. Fehey created a hand-coded player [2], Böhm et al. employed genetic algorithms [3], and Szita et al. used cross-entropy methods in Tetris [4]. These players can play the single game, clearing hundreds of thousands of rows, which would take several weeks or even months for a human player.

The competition in Tetris is certainly an interesting topic. In theory, the two-player Tetris is much more complex than the single one [5]. Assuming both human and the artificial player handle the piece with the same speed, human players can defeat the best artificial player with ease in the competition mode. To our knowledge, the existing artificial players cannot create many attacks in the competitions. The researchers evaluate their players mainly in single games.

Imitation is essential in social learning [6]. Assuming the similarities between the observations and themselves, humans acquire various skills via imitation. Imitation learning can be applied in robotics and automatic systems in several ways[7]. For instance, Billard et al. built a system according to the structure of the human brain [8]. Atkeson et al. developed a method to explain the actions of a demonstrator, and to use the explanations in an agent [9].

This paper was motivated by building an artificial player for the competitions in Tetris. As a human is superior in the competitions, we employed learning by imitation to clone the game skills of human players. The highlights of this paper can be summarized as follows:

- We developed an open source platform for the competitions.
- To our knowledge, learning by imitation is novel in Tetris.
- Our artificial player can acquire diverse game behaviors by imitating different players.
- Our player has chances to defeat the best-known artificial player in the competitions.
- The framework supports incremental learning.

This paper is structured in the following manner: first, the relation between this work and the literature is addressed in next Section. Then, an open source platform for Tetris competitions is introduced in Section -B. Next, a method is developed to map the imitation to a standard data classification problem in Section -C. After that, the performance of the developed methods is shown in Section -D. Finally, we draw the conclusion and discuss the future works in Section -E.

Related Works

Learning by imitation has been widely applied in robotics,

especially in humanoid robots [8]. The core idea of imitation is to improve the similarity between the imitated system and the imitator, even if certain physical or virtual dissimilarities exist. In this paper, a framework is developed to imitate both human and artificial players. The structure of our approach is certainly different from human brains or the models of the other artificial players. Generally, we follow the idea of learning by imitation. To our knowledge, it is the first time that imitation learning has been applied in Tetris.

The single Tetris games have been used as test-beds in several branches in artificial intelligence [1]. For example, the standard 10×20 Tetris game is still a challenging task for the methods in reinforcement learning [10] [11]. The number of rows that a player can clear is widely accepted as a criteria for the evaluation. So far, several successful artificial players e.g. in [3], [4], and [2], are based on building an evaluation function with linear combinations of the weighted features. These features were listed in [1]. We also employ 19 hand-coded features in our approach, some of which cannot be found in the list. Instead of a linear evaluation function, we use multiple support vector machines in our framework.

Support Vector Machine (SVM) was first proposed by Cortes and Vapnik in 1995[12], and became an important method for data classification. SVM is well-developed. I was implemented in several open source packages which were available in Internet. In this paper, SVM is used as a tool. Our implementation is based on LIBSVM [13]. We modeled the imitation tasks in Tetris as a standard data classification problem which can be finally solved by SVMs.

Incremental learning is mainly about a series of machine learning issues in which the training data is available gradually [14]. It is a special learning method with which a certain evaluation can be improved by the learning process during a fairly long period. In order to do that, we defined a learning paradigm: switching attention learning [15]. In the paradigm, there are multiple learners with their inputs and outputs forming a loop. The performance of one learner generates potential improvement space for the others. Following this idea, Tetris is used as a test-bed. Our artificial player can choose a game played by a stronger player as its target to imitate.

B. AN OPEN SOURCE FRAMEWORK FOR TETRIS

KDE [1] is an advanced desktop platform which provides user-friendly graphic interface. It is an open source project. KBlocks is the Tetris game in KDE. We developed KBlocks to a platform for researches in artificial intelligence. The system components of KBlocks is shown in Figure 1.

KBlocks can be run in two modes: KDE users can use it as a normal desktop game; researchers can choose to start a game engine, a GUI, or a player. The GUIs and the players are connected to the game engine via UDP sockets. The components can be run in one or several computers.

KBlocks can be configured with parameters defined in a text file. The game engine (and the GUI) supports game

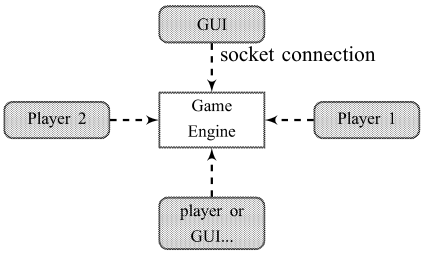


Fig. 1. The System Components of KBlocks

competitions among up to 8 players, in which one player could be a human. A hand-coded artificial player is integrated [16]. It can clear on average $2000-3000$ lines in single games. The competitions can be done in a synchronized mode, in which each player gets the new piece after the slowest player finishes the current placement.

A new artificial player can be integrated into the platform with ease. We provide a source code package in Internet [2], in which the class KBlocksDummyAI is a clean and simple interface for the further development. Graduate students can simply change the source code for their internship or thesis. Researchers can play around with some ideas or organize competitions.

C. LEARNING BY IMITATION

In Section -B, we addressed the functionality of the Tetris platform. In this section, the learning by imitation is discussed in details. First, we give a brief introduction to the system components. Then, the patterns, which are used in the filters and support vector machines (SVMs), are explained. Last, we address how the SVMs are used for data classification in our imitation learning.

The training data of the imitation learning are obtained from the imitated system. In this paper, they are the Tetris games played by the imitated player. We created several models to obtain the skills of the imitated player. The training process receives positive feedback if the models make the same decision as the imitated system. Otherwise, it receives the negative feedback. The imitation learning is successful if the trained models keep the similarity even if the data never appear in the training set.

The learning system consists of several components, as shown in Figure 2. We created three catalogs for these components: the data representation; the algorithms; and the learners. They are illustrated as the gray rectangles, the regular rectangles, and the round-cornered rectangles in the figure.

Figure 2 also shows the relations among the components. We align these components vertically according to the catalogs. A lower algorithm uses the outputs of the upper one as its inputs. The learners computes the models which are used in the algorithms.

The middle column with the dotted arrow lines shows the sequence of the computation in the games. With the current

[1] official cite: http://kde.org

[2] http://www.informatik.uni-freiburg.de/~kiro/KBlocksDummyPlayer.tgz

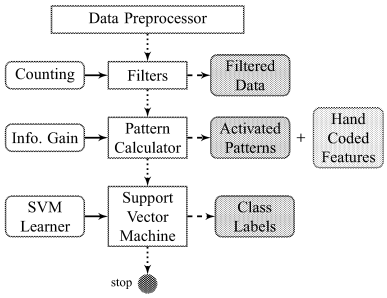


Fig. 2. System Components

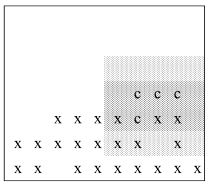


Fig. 3. An example of the Pattern

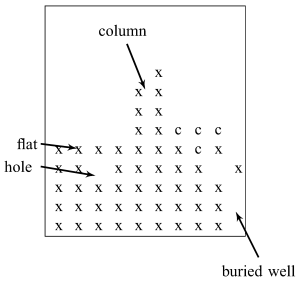


Fig. 4. The Illustrations of the Features

board state and the piece (s, p), the data preprocessor can generate up to 34 candidate placements by enumerating all the rotation and the position of the p. The candidates are filtered because of the heavy computational power required by training the SVMs. The rests of the candidates are passed to the pattern calculator and the hand-coded features. Each candidate is transferred into a vector of the values of the patterns and the features. The vectors are used as the input of the SVM for the prediction. The output of the SVM can be described as how similar a candidate is to the choice of the imitated player. Consequently, the most similar one is labeled as the final choice.

The Learning of the Patterns

Training the SVMs is time consuming. There are 7 different pieces in Tetris: L, J, O, I, T, Z, and S. To place one of L, J, or T, there will be 34 candidates by combining all the possible rotations and positions; O has 9 combinations; I, Z, or S have 17. The candidate chosen by the imitated player is regarded as the positive case. The others are the negative cases. If the size of the training set is 10000, there are about 220000 tuples (cases) in the set. If each tuple is a vector of 39 values, training a SVM from these data would take more than a week using a $2.3GHz$ PC.

In order to reduce the data set, the types of pieces are used in the data preprocessor to separate the data into 7 subsets. Each subset is used to train its own filter, patterns, and SVM. In other words, seven SVMs work together in the artificial player.

When placing the current piece, human players can first reduce the candidates to a limited number by observing the surface of the accumulated blocks. Then, they choose one from the filtered candidates as their final decision. This idea was used to develop the filters for reducing the amount of data in the learning.

A filter consists of a set of patterns. Figure 3 shows the concept of the patterns. The current piece is denoted by 'c', it is an 'L' in the figure. We use 'x' to denote the already occupied cells. Around the placement, a small field, which is marked in gray, is chosen as the activated area for the patterns. The patterns are smaller than the small field. For example, the deeper gray area in the figure shows a pattern. It contains 5×2 cells. The cells with a 'c' or 'x' inside are occupied.

A pattern can be activated by a placement. As mentioned above, the small field is activated by the placement. All the 5×2 patterns can be enumerated. We move a pattern around the small field. It is activated by the placement, if the occupied cells in the pattern match the occupied cells in the background (the small field).

Filters can thus be learned by counting. If a pattern is never activated by the placements of the imitated player, it can be used to reduce the candidates. Each filter is a set of such patterns. It can be learned by running the activation tests over all the training data.

Support Vector Machines

The patterns are useful not only in the filters but also for modeling the skills of the imitated players. For instance, a pattern was activated 1000 times over the training set, among which 900 were activated by the positive cases. This pattern cannot be used in a filter because there are mixed negative and positive cases. However, activating it apparently indicates that the placement tends to be positive because of the positive to negative rate in the training data. Therefore, the patterns are also used in this section to compute the inputs of the SVMs.

However, the patterns can only get the "local" information. They are checked within the small field around the placement. From another aspect, it is important to consider some "global" parameters in Tetris. For example, a candidate placement can clear 4 rows. This would be important for the game. The patterns, however, cannot express this occurrence.

We designed hand-coded features to acquire "global" information. If the patterns can define the tactics of the games, the features can be used to describe the strategies. In order to define these features, we use Figure 4 to illustrate some phrases: *hole, flat, column*, and *well*. A well or a hole is buried if it is no deeper than three cells from the surface.

The features are listed in Table I. Items 2 and 3 are for the column. Items 4 – 6 are about the flat. 9 – 11 are for the hole. 14 – 18 are about the well. Our features are compared

TABLE I
LIST OF HAND-CODED FEATURES

1*	How many attacks are possible after the current placement.
2	The number of the columns.
3	The increased height of the column.
4	The increased number of the flat.
5	The decreased number of the flat.
6	The maximum length of the flat
7	The increased height of accumulated blocks.
8	The height difference between the current placement and the highest position of the accumulated blocks.
9	How many holes will be created after the current placement
10	How many holes will be removed after the current placement.
11*	How may occupied cells are added over a hole.
12	The number of removed lines of the current placement.
13*	How well will the next piece be accommodated.
14	If a well is closed by the current placement, how deep is the well.
15	If a well is open by the current placement, how deep is the well.
16*	How may occupied cells are added over a buried well.
17	The number of the open wells.
18	How deep is the well, if it is created by the current placement.
19	Whether a well is removed by current placement.

with the features listed in [1]; the items with * were not mentioned. There are differences in the descriptions of the features because we use them as the inputs of the SVMs. The other researchers developed the evaluation function with the linear combinations of the weighted features.

A large number of patterns can be created by enumeration. For example, an enumeration of 5×2 will create 1024 patterns. It is difficult to consider all these patterns as the inputs of the SVMs because of the required computational power. To our knowledge, there is no trivial way to compute a subset of the patterns which yield to the optimal performance of the SVMs. Therefore, we employ the information gain in decision tree for computing a subset of 20 patterns for each SVM.

SVMs are a popular method in data classification, in which the whole data set are globally classified with a set of the labels. Nevertheless, the data in Tetris are grouped by the current piece. Among the candidate placements of the current piece, the algorithm needs to choose the one which is closest to the choice of the imitated player. LIBSVM [13] provides an API to compute this probability, which is used in our implementation.

The values of the inputs should be within the same range in the SVMs. The patterns always have a value of 0 or 1, which denotes whether or not it is activated by the current placement. The value of features, however, can be much bigger. For example, the maximum length of the flat can be up to 9 in a standard Tetris game. In order to avoid this situation, the values of the features were mapped to 0, 0.5, or 1 in our implementation.

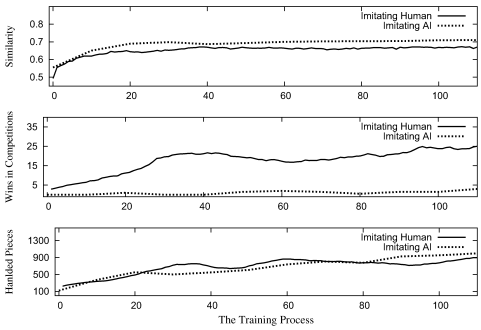


Fig. 5. The Training Process

D. EXPERIMENTS

The experiments were done in a grid system. There are 8 computers in the grid. Each computer has 8 2.3GHz AMD CPUs, and 32G memory. 64 processes can be run in parallel in the grid.

We recorded 10 games of a human player. Each game lasted more than one hour. The game speed was limited, so that the player had enough time for the game. The player can play Tetris at an amateur level. In total 6720 rows were cleared in these games. The human player was regarded as the first imitated player.

The Fehey's artificial player [2] was run for about 1 hour. It cleared 6774 lines without a restart. The game was recorded as the training set. Fehey's artificial player was the second imitated player.

The two imitated players had very different behaviors in the games. If the human player competes with the artificial player in the synchronized mode, the artificial player has very little chance to win, because it attacks only a few times.

The recorded data were divided into 150 subsets, 120 of them were used as the training set. The rests comprised the testing set, through which the similarity between the trained models and the imitated players can be calculated the rate that the trained model chooses the same placements as the imitated player. The results are shown in the upper plot of Figure 5. The data were averaged over 10 slices.

The solid lines show the performance of the player that imitates the human player. The dotted lines are the player that was imitating Fehey's player. Both imitations achieved a similarity of about 0.7. The curves resemble a typical learning curve because the similarity is regarded as the evaluation in the learning. The similarity cannot be higher because of the differences in the data representation and the models between the imitating system and the imitated systems.

The trained models compete against Fehey's player in the synchronized two-player games. 200 random piece sequences were generated for the 200 games, so that each model was evaluated in the same set of the games. The middle plot in Figure 5 shows the winning rates of the imitating players. The player imitating human finally achieved 0.25 as its rate of wins

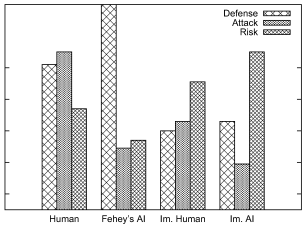


Fig. 6. Behavoiurs of Different Players

in the competitions against Fehey's player. The other imitator did not perform well because the competitions were between the imitating and imitated systems. As the similarity cannot be very high in our implementation, the imitated system should in principle be better than the imitating system.

The trained models also play the single games. The piece sequences used in the games were generated and fixed. The number of handled pieces was used as the evaluation of the player. The results are shown in the lower plot in Figure 5. Fehey's artificial player is better than the human player in the single games, which explains the observation that the imitator of Fehey's player is in the end better than the other imitator.

The training process was designed to search for the maximum rate of the similarity. The rate reached 0.68 at the 30^{th} data slice, and kept this value after that. The performance in the competitions and single games can still be improved after the 30^{th} data slice. This observation indicates that a bigger training set helps to improve the game skills, though it does not improve the similarity in the imitation.

The human player, Fehey's player, and their imitators have different behaviors in the games. In order to show the difference, we designed the evaluations for the attack, defense, and risk. Each player played the same sequences of the pieces in the single games. Attack is the average number of attacks that the player made to clear 100 lines. Defense is evaluated by the average number of cleared lines of each game. Risk is measured by the average height of the placements. The results are shown in Figure 6.

Fehey's player has a defense ability several levels of significance better than the other players. This information was shown as the open-end column in the figure. The other evaluations were mapped to a comparable range. The human player has the best attack ability, which explains how its imitator has chances to defeat Fehey's player in the competitions. The two imitators show quite different behaviors according to the evaluations, which means our imitation learning can generate various artificial players according to the imitated systems.

E. Conclusions

In this paper, we developed a platform for Tetris competitions. The platform is based on an open-source project. The GUIs and players can connect with the game engine via the socket connections. A dummy player was provided as an interface for further development.

We implemented a framework by using learning by imitation. The framework consists of several sets of filters, pattern calculators, and SVMs. The imitation tasks were mapped to a standard data classification problem. The experiments show that our imitators have chances to defeat Fehey's player, which is the best-known artificial player in single Tetris games. And the imitation learning can acquire diverse skills in Tetris games.

There are multiple learners in the framework. The learned artificial player can be used to select an interesting game for further training. The inputs and outputs of the learners form a loop so that each performance of one of the learners create improvement space for the incremental learning.

Discussions

The imitator did not win many games in the competitions. In the next step, we will develop an extra learner for better results in the competitions. The initial experiments showed that the wins in the competitions can be significantly improved by using the rate of wins as the evaluation in the learning.

Tetris was studied mainly in single games. If the sequence of the pieces are known, how can a player win the competitions? AI planning is an interesting direction for further development. We are going to implement the bandit based Monte-Carlo planning in Tetris.

References

[1] S. B. T. Christophe, "Building Controllers for Tetris," *International Computer Games Association Journal*, vol. 32, pp. 3–11, 2009.

[2] C. Fehey, "Tetris AI," http://www.colinfahey.com/tetris/tetris_en.html, 2003, www accessed on 02-August-2010.

[3] G. K. S. M. N. Böhm, "An evolutionary approach to tetris," 2005, in Proceedings of the sixth metaheuristics international conference (MIC2005).

[4] I. A. Lőrincz, "Learning tetris using the noisy cross-entropy method," *Neural Computation*, vol. 18, pp. 2936–2941, 2006.

[5] L. Reyzin, "2 player tetris is pspace hard," 2006, in Proceedings of 16th Fall Workshop on Computational and Combinatorial Geometry.

[6] A. Bandura, "Social learning theory," *New York: General Learning Press*, 1971.

[7] B. S. C. Breazeala, "Robots that imitate humans," *Trends in Cognitive Sciences*, vol. 6-11, pp. 481–487, 2002.

[8] M. M. A. Billard, "Human arm movement by imitation:evaluation of biologically inspired connectionist architecture," *Robotics and Autonomous Systems*, vol. 35, pp. 145–160, 1998.

[9] S. S. C.G. Atkeson, "Learning from demonstration," 1997, pp. 12–20, in Proceedings of 14th International Conference on Machine Learning (ICML97).

[10] K. Driessens, "Relational reinforcement learning," 2004, phD thesis, Catholic University of Leuven.

[11] D. Carr, "Adapting reinforcement learning to tetris," 2005, bachelor Thesis of Rhodes University, Grahamstown 6139, South Africa.

[12] C. Cortes and V. Vapnik, "Support vector networks," *Machine Learning*, vol. 20, pp. 273–297, 1995.

[13] C.-C. Chang and C.-J. Lin, *LIBSVM: a library for support vector machines*, 2001, software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm.

[14] Z. S. A. Bouchachia, B. Gabrys, "Overview of some incremental learning algorithms," 2007, pp. 1–6, in Proceedings of Fuzzy Systems Conference, 2007. FUZZ-IEEE.

[15] A. H. D. Zhang, B. Nebel, "Switching attention learning - a paradigm for introspection and incremental learning," 2008, pp. 99–104, in Proceedings of Fifth International Conference on Computational Intelligence, Robotics and Autonomous Systems (CIRAS 2008).

[16] J. Tang, "Ai agent for tetris," 2009, bachelor Thesis, University of Freiburg, Germany.

EFFICIENT MULTIPLE-AGENT PATH PLANNING IN GRID AND NON-GRID WORLDS

Jürgen Eckerle
Markus Roth

University of Applied Sciences of Berne
Quellgasse 21, CH-2501 Biel, Switzerland
{erj1, rothm23}@bfh.ch

KEYWORDS
Artificial Intelligence, Collaborative Pathfinding, A*

ABSTRACT

Multi-agent pathfinding is a fundamental problem in robotics and computer games. Silvers WHCA* and many other approaches run A* repeatedly for each agent in successive iterations. These algorithms partition the world into a grid where each agent fills exactly one grid element. Collisions are avoided by using a space-time reservation table which maintains the reservations of former iterations. Silver proposed to realize the reservation table by using a hash table. However, that is not suitable either for grid worlds if the moving objects fills more than one grid element or for non-grid worlds.

We present a novel multiple-agent pathfinding algorithm which improves and generalizes Silvers WHCA*, where the reservations are organized as a geometric data structure instead of a hash table. This has two main advantages: 1) The reservation list allows reservations for moving objects with different speeds and sizes and arbitrary move directions without any additional effort 2) Space complexity is limited to O(d·n), where d is the search depth and n is the number of agents. In addition, we present some experimental results in grid and non-grid worlds.

I. INTRODUCTION

A* solves the single-source shortest path problem and is widely used for pathfinding in game programming. It works well if applied to a stationary environment where only a small number of agents act and move independently of each other. However, when applied to a team of simultaneously cooperating agents for pathfinding it suffers from serious drawbacks. For example, when applied to an army of independently acting soldiers crossing a bridge, the usage of A* only leads to very inefficient group behavior. While all the agents follow their own best path individually, they block each other and, as a result, a large number of of waiting and replanning operations are required. Obviously, the group behavior will be improved if all agents follow a global plan that enables them to cooperate.

An optimal solution in this kind a cooperative environment minimizes the maximum cost of all individual paths. A minimal cost path is not a path of minimal length, but a path of minimal simulation time. Of course, a state space search considering all the possible agent moves would solve our optimization problem. However, when we have n agents with b possible moves on average, the search complexity in each search step is given by $O(b^n)$ which is completely intractable for real-time games. More precisely, the problem is proven to be PSPACE-hard (Hopcraft et al. 1984).

We distinguish between *Cooperative Pathfinding*, *Non-Cooperative Pathfinding* and *Antagonistic Pathfinding*. In the first approach, it is assumed that each agent has full knowledge of all the other agents and their planned routes. In the second approach, the agents have no knowledge of each other's plans and in the third approach, each agent tries to reach its goal while preventing other agents from reaching theirs. Another classification is proposed by Latombe (Latombe, 2006) and Fujimura (Fujimura, 1991) which differentiates between two categories of multi-agent pathfinding approaches: centralized and distributed path planning algorithms. A centralized approach takes all agents into account at once, whereas the distributed approach partitions the problem into independent or weakly-dependent tasks. Both our novel pathfinding approach GCA* (Generalized Cooperative A*) as well as Silver's WHCA* (Windowed Hierarchical Cooperative A*) belong to the class of distributed cooperative pathfinding algorithms where the original multiple-pathfinding problem is reduced to a sequence of successively executed single agent A* searches.

In the mean time, many of papers have been published to overcome the space and time limitations of Silver's original cooperative pathfinding algorithm, either by using greedy pathfinding techniques, or additional techniques such as map abstraction and hierarchical search. However, these approaches lead to a loss of information useful for computing shortest paths. To the best of our knowledge, all these approaches are limited to or realized exclusively for grid worlds, where an agent fills exactly one grid element and a move leads to one of the eight grid elements adjacent to the current position. In (Sturtevant, 2009) a more complex motion model is presented. It defines a set of moving objects with different assigned speeds and many move directions. Each move direction is also defined by grid positions and a given movement occupies exactly one grid position. However, our approach is quite different from the other and better suited for non grid worlds.

Our approach keeps the simple structure of Silver's original algorithm, but uses a geometric data structure for

the reservation list which is new in this area. The main advantage of our approach is that reservations can be made for moving objects with different speeds and sizes and arbitrary move directions in an efficient and natural way without any additional effort (with a slight time overhead in maintaining the geometric data structure). Furthermore, our proposed algorithm can be combined with additional techniques such as map abstraction or hierarchical search as easily as Silver's WHCA* (as done in (Jansen and Sturtevant, 2008). Additionally, the space complexity of our approach is limited to $O(d \cdot n)$, where d is the maximal search depth (measured in the number of unit time steps) and n is the number of agents, which is equal to the space complexity for Silver's approach when applied with a hash table. However, the use of a hash table in non-grid worlds has other drawbacks.

The structure of the paper is as follows. In the next section, we present related work including a discussion of Silver's cooperative pathfinding algorithm. In the third section, we present our new path planning algorithm, called GCA* (Generalized Cooperative A*). A complexity analysis and a theoretical comparison between Silver's and our approach follows in the fourth section. In the fifth section, we present our experimental results in a grid and in a non-grid world, comparing our approach with Silver's WHCA*.

II. MOTIVATION FOR OUR WORK.

One of the authors is currently in the process of developing a browser-based, multiplayer real-time strategy game. In this game, each player controls large armies of units of different sizes and speeds that need to independently navigate to their assigned goals while avoiding each other and any obstacles and terrain features that might be in their way. While many games of the genre either simplify the problem by allowing units to pass over each other, or by only allowing a small number of units, for this game the effort will be made to properly solve the problem.

III. RELATED WORK

WHCA*

Silver presented a decoupled multi-agent path planning algorithm best suited for grid worlds called WHCA* (Silver, 2005) and (Silver, 2006). The main idea is to replace a simultaneously executed multiple agent search by a sequence of single-agent A* searches, one for each agent. Collisions among the cooperating agents are avoided completely by using a time-space reservation table for maintaining the time-space reservation pairs done in previous runs. A reservation is given by a triple (x, y, t), which means that the grid position (x, y) is occupied at time t. This way, the other agents are considered obstacles whose positions cannot be occupied at a given time step. Because of the high computation cost, the search is limited by a search window of fixed size T (maximal search depth). When the current search iteration terminates, a (partial) solution path is selected and reserved in the reservation list. Silver proposed to replan the path when the agent traversed half of its planned path, since the situation can change completely in such a highly dynamic environment.

This approach is not without drawbacks, however: A natural, but naive realization of the reservation table by a three-dimensional table (two dimensions for the x- and y-coordinates and one dimension for time) leads to a huge space requirement, more precisely, to $O(d \cdot m)$, where d is the maximal search depth (measured in the number of unit time steps) and m is the number of possible locations in the 2D-world. Using a hash table instead of a simple matrix, also originally proposed by Silver, reduces the space requirement to $O(d \cdot n)$, where n is the number of agents and d is defined as above. Silver claims that his algorithm can be applied to any continuous environment or navigational mesh, but there are drawbacks when the search is applied to non-grid worlds where the agents have different sizes, speeds and shapes. Of course, if a finely-granular grid is used, an arbitrary non-regular shape can be approximated by a set of grid positions (precisely, by those grid positions which have an overlap with the agent's shape). As a result, you either have to compute the set of overlapping grid positions after each move step, or you allow only those moving operations which are defined by a start and a goal position in the grid of a reference element (the set of overlapping grid positions are then given for each other grid element by an offset to the reference element). Additionally, after each move step the entire set of grid-elements approximating the agent's shape has to be checked for a collision with the reservation list.

Other Related Algorithms

A lot of papers were published to overcome the space and time limitations of Silvers WHCA*, either by using greedy pathfinding techniques (Wang and Botea, 2008), (Wang and Botea, 2009), or by using direction vectors to be assigned to each position of the map and to be adapted by a learning process (Jansen and Sturtevant, 2008), or by using additional techniques like map abstraction and hierarchical search (Sturtevant and Buro, 2005) (Sturtevant and Buro, 2006) (Sturtevant 2007). Usually, these techniques lead to a loss of information useful for computing shortest paths. In (Sturtevant 2009) a more complex motion model is used, which handles four different types of moving objects with different speed sizes, move directions motion behavior. However, the movement is also based on a grid as an underlying navigation graph, especially since the move directions are defined by grid positions.

IV. OUR APPROACH

Our key algorithm GCA* works quite similarly to Silver's WHCA*. In both algorithms an independent A* search iteration is done for each agent, limited by a maximal search depth T. As a result, if both algorithms are applied to a grid world and use the same set of move operations (including a waiting operation), then GCA*

generates the search tree in the same way as WHCA*. But there are several relevant distinctions pertaining to the characteristics of the navigation graph, the state representation, the reservations and the structure of the reservation list. For example, we use a geometric data structure for the reservation list (instead of a hash table) which is novel in this context and has two main advantages: First, the original approach can be applied in a simple and efficient manner for any continuous environment. Second, the space requirement can be reduced to *O(d·n),* where *d* is given as above and *n* is the number of agents moving in the virtual environment (Note, *n* is much smaller than *m*).

The main differences between GCA* and WHCA* are as follows: An agent has a shape, instead of being represented by a single grid element. If an agent's move step intends to occupy a new position (x, y), in Silver's approach it suffices to simply check whether the destination position is available or not. In contrast to that, in our approach it is necessary to test whether the agent's shape overlaps with an already existing reservation. The reservation list is organized as a geometric data structure which supports range queries. In our implementation the reservation list is organized as a balanced binary search tree ordered with respect to the x-coordinate. Each of the leaf nodes, which are additionally organized as a doubly linked list, represents exactly one reservation element. The complexity of the data structure is analyzed below.

Classical A:* Recall that A* is a universal problem solving algorithm which can be used either to compute a shortest path from a start to a goal node in any navigation graph or to solve an arbitrary state space problem represented by a problem graph. The nodes of the search graph are maintained in two lists: a) OPEN, which contains the generated, but not yet expanded nodes, and which is initialized by inserting the start node b) CLOSED, which contains all expanded nodes, and which is initialized by the empty list. A* is a best-first search algorithm, that is, in each iteration step the node with minimal cost value (given by a cost function *cost*()) is expanded. OPEN is organized as a priority queue for an efficient access to the minimal element. Classical A* also uses a hash table to avoid duplicates in OPEN and CLOSED and, as a consequence, to avoid traversing cyclic paths. In this context, however, it may be useful to visit the same positions twice (if the agent returns from letting another agent pass). For this reason, we run experiments without spatial duplicate detection.

Assumptions: The branching degree is bounded by b, $b > 0$, that is, each agent has at most b possible, arbitrary selected move directions at any given time step t. Note that the moving steps of the agents may differ by direction or length. Let Δt be the simulation time unit, that is, the simulation time advances in steps of Δt. In our approach, an agent may have an arbitrary shape. The current position of an agent is described by a reference point located within its shape. For the sake of simplicity, we assume that the following conditions hold. First, the agents extension shape is represented by a circle with diameter d_a and its center as reference point. It is easy to determine whether two agents with centers (x_1, y_1) and (x_2, y_2) and diameter d_1 and d_2 overlap: They overlap if and only if the following condition (c1) is fulfilled:

(c1): $(x_1 - x_2)^2 + (y_1 - y_2)^2 \leq \frac{1}{2}(d_1 + d_2)$

Second, we assume that $v_a \cdot \Delta t < d_a$ for each agent *a,* which guarantees that the simulation tic is small enough to avoid head-on passings of two crossing agents. Both of these conditions can be dropped with some additional technical overhead.

State representation: When an agent traverses a navigation graph, the reservations are made step by step along the generated path. A state of the search tree is given by a vector $(a, d_a, (x_a, y_a), v_a, t_a)$, meaning that agent a has a circle shape of diameter d_a and center position (x_a, y_a) and has speed v_a at time t_a. Each problem state *s* is assigned a unique reservation element for the agent's current position. Each reservation is done for the duration of time *Δt.* The reservation list maintains such reservation elements (of course, a space efficient implementation would store the shape information which does not change over time only once for each agent). For the sake of simplicity, we identify the problem state *s* with its assigned reservation element. Additionally, we denote the agent, the diameter, the *x*- and *y*-coordinates, the speed and the time step of reservation *s* by *s.a*, *s.d, s.x, s.y, s.v,* and *s.time*. A node which represents a state *s* with *s.time* = *T* is called a horizon node. Additionally, the generating path of state *s* is denoted by *s.path*.

State transition: A state transition from *s* to *s'* represents a movement of an agent from location $(s.x, s.y)$ to location $(s'.x, s'.y)$. If the move can be made immediately, since there is no collision with another reservation element, then $s'.time$ is set by $s'.time = s.time + \Delta t$. Otherwise, a wait operation is done. During the next time step the possibility of the intended movement is checked again.

Node expansion: To expand a node means to generate all the successor states of a given state s. The set of successor states are generated as follows: First, for each possible move direction a successor state is generated. More precisely, for each (x', y') adjacent to the current position $(s.x, s.y)$ (that is, (x', y') can be reached by a single move step) such that agent a can occupy this location at time $t' = s.t + \Delta t$ without any collision, a successor state s' is generated with $s' = (a, d_a, (x', y'), v_a', t')$, where v_a' is the speed of agent a at time t'. Second, a wait operation is done if the current position can be occupied at time t'. No successor state is ever generated exceeding the given search horizon.

Heuristic and cost function: The cost function used to focus the A* search is defined by

$$s.cost = s.time + \Delta t + h(s, s.goal)$$

for each node *s*, where *s.goal* is the current goal position and *h* the heuristic function. Heuristics, well known in literature and applied in former approaches, are the Manhattan distance, the Euclidian distance and the reverse resumable *A** heuristic. We have implemented all of them. In maze-like worlds, however, the Manhattan distance and the Euclidian distance heuristic proved to be unsuitable both in former approaches and our experiments. The

reverse resumable A* heuristic is an optimal distance heuristic for the single-agent A* search and well-suited, but entails high computational cost (it is called map abstraction in (Silver, 2005)). In most of our experiments, we replaced the resumable A* heuristic by a look-up table precalculated by the Dijkstra algorithm for each pair of grid positions (which yields exactly the same distance values).

The pseudo code description of CGA* is given as follows:

```
GCA*(a /* agent */, T_start, T_limit, RESERVED )
    Begin
(1)   initialize(CLOSED, OPEN, RESERVED);
(2)   best = nil;
(3)   While (Not (OPEN.empty()) )
        /* select state with minimal cost */
(4)     s = OPEN.accessMin();
(5)     If (s == s.goal)
          /* continue search until horizon is reached */
(6)       s.setNextGoal(); /* setup the next goal */
        End;
(7)     If (s.time == T_limit − Δt) {
          /* s is horizon node */
(8)       If (best == nil OR s.cost < best.cost)
(9)         best = s; /* update the best horizon node */
          Endif;
        Elseif { /* search goes on */
(10)      Γ(s) = expand(s);
(11)      changeList(s, Γ(s), OPEN, CLOSED);
          Endif;
      Endwhile;
      /* search terminates */
(12)  If (best <> nil)
      /* make reservations for the best horizon path */
(13)    RESERVED.insert(best.path);
      Else { /* no horizon path has been found */
(14)    End irregularly;
      Endif;
    End;
```

Our algorithm: Our key algorithm GCA* runs a search similar to A* for each agent. The input of the algorithm is the start and end time of the current search iteration, here denoted by T_{start} and T_{limit}, and the reservation list, denoted by RESERVED. For the sake of simplicity, we measure the simulation time in multiples of the time unit Δt.

Step (1) initializes the data structures: 1) it removes all the reservation elements with start time later than T_{start} from the reservation list of the agent, 2) it creates an empty OPEN and CLOSED list, and 3) it creates an initial reservation element for agent a to be inserted into OPEN, here given by $(a, (x_a, y_a), d_a, v_a, T_{start})$.

Step (2) initializes the pointer to the best horizon node.

Step (3) defines the termination condition of the while-loop. The search terminates when OPEN is empty, that is, all partial paths are either proven to be dead ends or have been expanded to the search horizon. This is the same termination condition as in Silver's WHCA*.

Step (4) yields the member of OPEN with minimal cost.

Step (5) and *step (6)*: When a solution has been found, the search cannot stop immediately (an agent cannot disappear). The search must continue until the solution path has reached the horizon, possibly by redirecting the search to the next goal. This behavior ensures that an agent that has reached its does not become a stationary obstacle, even if the agent tries to remain near the goal. The algorithm structure can be kept simple, assuming the heuristic function favors the exploration of a solution path. This is achieved when the heuristic function guarantee $sh(x, x.g) < h(y, y.g)$, if $x.path$ and not $y.path$, contains the goal node.

Step (7) guarantees that only reservations within the given search horizon are made. If the given state s is a horizon node, it may be necessary to update the best solution found using *step (8)* and *step (9)*.

In step (10) the minimal node is expanded. That is, each successor state s' of s is generated and inserted into the search tree (including the computation of *cost*(s')).

Step (11): The *changeList* procedure updates OPEN and CLOSED, that is, the selected and expanded node s is deleted in OPEN and moved to CLOSED, while the set of successors are inserted into the search tree and into OPEN.

If the search has terminated, a check is performed in *step (12)*. If the search horizon has been reached, the generated path to the best horizon node is reserved in the reservation list (*step (13)*). This is done by traversing back the parent links from the generated horizon node and adding each state into the reservation list. Since no exhaustive search has been done, it is possible that such a horizon path is not found for every agent, although they exist. In such cases, the search iteration is ended irregularly (*step (14)*).

V. ANALYSIS

Space complexity: Since there are only n (partial) solution paths saved in the reservation list and each solution path uses at most d reservations, the space requirement of the reservation list (realized by a balanced binary search tree) is given by $O(n \cdot d)$, where d is the maximal search depth and n the number of agents. The space requirement of the data structures used in a search iteration is given by $O(m)$, where m is the number of generated nodes, that is $m = |OPEN| + |CLOSED|$ (|M| denotes the cardinality of M).

Time complexity: GCA* and WHCA* generate exactly the same search tree (if the same tie breaking rule is used) and as a result, the number of node expansions is equal. However, the branching degree of both algorithms is higher than that of the classical A* search, in this context often called spatial A*, since explicit wait actions are needed. For grid-worlds, the time complexity of GCA* is

slightly higher than that of WHCA*, since range queries need to be supported by the reservation list, and this costs more than an access to a hash table. However, the main focus of our algorithm is the non-grid scenario, where WHCA* is not suitable.

Let us now compute the time complexity. First, consider the computational effort of one iteration of the while-loop of CAA*. *Step (4)* takes $\log(m)$ time (m given above). All the other steps with the exception of *step (10)*, the node expansion, use O(1) time. The complexity of *step (10)* needs further consideration. The number of generated successor nodes is limited by $(b+1)$, where b is the maximal number of move directions (one additional successor is given by the waiting operation). For each successor node, the possibility of the intended move step is checked, that is, that there are no collisions with reservations made in previous search iterations and maintained in the reservation list. Let us assume that agent a with diameter d_a intends to occupy move position (x_a, y_a) at time step t_a. A reservation element $(a', d_{a'}, (x_{a'}, y_{a'}), v_{a'}, t_{a'})$ can overlap with this position if and only if $t_a = t_{a'}$ and condition (c1) is fulfilled. We use a weaker condition (c2) to reduce the number of reservations to be checked for overlapping by condition (c1):

(c2) $|x_a - x_{a'}| \leq \frac{1}{2}(d_a + d_{a'})$

The reservation list is organized as a balanced binary search tree with respect to x-coordinates. Let D be the maximal diameter of all moving objects. We have to check only those reservations which satisfy condition (c2). These reservations can be found efficiently by traversing the range $(x_a - D, x_a + D)$ in the reservation list in $O(\log(|\text{RESERVED}|) + k \cdot \log(k))$ time, when k leaves in RESERVED have to be traversed (we need to search for the (x1-D) and then traverse the chain of leaves until x2 is reached). As a result, we have the time complexity $O(\log(m) + k\log(k))$ for one while-loop, assuming that |RESERVED| < |OPEN| and the number of move directions is limited by an upper bound.

Therefore, the overall complexity of GCA* is $O\left(m\left(log(m) + k \cdot log(k)\right)\right)$ instead of $O(m \cdot \log(m))$ for Silver's WHCA* (k should be much smaller than m).

VI. EXPERIMENTAL RESULTS.

We conducted experiments in two different scenarios: A grid world and a non-grid world. In the grid world, each agent a has the same speed value and we assumed that the distance between a pair of adjacent grids is exactly given by $v_a \cdot \Delta t$, where v_a is the assigned speed of an agent a. This assumption does not apply to the non-grid scenario.

We used two grid-world scenario variations for experimental tests, namely S1 in Fig. 1 and S2 in Fig. 2. These are quite similar to those in (Sturtevant and Buro, 2005). We also used a non-grid scenario S3 given in Fig. 5 and 7. Both Silver's WHCA* as well as GCA* traverse the search tree in the same manner when applied to a grid world. Therefore, no experimental comparison between them has been done. S1 has a size of 36 x 64 and S2 has a size 44 x 110. For each experiment we placed a given number of agents randomly on one side of the maze and their target positions on the other side. All agents have equal speed: one grid element per time step. When an agent reaches its goal, the former start position becomes the new target position, and so on. The agents have to patrol ten times between their respective start and goal positions.

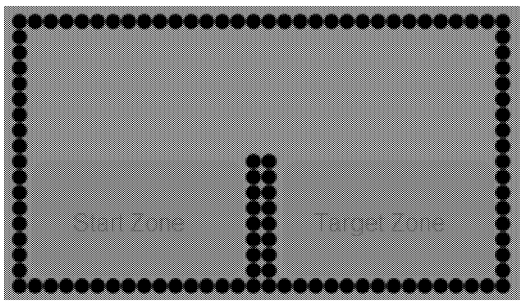


Fig. 1: Map of Scenario 1 with the start and target zone.

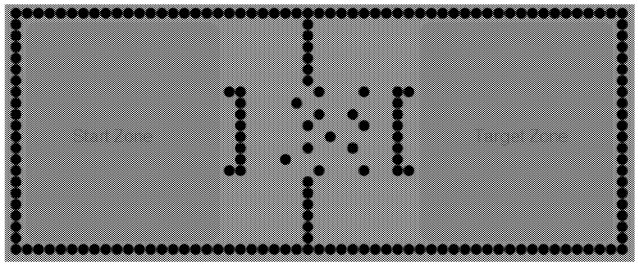


Fig. 2a: Map of Scenario 1 with the start and target zone.

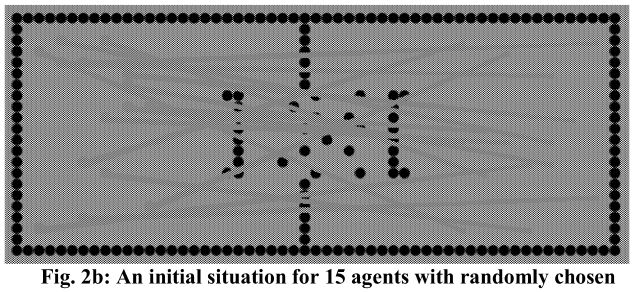


Fig. 2b: An initial situation for 15 agents with randomly chosen start-target pairs (connected by lines).

#Agents	Rounds needed	#Node Expansions	Max. #Open	Max. #Closed
10	699.40	15'424.00	43.40	48.20
15	770.80	25'975.40	59.00	69.60
20	764.40	37'740.40	61.60	70.00
25	793.20	52'152.80	75.00	77.40
30	834.40	70'670.00	89.60	95.80

Fig. 3: Experimental results of CGA* or WHCA*, resp., gained in Scenario 1.

#Agents	Rounds needed	#Node Expansions	Max. #Open	Max. #Closed
10	1'161.40	24'848.40	57.00	62.80
15	1'204.80	39'574.60	75.40	70.40
20	1'271.50	65'296.80	95.00	88.20
25	1'202.00	76'036.60	113.80	81.20

Fig. 4: Experimental results of CGA* or WHCA*, resp., gained in Scenario 2.

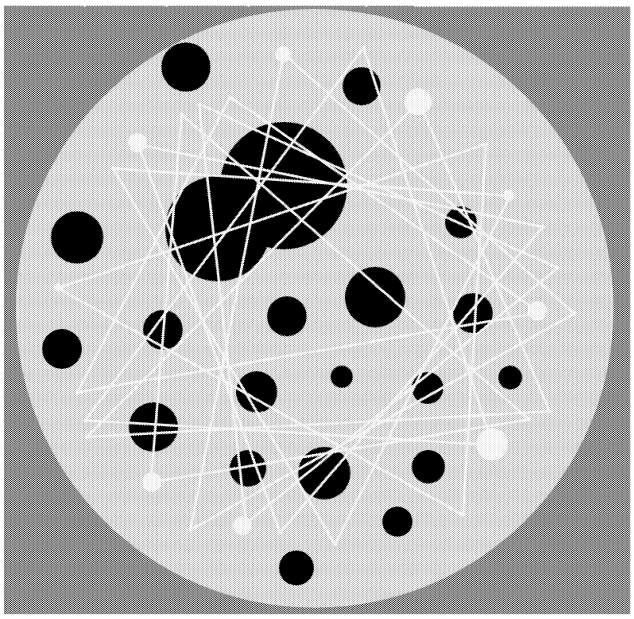


Fig. 5: Scenario 3: Agents of varying in size and speed navigate in a non-grid map. A given number of agents patrol between three goal positions.

In all scenarios we followed both Silver's suggestion of interleaving searches for ordering the agent's A* run and his suggestion of replanning paths after executing the reserved path half-way as described in (Silver 2005) and (Silver 2006). Interleaving means that the start time of the A* iteration is shifted by one time unit with respect to the priority order. That is, the first search iteration starts at time step 0, the second at time step 1, and so on. This distributes the pathfinding calculations to different rounds, and leads to a more stable and better pathfinding. Furthermore, we used a search depth of 10 and as a heuristic function we used the reverse resumable A* heuristic and the pre-calculated look-up table.

We measured the total number of time steps (rounds) needed for all units to reach their respective targets, the total number of node expansions needed, the maximal number of nodes in the closed set at any given time and the maximal number of nodes in the open set at any given time (see Fig. 3 and Fig. 4). The presented results are typical for all the experiments which we conducted with other settings (the heuristic function, the priority order, and the number of units).

A last Scenario 3 in Fig. 5 shows that our algorithm GCA* works in a non-grid world. In a circular field, various circular obstacles of different sizes are located randomly. A given number of agents, varying in size and speed, have to patrol ten times between three given target positions assigned to each agent. The maximal search depth is given by 40 time units. Additionally, re-planning is done after executing the reserved path half-way and interleaving as described above. Previous queries are saved in the cache and can be used for all the agents. As a result, it is possible to re-use previous distance calculations. For the non-grid scenario the pre-calculation of shortest path distance between pairs of locations has to be changed slightly: The pre-calculation is done only for a subset of locations (the locations given on a regular grid). The heuristic distance calculation for an arbitrary pair of locations (l_1, l_2) is given by the minimum distance to the nearest grid location of l_1 and the nearest grid location l_2 and the pre-calculated path distance of both grid positions. We used both the four directions of a square and the six directions of a hexagon given by a local coordinate system to determine the next move positions. Instead of using fixed move directions, the agent's local coordinate system is rotated in such a manner that it is always exactly facing the current goal after each move step (in Fig. 6 named Rotated Square and Rotated Hexagon).

Fig. 6 shows the experimental results. We measured the number of rounds needed to reach the agent's last target position, the total number of node expansions and the total number of node generations. Using a (rotated) hexagon for move directions is much better than using a square. For comparison, the search results for a fixed square are also given. This approach shows that graphs given in a waypoint representation can be searched in the same manner as graphs with an underlying grid structure.

Move Directions by	Rounds needed	#Node Expansions	#Node Generations
Square	4460	66150	298230
Rotated Square	4253	63612	287276
Hexagon	4799	69099	429683
Rotated Hexagon	3950	66902	404050

Fig 6: In these experiments we used the directions of a square and a hexagon which either are fixed or are rotated after each move step in direction to the current goal state.

VII. FUTURE WORK AND OPEN PROBLEMS.

We will determine theoretically and experimentally which geometric data structure is best suited for our reservation list. Perhaps a two dimensional range tree or a

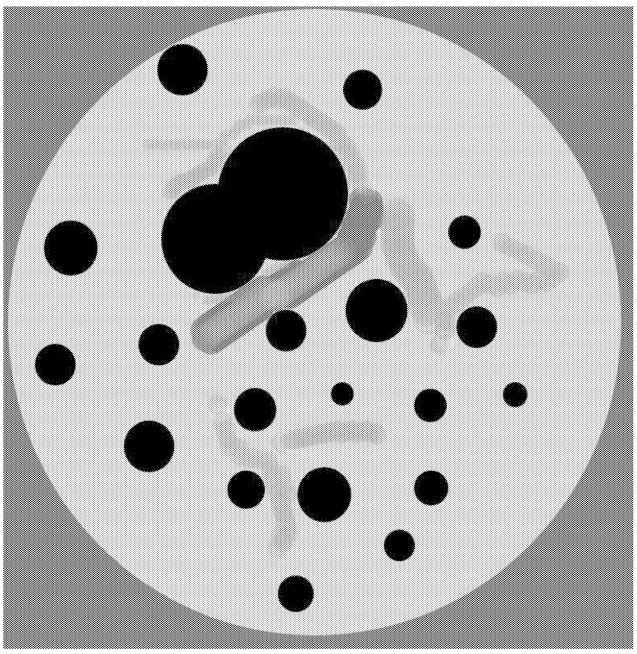


Fig. 7: Here we see a seqeunce of snapshots during the search above.

kd-tree is more efficient than our balanced search tree. Additional experiments should be done.

It is obvious that our approach has great potential for application to non-grid worlds and to waypoint representation techniques. As a result, another main focus is to generalize our approach in such a manner that navigation graphs with arbitrary granularity, smoothing and waypoint navigation techniques can be applied, thus proving it is suited to real-time games.

Path smoothing can be done either by a post-processing process where unnecessary kinks of the solution path are cut or by penalizing change in directions while searching a solution path. In our opinion, both methods of path smoothing can be combined with our approach in a natural manner in contrast to all the former approaches.

Another focus is searching in open terrains instead of maze-like worlds. In (Goodwin et al. 2009) it is stated that the presented A* adaption has potential for searching in open terrains. We are convinced that our approach is much more suitable for searching in open terrains than all the other former approaches including Silver's WHCA*.

VIII. CONCLUSIONS

We have presented a cooperative path planning algorithm for multiple agents. The main structure of the algorithm is quite similar to Silvers WHCA* algorithm. Our approach allows to overcome serious weaknesses of former approaches, including WHCA*: When trying to apply WHCA* to a non-grid environment the possible move directions and the possible speed sizes of the moving objects are always restricted by an underlying grid map. Furthermore, when the number of move directions or the number of speed sizes of the moving objects are increased, the complexity of the underlying moving model increases rapidly, which makes it unsuitable for non-grid worlds. In contrast, our approach uses a geometric data structure, which is able to handle range queries, instead of a hash table or a three-dimensional array for the reservation list. Based on this geometric data structure, the moving objects of the environment may move in an arbitrary direction and with an arbitrary speed size without any additional complexity. For the sake of a time and space efficient geometric data structure, there is only a slight time overhead for maintaining the reservation list. As a result, GCA* is, in contrast to all the former approaches, well suited for continuous environments of non-grid worlds or mesh navigation graphs. It allows each agent to be moved according to its own move directions at each time step.

Since the reservation list supports range queries, a non-colliding move step in any direction can be computed efficiently. The experimental results give some preliminary indication of the potential merit for searching in non-grid worlds and for searching with way-point techniques. Additional experiments need to be done: To find the best suitable geometric data structure, as well as to find the best way to construct an underlying navigation graph or to adapt the waypoint search techniques. Additionally, the possible efficient adaptation of other techniques, such as smoothing, needs to be examined.

ACKNOWLEDGMENTS

This work has been supported by the research program of the University of Applied Sciences of Berne.

REFERENCES

M. Erdmann and T. Lozano-Perez. 1987. "On Multiple Moving Objects". Algorithmica 2, 477-521.

K. Fujimara. 1991. "Motion Planning in Dynamic Environments". Springer, New York.

J. Hopcraft, J. Schwartz and M. Sharir. 1984. "On the complexity of Motion Planning for Multiple Independent Objects: PSPACE-hardness of the warehouseman's problem". International Journal of Robotocs Research 3(4), 76-88.

Renee Jansen and Nathan R. Sturtevant. 2008. "A New Approach to Cooperative Pathfinding". AAMAS (3), 1401-1404.

Renee Jansen and Nathan R. Sturtevant. 2008. "Direction Maps for Cooperative Pathfinding". AIIDE.

J.-C. Latombe. 1991. "Robot Motion Planning". Kluwer Academic Publishers, Boston, MA.

David Silver. 2006. "Cooperative Pathfinding". AI Game Programming Wisdom 3.

David Silver. 2005. "Cooperative Pathfinding". AIIDE, 117-112.

Nathan R. Sturtevant and M. Buro. 2005. "Partial pathfinding using Map Abstraction and Refinement". Proceedings of Association for the Advancement of Artificial Intelligence (AAAI), 47-52.

Nathan R. Sturtevant. 2007. "Memory-Efficient Abstractions for Pathfinding". AAAI.

Nathan R. Sturtevant. 2009. "Optimizing Motion-Constrained Pathfinding". AIIDE.

Ko-Hsin. Cindy Wang and Adi Botea. 2008. "Fast and Memory-Efficient Multi-Agent Pathfinding". ICAPS, 380-387.

Ko-Hsin Cindy Wang and Adi Botea. 2009. "Tractable Multi-Agent Path Planning on Grid Maps". IJCAI-09, 1870-1875.

WEB REFERENCES

S.D. Goodwin, S. Menon, R.G. Price. 2009. "Pathfinding in Open Terrain". http://www.scientificcommons.org/53571097

A HEURISTIC BASED APPROACH TO TEAM BASED BEHAVIOURS IN REAL TIME STRATEGY GAMES

Nigel Burke and Colm O'Riordan
IT Department, National University of Ireland, Galway
University Road,
Galway,
Ireland
E-mail: nigelburke1@gmail.com, colm.oriordan@nuigalway.ie

KEYWORDS
Artificial Intelligence, RTS Games, Teamwork, Heuristcs.

ABSTRACT

This paper explores aspects of high level strategic cooperation and coordination between AI players. A cooperative team-based AI player for the commercial RTS game *StarCraft* is created using team-based heuristics, state machines and influence maps. Effective teamwork inherently improves the performance of teams in team-based games; as a result, the effectiveness of the AI implementation is determined by measuring the improvement in the AI's overall performance when playing team-based games. Various experiments are carried out to test the effectiveness of each team-based behaviour. Results demonstrate that the coordinated and cooperative team of AI players gain a decisive advantage over non-cooperative teams, and that, as expected, teamwork is vital to the success of a team in RTS games. It is concluded from these results that an AI which is capable of team-based cooperation can be successfully created for an RTS using a heuristic based approach.

INTRODUCTION

For games to be totally immersive each aspect of a game must be as realistic as possible and free of noticeable software bugs or flaws that will distract the player from their game experience. An important factor in creating realistic game environments is the game's AI. More research is being undertaken in this area than ever before, and in recent years many advancements have been made in creating realistic AI behaviours (Champandard, 2007).

Despite these improvements, relatively little work has been done on creating truly successful team-based behaviours and teamwork in game AI, especially in the real time strategy (RTS) game genre. RTS AIs are not programmed to play as part of a team. They are typically developed to successfully play 1vs1 matches, and then these AI behaviours are used in team games, with either little or no modifications made to account for the change in gameplay caused by the addition of teams and teammates. Current RTS AIs only view their allies as players that they cannot attack, rather than as players with whom they can or should cooperate (McGee, 2010). The result is a team of AIs who behave as if they have no teammates, usually with an attitude towards self-preservation that rarely results in the AI doing what is best for their team. This often leads to unrealisitic AI behaviours and usually has a negative impact on the game experience/enjoyment of a human player. When human players play RTS team games together they employ a variety of team-based strategies, tactics, and techniques. Cooperative teams of human players will attempt to coordinate their actions so as to gain an advantage over other, less cooperative, teams.

This papers focuses on the development of intelligent team-based behaviours for RTS game AI. More specifically, this paper deals with higher level strategic cooperation, and coordination of each player's overall army (rather than team coordination of individual units or squads within those armies). We hypothesise that team-based heuristics can be identified for successfully playing real time strategy games as part of a team, and that using these heuristics an AI player with intelligent, cooperative, team-based behaviours can be developed for an existing real time strategy game. We attempt to investigate if a team-based AI can cooperate with other players to implement a shared team strategy, and to combine their forces to defeat enemy players. The success of individual team-based behaviours are also analysed.

The layout of this paper is as follows: We begin by presenting a discussion of related research in the areas of team-based game AI. Then, we describe the approach used to test our hypothesis, and the design of our AIs. Following this, a number of experiments and their results are presented and discussed. Finally we draw conclusions from our experimental data and examine potential future work in this area of research.

RELATED WORK

As AI in games became more intelligent, developers started to improve enemy AI by programming them to cooperate with each other and also began introducing AI characters as the player's ally. The enemy AI in *F.E.A.R.* (an FPS game developed by *Monolith Productions* in 2005) implement squad tactics, and actively try to flank the player and employ suppressing fire to help their teammates (Reynolds, 2004). In 1998, a game called *Rainbow Six* revolutionised the sub genre of squad based shooter games and since then more and more squad based games have been emerging (Monteiro & Alvares, 2009). Games such as *Gears of War*, *Left4Dead* and *Killzone* all have single player modes in which the player has AI allies that work with the player. The player completes the game as part of a squad, rather than as an individual. In these three games, AI teammates provide the player (and each other) with covering fire when an ally in vulnerable to attack, heal the player using first aid kits, drag wounded players into cover, and revive fallen team members (Champandard, 2007). Furthermore, Genetic Programming has been used to successfully implement AI teams for squad based shooter games (Doherty and O'Riordan 2009). While

clearly some work has been done in the area of AI teamwork, it has mostly been in game genres where teamwork in inherent in single player game modes (first and third person squad based shooter games), where a lack of competent team based AI would be very noticeable, and may even make a game unplayable. By comparison, very little progress has been made in team-based behaviours for RTS games.

AIs in both *Age of Empires* 2 and 3 communicate with the player via a chat menu, where the player can select a pre-programmed request that they can send to the AI. However, these requests are limited and don't allow true teamwork. The AI can be told to "attack now", but not where or when. This makes cooperation and coordination very difficult. AI players in both *WarCraft 3* and *Age of Empires 3* use a map ping (a flashing alert displayed on an ally's game map at a chosen location) to indicate where it will attack the enemy. However, the AIs don't wait for the human player before attacking. This gives the player no time to coordinate its forces before the AI attacks. While some form of team behaviours have been implemented in commercial RTS games, these features are mostly superficial and don't allow for any true teamwork or co-ordination among teammates.

APPROACH

AIs were designed and implemented for the commercial RTS game *StarCraft*, using the *Brood War Application Programming Interface* (BWAPI). The BWAPI accesses *StarCraft* game state information and allows AI commands to be injected into the game as if a human player were playing.

StarCraft is a commercial RTS game developed by Blizzard Entertainment and released in 1998. The game follows the typical format of an RTS game; each player must collect resources, develop their economy, and then create a military force with which to destroy enemy players. In StarCraft a match is won by destroying *all* of the enemy players' buildings or structures. Once a player has lost all of their buildings their remaining military units become inactive.

Firstly, a single-player AI was developed using the BWAPI. Team-based behaviours were added to the single-player AI to create a cooperative team-based AI. This team AI is capable of coordinating with other players, in all aspects of: attacking the enemy, defending allied bases, team communication and the implementation a shared team strategy. To simplify development and testing: each AI is given complete game state information (i.e. 'fog of war' is disabled).

The AI uses Finite State Machines (FSM) to make decisions. The FSM consists of three main states, these are: Defending, Attacking and Searching. The AIs use influence maps to categorise enemy units and determine the strength of enemy forces in a given area of the game map (such as an enemy base). The AI monitors its overall military strength and searches for enemy bases which are weak enough for its army to successfully destroy. Once a weak base is found, the AI moves its forces to attack that base. Also, the AI has a minimum strength requirement which must be met before it can begin attacking the enemy. If multiple bases are weak enough to be attacked, the AI chooses between them by either attacking the closest base, or the weakest base; this is defined by the strategy the AI is implementing.

When testing team behaviour, if each single-player AI on a team is performing the same hardcoded strategy, then homogenous cooperation may occur accidentally. For example, both AIs may individually decide to attack after three minutes have passed, because both are individually programmed to do so. We will refer to this phenomenon as 'accidental cooperation'. In team games which consist only of human players it is possible for this kind of accidental cooperation to occur; however, for the purposes of testing it was assumed that an uncooperative team never implements the same strategy. To implement this, multiple strategies were created for both the single-player AI and the team AI; the single-player AIs always use two different strategies.

By altering the heuristics used to define a "weak enemy base" and the minimum strength requirement for attacking, three different strategies were created: *aggressive*, *defensive*, and *balanced*. When aggressive, the AI attacks more often and with less units.When defensive, the AI builds up a large force before engaging the enemy. The 'balanced strategy' is somewhere in between.

TEAM-BASED HEURISTICS

The team-based AI was created by adding relevant heuristics to the single-player AI. A critical analysis of existing team-based behaviours in RTS games was performed, and important team-based heuristics were identified for playing as part of a successful, cooperative team in an RTS game.

Team Communication: Almost all aspects of effective teamwork depend on good communication between team members and without it teamwork would be impossible to implement effectively. Players must communicate in order to plan their team strategy, as well as attack and defend together. A custom AI language was developed which the AIs to communicate with each other.

Overall Team Strategy: To be successful a team must share a common strategy and implement it together as a team. This strategy includes each team members army composition, technological advancement, and most importantly, when, where and how to attack the enemy. By working together a team can negate each other's weaknesses, outnumber their enemies in battles, control more territory and operate more efficiently than an uncooperative team. At the start of every game each AI votes for their preferred strategy, in the event of a tie the AIs default to a balanced strategy.

Monitoring of Teammates: Human players monitor their teammates' armies and overall progress. This information is important when coordinating attacks, planning base building locations, and sharing extra resources if an ally needs them. The AIs track and store important information relating to their allies.

Combining Forces/Armies: In team games it is important for teammates to stick together and fight battles together with their allies. This simple aspect of teamwork can often be enough to win a game against an unorganised team. By sticking together, a team can create a 2vs1, 3vs1 or 3vs2 battle scenarios in their favour. AIs communicate when and where to combine their armies.

Attack Co-ordination: When attacking the enemy a team should bring their armies together (as described above).

It is important that all team members initiate the attack at the same time and if necessary retreat at the same time. Each AI communicates when they are ready; once all team members are ready, the attack will begin.

Defence Co-ordination: In a team game, when a player's base is attacked by the enemy, all of that player's teammates will attempt to help defend. It is important for the defending player to communicate that they need help in case their allies don't immediately notice the incoming attack. The AIs monitor their allies territory and army strength to determine when they need assistance.

Spatial Co-ordination Bases: Players should take their allies' locations into account when building new bases and buildings to acquire more resources. Typically players should try to build closer to their allies than their enemies. This makes a joint defence of each base easier as they are closer to allied players. However, players should also avoid building in the same location, as this reduces their overall access to resources and their control of the game map.

Spatial Co-ordination of Defensive Structures: Players will work together when building defensive structures, especially if their starting locations are close together, or their bases share an entrance. This prevents players from wasting resources by placing defences at a location that has already been fortified by their ally.

EXPERIMENTS

Effective teamwork improves a team's overall performance. Therefore, to determine the effectiveness of team behaviours, the single-player AI and team AI were compared. First, it was necessary to determine the effectiveness of the single-player AI, and which behaviours contribute most to winning an RTS game. To determine this, the single-player AI played a series of game against *StarCraft's* in-built AI. Then, the team AI played a series of 2vs2 games of *StarCraft* against the single-player AI. Additionally, individual team-based behaviours of the team AI were tested in isolation, to determine their individual contribution to the overall success of the AI.

Experiment 1: Effectiveness of the Single Player AI

The single-player AI's main behaviours are as follows: Base Expansion, Defending, Attacking, and Searching. Base Expansion refers to building additional smaller bases (expansion bases) to increase economic income. Searching refers to searching for and destroying remaining enemy buildings once all enemy bases have been destroyed. Additionally, the Defending behaviour was split into defending the main base, and defending expansion bases.

To determine the overall effectiveness of the single-player AI, it played twenty 1vs1 games against StarCraft's in-built AI (developed by Blizzard Entertainment). Following this each of the AI's individual behaviours were tested in isolation to determine their overall contribution to winning a game. We can measure the success of each individual behaviour by removing each of them from the AI one at a time. To perform each test a total of six test AIs were created, and are as follows:

- AI #1: This is the fully functional single-player AI.
- AI #2: This AI does not try to increase its resource income by building extra bases.
- AI #3: This AI does not defend its main base, but does defend its expansion bases.
- AI #4: This AI does not defend its expansion bases, but does defend its main base.
- AI #5: This AI does not attack enemy bases
- AI #6: This AI does not search for, and destroy, remaining enemy buildings after it has successfully destroyed all of the enemy's bases.

These test AIs were chosen based on the heuristic design of the AI. It was decided that twenty games provided a suitable sample size to accurately measure the AI's performance. At the end of each game the winner of the game and the time taken to win was recorded. The game time output by the StarCraft engine is in units of 'in-game frames'. All games were played on the 1vs1 StarCraft map called 'Astral Balance'. The results of these games are presented in Table 1.

Table 1: Results of Single-Player AI Testing

AI #	Description	Win%	Average Win Time	Average Lose Time
AI #1	Full Single Player AI	100	22013	N/A
AI #2	No Base Expansion	40	37697	48082
AI #3	No Defending Main Base	100	22278	N/A
AI #4	No Defending Expansion Bases	70	24463	60293
AI #5	No Attacking Enemy Bases	0	N/A	46732
AI #6	No Search & Destroy	60	22405	N/A

Experiment 2: Effectiveness of Team Work in RTS Games

To determine the effectiveness of the AI teamwork, the team AI was compared to the single-player AI. The AIs were compared by playing a series of 2vs2 team games. The first team consisted solely of single-player AIs, each performing their own individual strategy; the other team consisted only of team-based AIs, which decided on a single team strategy and cooperated to implement it together. To ensure that the individual strategies were well balanced and that neither team would gain an advantage based on their strategy choice (i.e. a team wins because of a superior strategy, rather than superior teamwork), the three strategies were compared. Two single-player AIs played a number of 1vs1 games against each other, with each using a different strategy. It was concluded from this testing that the strategies were well balanced and would not bias results.

Table 2: Possible Strategy Combinations of AI Teams.

Team Based AIs	
Player 1 Strategy	Player 2 Strategy
Aggressive + Aggressive	
Balanced + Balanced	
Defensive + Defensive	

Single-Player AIs	
Player 1 Strategy	Player 2 Strategy
Aggressive + Balanced	
Balanced + Defensive	
Defensive + Aggressive	

From these combinations we identify that there are 9 possible 2vs2 match ups that can be played. Of these nine match ups, three were chosen as test cases. These test cases are shown in Table 3.

Table 3: Test Cases used for Experiment 2

Test Case	Team base AIs	vs	Single-Player AIs
1	Aggressive + Aggressive	vs	Balanced + Defensive
2	Balanced + Balanced	vs	Aggressive + Defensive
3	Defensive + Defensive	vs	Aggressive + Balanced

These test cases picked as each of them represents a different scenario. In test case 1 the Team AIs are the aggressors, and are the first to attack. In test case 3 the Team AIs are defensive, and usually get attacked by the enemy team first. Test case 2 is somewhere in between. This allows us to analyse the effectiveness of teamwork under a variety of conditions. Also, to determine the effect of map size on test results, two different maps sizes were used.

Table 4: Results of Experiment 2

		Test Case 1	Test Case 2	Test Case 3
Small Map	Win % of Team AI	100 %	100 %	100 %
	Avg Win Time	294.7	315	460
	Avg Lose time	N/A	N/A	N/A
Medium Map	Win % of Team AI	100 %	100 %	100 %
	Avg Win Time	367	433	509
	Avg Lose Time	N/A	N/A	N/A

Experiment 3: Effectiveness of Individual Team Behaviours

Team behaviours were divided into 'Attacking' and 'Defending' behaviours. Two new variations of the Team AI were created, one which only coordinates attacks with its allies, and one which only coordinates defence; these AIs will be referred to as the 'Attack AI', and 'Defence AI' respectively. Each of these AIs was compared to the full team AI. Test cases were identified using the same criteria as Experiment 2 (i.e. situations in which: the full team AIs are more aggressive, the full team AIs are more defensive, and both teams are equally aggressive). Sixteen games were played for each test case, ten games on the small map and six on the medium sized map.

Experiment 3a: Team Attack Coordination

Experiment 3a examines the effect of attack coordination on overall team performance. The experiment parameters are the same as for experiment 2: 2vs2 team games are played with two *Attack AIs* on one team and two team AIs on the other. The test cases used are listed in Table 5.

Table 5: Test Cases used for Experiment 3a

Test Case	Full Team AIs	vs	Attack AIs
1	Balanced + Balanced	vs	Defensive + Defensive
2	Balanced + Balanced	vs	Balanced + Balanced
3	Defensive + Defensive	vs	Balanced + Balanced

Note that both teams are implementing a shared strategy, and the three test cases can be implemented using only two strategies.

Table 6: Results of Experiment 3a

		Test Case 1	Test Case 2	Test Case 3
Small Map	Win % of Team AI	100 %	80 %	60 %
	Avg Win Time	314	427	528
	Avg Lose time	N/A	455	468
Medium Map	Win % of Team AI	100 %	83 %	66 %
	Avg Win Time	437	469	522
	Avg Lose Time	N/A	704	736

Experiment 3b: Defence Coordination

Experiment 3b examines the effect of defence coordination on overall team performance. Similar to previous experiments, 2vs2 team games are played with two *Defence AIs* on one team and two team AIs on the other. The chosen test cases can be found in Table 7. The results of the experiment can be found in Table 8.

Table 7: Test Cases used for Experiment 3b

Test Case	Full Team AIs	vs	Defence AIs
1	Balanced + Balanced	vs	Balanced + Defensive
2	Balanced + Balanced	vs	Aggressive + Defensive
3	Defensive + Defensive	vs	Aggressive + Balanced

Table 8: Results of Experiement 3b

		Test Case 1	Test Case 2	Test Case 3
Small Map	Win % of Team AI	90 %	90 %	90%
	Avg Win Time	452	483	466
	Avg Lose time	584	514	392
Medium Map	Win % of Team AI	100 %	100 %	83%
	Avg Win Time	714	586	577
	Avg Lose Time	N/A	N/A	793

DISCUSSION

Experiment 1: From analysis of the results in Table 1 we can conclude the following about 1vs1 RTS games and the single-player AI: the single-player AI is a successful implementation of an RTS AI for playing 1vs1 games of StarCraft. Attacking enemy players is the most important factor in winning a game. Building expansion bases and increasing resource income is the second most important factor. Defending these expansion bases is the third most important factor. Finally, an aggressive counter-attacking approach to playing StarCraft is an effective replacement for defence.

Experiment 2: The team AIs won every game played. The difference in win time is caused by the AIs attacking at different points in time, depending on their chosen strategy. Note that the single-player AI and team AI are identical, except for the addition of team behaviours. Also, the team AI won regardless of the strategy it implemented and regardless of map size.

When the single-player AIs were more aggressive they attacked individually and at different times; each of these attacks was met with a coordinated joint defence by the team AIs. When the single-player AIs were more defensive, their lack of defensive cooperation meant that each of them was defeated in turn by a coordinated, joint attack by the team AIs. In test case 2, the aggressive single-player AI usually attacked first and was defeated, the team AIs then counter-attacked; the defensive single-player AI sat idle while its ally was being defeated. Eventually the defensive single-player AI faced both team AIs alone. In each of these situations the team AIs created favourable 2vs1 battles that allowed them to easily win each game.

Experiment 3a: The *Attack AIs* managed to win 19.33% of games. A difference in map size showed no significant changes in the results of each test case. The results from test case 2 show that defence coordination is an important part of teamwork, and contributed to the team AIs winning the majority of games played. The win percentage and game times from test case 1 are equivalent to those found in experiment 2; from this we can conclude that on its own, attack coordination is redundant if a team does not adopt an aggressive strategy. In test case 3 the team AIs lost 40% of their games. These results indicate that an attacking team This indicates that continuous coordinated attacks can eventually break through a coordinated joint defence; an aggressive team will eventually win if their enemies never counter-attack.

Experiment 3b: The overall win percentage of the *Defence AIs* is only 8.33%, but generally the *Defence AIs* took much longer to be defeated than the *Attack AIs* or the single-player AIs. Again, map size made little difference to the overall results.

In test case 1, when adopting a defensive strategy the *Defence AIs* performed far better than the *Attack AI,* an expected result. In test case 2 the results are similar to Experiment 2, however the time taken for the team AIs to win is significantly longer. In test case 3 the *Defence AIs* perform just as poorly as the single-player AI. We can conclude from this that while an aggressive strategy can give a team an advantage, uncoordinated aggression only leads to a quick defeat.

Neither the *Attack AIs* nor the *Defence AIs* came close to winning the majority of their games. From this we can conclude that the unison of both these aspects of coordination is a large contributing factor in winning team match ups in RTS games.

CONCLUSIONS AND FUTURE WORK

From this research we conclude that team-based behaviours and heuristics can be successfully identified and implemented in an RTS AI. A cooperative team-based AI was successfully created for the commercial RTS game *StarCraft* which is capable of high level cooperation with other players. From testing it was shown that the cooperative "Team AI" is far superior to the uncooperative "Single-Player" AI when playing team games, regardless of strategy or map size. Team coordination in both defence and attack allowed the cooperative team AIs to create favourable 2vs1 battle situations against their uncoordinated enemies. Also, because the core behaviours of the team AI and single-player AI used in testing were identical, we can conclude that the team behaviours were the only factor contributing to the team AI's success. Also, neither attack or defence coordination alone is enough to successfully win team games and the unison of both is large factor in implementation of successful teamwork.

The scope of this research involved high level cooperation between AI players in RTS games. This included: collaboration on a team strategy, management of base expansions, and overall military coordination. However, research into low level cooperation, such as tactical coordination of combat units amongst teammates, represents a large area for future work in the area of team-based RTS AI.

REFERENCES

Champandard, A. J. (2007, September 12). Top 10 Most Influential AI Games. Retrieved August 19, 2010, from AI Game Dev: http://aigamedev.com/open/highlights/top-ai-games/

Doherty, D., & O'Riordan, C. (2009). Effects of Shared Perception on the Evolution of Squad Behaviours. *IEEE Transactions on Computational Intelligence and AI in Games.*

McGee, K. (2010). Real-time team-mate AI in Games. *International Conference On The Foundations Of Digital Games.* Monterey, CA.

Monteiro, I. M., & Alvares, L. O. (2009). A Teamwork Infrastructure for Computer Games with Real-Time Requirements, *Agents for Games and Simulations*, vol. 5920 of Lecture Notes in Computer Science, chap. 4, pp. 48–62. Springer Berlin / Heidelberg, Berlin, Heidelberg.

Reynolds, J. (2004). Team Member AI in an FPS. *AI Game Programming Wisdom 2* , 207-216.

Stene, S. B. (2006). Artificial Intelligence Techniques in Real-Time Strategy Games - Architecture and Combat Behavior. Retreived From Norwegian University of Science and Technology Website: http://ntnu.diva-portal.org/smash/record.jsf?pid=diva2:121581.

The BWAPI Group. (2010, July 25). *Brood War API Manual.* Retrieved August 15, 2010, from BWAPI Project Website: http://code.google.com/p/bwapi/wiki/BWAPIManual

A SCALABLE APPROACH TO BELIEVABLE NON PLAYER CHARACTERS IN MODERN VIDEO GAMES

A. Rankin, G. Acton, and M. Katchabaw
Department of Computer Science
The University of Western Ontario
London, Ontario, Canada N6A 5B7
arankin@csd.uwo.ca, gacton@csd.uwo.ca, katchab@csd.uwo.ca

KEYWORDS

Believable decision making, non player characters, performance, scalability, artificial intelligence, video games

ABSTRACT

Recognizing the importance of artificial intelligence to modern video games, considerable efforts have been directed towards creating more believable non player characters in games, complete with personalities, emotions, relationships, and other psychosocial elements. To date, research in the field has primarily focused on the quality of character behaviour, and not on aspects of performance and scalability. This is increasingly a problem as we strive for more complex, dynamic, and realistic behaviour in an ever-growing population of characters in a virtual game world.

In this paper, we explore the performance and scalability of believable non player characters in modern video games. We propose an approach to scalability that is able to improve performance and increase the ability to support a larger quantity of characters without sacrificing the believability or quality of behaviour. We then discuss a prototype implementation of this approach, as well as experiences in experiments and simulations using this prototype. Results to date have been quite positive, and show tremendous potential for continued work in the future.

INTRODUCTION

Artificial intelligence is an increasingly important aspect of modern video games, and can be a determining factor in the overall success of a game (Cass 2002; Champandard 2004; Orkin 2004; Roberts and Isbel 2007; Sweetser 2008). One of the more active areas of research in game artificial intelligence is making more believable characters, also known as non player characters (Baille-de Byl 2004; Funge 2004;Guye-Vuilleme and Thalmann 2001; Lawson 2003; Predinger and Ishizuka 2001; Rizzo et al. 1997), as this can lead to games that are ultimately more immersive, engaging, and entertaining to the player (Dias and Paiva 2005; Livingstone 2006; Sweetser 2008). Doing so, however, requires developers to reach beyond traditional approaches to game artificial intelligence, including finite state machines, rule based systems, and static scripting (Bailey and Katchabaw 2009).

The requirements for character believability tend to be quite steep (Loyall 1997). They include elements such as personality, emotion, self-motivation, social relationships, consistency, the ability to change, and the ability to maintain an "illusion of life", through having goals, reacting and responding to external stimuli, and so on (Loyall 1997).

Computationally, these believable non player characters are orders of magnitude more complex than traditional approaches to character artificial intelligence, and so they introduce the potential for serious performance problems, especially when there are a great number of them inhabiting a game world (Bailey and Katchabaw 2009). This problem is only exacerbated by the computational needs of other game sub-systems, which together put an even larger strain on often limited and over-taxed system resources. Consequently, if we are to truly take advantage of believable characters in modern video games, we must take into consideration issues of performance and scaling with artificial intelligence as it has been done with other aspects of games, such as graphics (Rankin 2009).

To improve performance and scalability of games, techniques generally involve various different kinds of pre-processing of game data and various types of run-time optimizations to reduce the processing load as the game runs. From an artificial intelligence perspective, we can adopt both techniques as well, to optimize both the believable character models used prior to execution and their use in-game at run-time.

This paper introduces and examines a scalable approach to believable non player characters that utilizes several techniques to improve in-game performance. These include dynamic importance calculation, capability scaling or reduction, and pre-emptive, priority-based character scheduling and dispatching. Through utilizing these techniques, we can ensure that scarce computational resources are allocated to characters where they are most needed, and that characters are using the decision-making processes best suited to their current state and importance to the game, while still maintaining believability.

To take advantage of these performance and scalability optimizations, this paper also introduces a new approach to believable non player characters using utility based planning and action selection combined with dynamic, emergent behaviour. To support believability, various psychosocial elements are captured in this approach, including personality, emotions, relationships, roles, beliefs, desires, intentions, and coping. This extends our

earlier work in (Bailey and Katchabaw 2008; You and Katchabaw 2010), enabling richer and more compelling non player character behaviour.

The remainder of this paper is organized as follows. We begin by presenting and discussing related work in this area. We then describe our approach to performance and scalability for believable characters, outlining the various optimization techniques and strategies used in our own work. We then discuss our prototype system, including the implementation of both the non player characters and the various performance and scalability enhancements we employed. We then present and discuss our experiences from using this in a variety of experimental scenarios. Finally, we conclude this paper with a summary and a discussion of directions for future work.

RELATED WORK

To date, unfortunately, a literature survey reveals relatively little in performance and scalability specifically intended for affective artificial intelligence systems for games. That said, there is important and relevant research to note, as discussed in this section below.

Affective approaches designed for improved believability, such as the work in (Bates et al. 1994; Gratch and Marsella 2004; Imbert et al. 2005; Reilly and Bates 1992), unfortunately, make little mention of performance or scalability issues. Most frequently, this work focuses on the creation and simulation of a single character without considering the issues that arise in placing this character within a living game world, with numerous other inhabitants. While some work examines interacting characters, there are no measures of performance or scalability provided, and no mention of allocating computing resources to the characters (Rankin 2009).

Looking at other game artificial intelligence research, some attention has been given to issues of performance. The work in (Wright and Marshall 2000) is notable for providing a flexible general-purpose framework for artificial intelligence processing in games. While taking an "egocentric" approach, this work, unfortunately, provides no specific insight or performance optimizations for characters with psychosocial systems for believable behaviour, as in our current work. Work towards scalable crowd behaviour, such as (Pettré et al. 2006; Sung et al. 2004), is also relevant, but tends to focus on the believable simulation of groups, as opposed to the believable simulation of a large number of individuals, which can be very different problems.

We can also view believable non player characters as a form of agent, and games themselves to therefore be a kind of multi-agent system (Rankin 2009). Performance and scaling in multi-agent systems is still not a thoroughly explored area of research, however, but there are works of interest that borrow heavily from such diverse areas as operations research (Baker 1998; Haupt 1989; Holthaus and Rajendran 1997), distributed computing (Rana and Stout 2000), and operating system process scheduling (Ramamritham and Stankovic, 1994; Tanenbaum 2008). While not directly involving games, this work provides insight into the issues at hand, as well as potential solutions relevant to this problem domain.

While there is a lack of literature focused on performance and scalability in affective artificial intelligence for video games, there are tremendous opportunities for research in this area. Fortunately, insight and experience is available to be borrowed from other aspects of gaming research, as well as other disciplines, and can applied to this problem in a rather unique and innovative fashion, as we have done as described in this paper.

GENERAL APPROACH

As discussed earlier, we take advantage of several techniques in unison to improve performance and scalability of non player characters in video games. In this section, we present these various techniques.

Overview

We begin by recognizing that not every non player character in a game has equal importance to the player and the game at any point in time. Some characters are the focus of attention, are engaged in significant activities, or are otherwise critical to what is unfolding in the game. Others are of less importance, and their activities will largely, if not completely, go unnoticed to the player. As the player moves through the game world and plays the game, the importance of the various characters changes, with some becoming more important and others less so.

Furthermore, we note that characters that are not visible to the player, or otherwise not important to the player, do not need the same richness, detail, and fidelity in their behaviour as those that are the focus of attention, or are otherwise important, to maintain believability. In those cases, a game can safely do less with those characters, in many cases significantly so, with no perceptible difference to the player's experience. For example, suppose we have two non player characters in a game that are hungry; one, Alice, is in the same room as the player, while the other, Bob, is in a different locale far across the game world, outside of the player's ability to sense. To maintain believability, Alice must recognize her hunger, formulate a plan to satisfy her hunger considering her current psychosocial and physiological state and surroundings, and then execute this plan. After all, the player is in the same room and is able to see and hear everything she does; the slightest out of place action could sacrifice believability and adversely affect the player's experience. On the other hand, Bob would not need to do anything except simply continue to live to maintain believability. In the player's absence, it is likely reasonable to believe that he could provide himself with the necessities of life, without actually needing to do anything about it. In fact, the game would not even need to track Bob's hunger level to maintain the same level of believability. (That said, Bob would still need to progress in his life in the absence of the player, as it is likewise unreasonable to believe that he would do absolutely nothing at all when the player is not around. How this should be done to maintain believability is discussed below.)

With this in mind, we can safely scale or reduce the capabilities of a non player character according to their current visibility and importance to the player while still maintaining believability across all characters in the game, as shown in Figure 1. In doing so, we can achieve tremendous performance savings in characters with reduced capabilities as they require significantly less processing and this processing is far less complicated than characters operating at full capacity. This, in turn, allows the game to support a much larger number of characters without requiring additional resources to be dedicated to artificial intelligence processing.

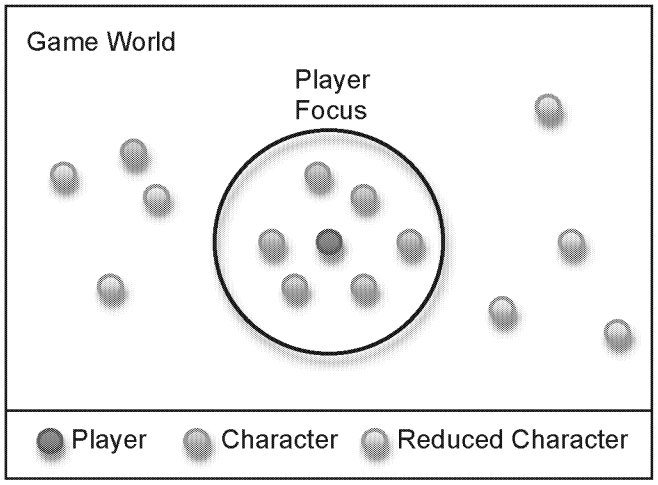


Figure 1: Characters in the Game World

To support this approach, each non player character in a game is a separately schedulable and executable entity, enabling computational resources to be allocated by a scheduling and dispatching subsystem on a character-by-character basis according to their importance. Furthermore, each character has access to a range of decision-making and processing algorithms, allowing their capabilities to be scaled or reduced based on their importance. Lastly, character importance is calculated and updated regularly based on a variety of factors to ensure that scheduling, dispatching, and capability adjustment are all carried out using the best available information. Each of these activities is discussed in further detail in the sections that follow below.

Scheduling and Dispatching of Characters

To allocate computational resources to non player characters in a game, a scheduling and dispatching subsystem is added to the game. The goal of this subsystem is rather simple—choosing the most appropriate characters to run at each game tick or frame of game execution.

Producing an optimal schedule for each tick is itself a computationally expensive task, and so we take a simpler, more heuristic approach using a system of priorities assigned to each non player character in the game. (Priorities are derived from the perceived importance of the characters, as discussed earlier in the paper, and below in further detail.) With this in mind, the most appropriate characters to execute in any tick are simply those with the highest priorities. Resource starvation is averted in characters of low importance through an aging mechanism built into the calculation of priorities.

Each game tick, the x characters with highest priorities are selected to run, where x is a positive integer configurable and tunable at run-time, based on a number of factors including the number of available processor cores, the workload being generated by other game subsystems, and so on. Each of the x selected characters is allocated one or more update cycles, depending on their relative importance. Each update cycle allows a character to execute for a period of time. This execution can be pre-empted, and does not necessarily allow the character to complete the task on which it was working. If the task is not completed when the character's update cycle ends, the character is paused and its priority is adjusted to reflect the work in progress.

Depending on the number of characters in the game, the priority system could be realized as either a single list sorted by priority, or a series of priority queues. While the mechanics of each approach is different, they can both deliver the same pre-emptive, priority-driven scheduling and dispatching of non player characters in a game.

Capability Scaling or Reduction

As discussed earlier, not every non player character in a game needs the highest levels of richness, detail, and fidelity in their behaviour in order to be perceived by the player as believable. Indeed, some characters could function believably with greatly reduced capabilities, depending on the state of the player, the game, and the various characters within it.

To better understand how capability scaling or reduction can be accomplished, we first outline how a believable non player character must function in the first place. We assume at least a basic set of psychosocial/cognitive processing elements, as without some form of these elements, it is extremely difficult for a character to deliver behaviour that could be truly considered believable (Acton 2009). This results in the high level character decision-making process shown in Figure 2. Each of the stages from this process is discussed below.

- **Event:** A notification of activity in the game world, whether it is from other characters, the world itself, or simply the passage of time.

- **Appraisal:** The event is examined to see if it is of interest to the character, and if so, does the event or its consequences have sufficient importance to the character to warrant further consideration. Deadlines for action/reaction may also be set, depending on the event.

- **Coping:** The character reflects on the event and updates its internal psychosocial and physiological state based on the event. This adjusts the character's mood, emotional memory, and so on.

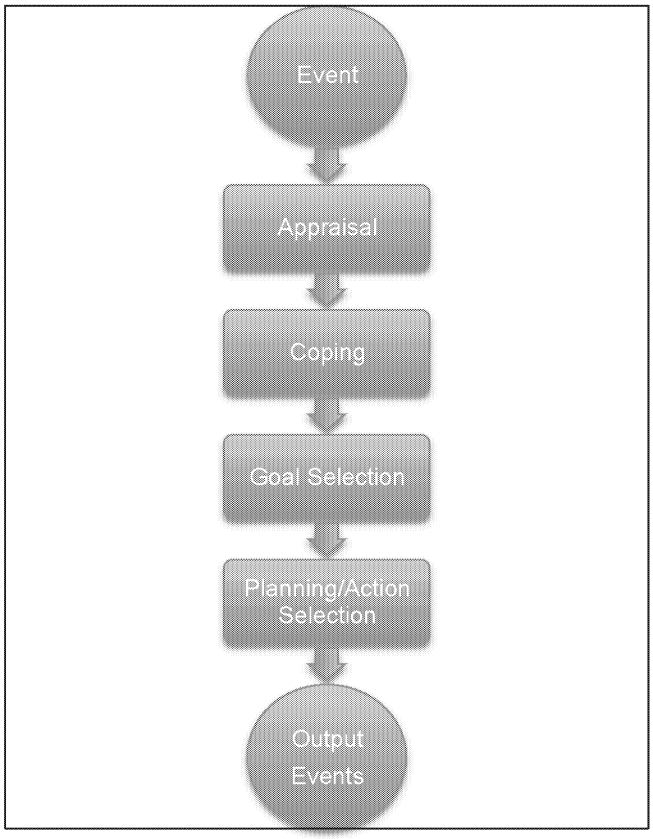


Figure 2: High Level Character Decision-Making Process

- **Goal Selection:** Given the current state of the character, its surroundings, and recent events, the character determines the goals that it should be trying to meet. Goals might be immediate, short-term, medium-term, and long-term.

- **Planning/Action Selection:** Considering the goals in place, a plan is created to achieve the goals with the desired results and acceptable side effects and consequences. If a plan already exists and is in process, it might be revised to reflect changes in goals, character state, and so on. With the plan in mind, appropriate actions are selected to have the plan implemented. Note that if events indicate a quick reaction is required, the current plan may be temporarily bypassed or abandoned to carry out alternate actions. (This may be the case, for example, to ensure self-preservation.)

- **Output Events:** Based on the actions selected, new events are generated and propagated into the game accordingly.

When this approach to character decision-making is considered, scaling a non player character can be accomplished in several ways. Generally, these methods can be grouped into either data reductions or processing reductions.

The purpose of data reductions is to limit the amount of information that is used in the various stages of the decision-making process. When done properly, this can greatly collapse decision spaces into smaller problems, making them far more efficient and less costly to work with. With care, these reductions can be carried out with little to no perceivable change in character believability. Data reductions can themselves take many forms.

One such form is a model or state reduction. As discussed earlier, in our approach, every character has a psychosocial model associated with them, defining their personalities, emotions, relationships, roles, values, and so on. A fully populated model could have a character's state defined by dozens of traits, all of which need to be maintained, and all of which could affect decisions made by the character. While rich, expressive, and useful during design to fully define a character, this can be very expensive computationally to use at run-time, as these factors must be consulted during appraisal, coping, goal selection, and planning/action selection. A careful reduction of this design model to a few key traits can produce a run-time model that is orders of magnitude more efficient, and far less costly to use within a game. This process is shown in Figure 4.

Another form of data reduction is event reduction. In this case, the number of events used to trigger decision-making or provide information during decision-making is limited, by reducing the types of events of interest, the importance of the various events or their consequences, or the frequency of their reporting to the character. This results in either fewer events reaching the character, fewer events moving past the appraisal stage (as the events are now deemed irrelevant, unimportant or inconsequential), or simpler decisions throughout the various stages of processing with fewer variables and factors to consider. All of these alternatives have the potential to improve performance substantially.

Processing reductions, on the other hand, generally involve altering a character's decision-making processes by changing algorithms or omitting aspects or complete stages of decision-making to improve performance. Again, if done with care, these performance improvements can be achieved without sacrificing believability. Possible processing reductions include the following.

- **Use of Defaults.** To accelerate the various stages of decision-making, default strategies can be used instead of analyzing and developing them dynamically on demand. For coping, events could have default impacts on character state, instead of determining this impact from the current state and other factors. For goal selection, characters could be assigned default goals to meet instead of developing them from scratch. For planning and action selection, default plans could be provided for each possible goal, complete with prescribed actions so that they do not need to be developed at run-time. While the defaults used will not do as well at reflecting the current state of characters or the game, the performance savings can be substantial even if this approach is used only for out of focus characters, or those that are unimportant.

- **Use of Randomization.** Much like through the use of defaults, randomization can be used to replace various aspects of behaviour, although doing so likely makes sense only with unimportant characters in the game. It

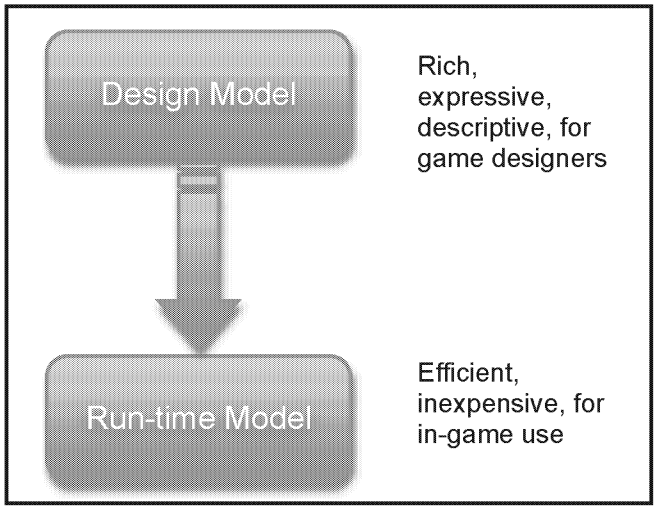


Figure 4: Character Model Reduction

is likely also wise to constrain randomness to ensure that characters are still acting believably. This can be accomplished in a variety of ways, such as constraining randomness to choose from a pre-defined set of defaults, or by permitting some level of initial processing develop sensible options that are chosen from randomly, instead of using a more expensive algorithm to make a more optimal choice.

- **Streamlining of Coping.** Integrating the impact of events into a character's internal state can be expensive, especially with a complex psychosocial model in use. To improve performance we can perform coping only in response to a limited set of critical events when characters are outside of the focus of the player. While this will result in characters whose moods and emotions change only in extreme circumstances, the player will be largely unaware of this.

- **Disabling of Goal Changing.** Whenever a character changes goals, any existing plan must be discarded and a new plan must be formulated, which can be quite expensive. To improve performance, characters can be prevented from modifying their goals while a plan is executing to avoid re-planning, unless very exceptional circumstances arise. While this will result in characters sticking with plans when they should likely be changed, this should not have too large an impact on their believability, provided that they are outside the focus of the player.

- **Automatic Achievement of Goals.** As mentioned above, planning and action selection can be computationally expensive tasks. When a character is outside the focus of the player, these tasks can be avoided entirely by simply allowing the character's goals to be achieved automatically. After all, if a goal is achievable by the character, and the player is not present to witness the goal actually being achieved, planning and action selection and execution are not required for the player to believe what has happened. It is important to ensure, however, that the goal is likely to be achieved by the character, and that the time required to meet the goal is properly taken into account; otherwise believability may be inadvertently sacrificed.

- **Disabling of All Processing.** If a character is relatively unimportant and is someone with whom the player has had no prior personal contact or knowledge thereof, it is possible to disable all, or nearly all, decision-making in the character. After all, the player would have little to no expectation of the character's mood, goals, or actions and so it is believable for the character to be in their initial state when first encountered by the player. The player has no way of knowing that the character was largely inactive up until when they entered the focus of the player.

  This strategy might also be applicable to important characters or characters that have been previously encountered, provided that they are out of the player's focus and will remain out of their focus until the next break in the game, such as a cut-scene, level transition, and so on. If important events are recorded, their effects on the character, as well as the character's goals, plans, and actions can all be simulated during the break, so that they are up-to-date when the player next encounters them.

When we combine the various forms of data and processing reductions together, we have great flexibility in the amount of capability scaling or reduction available to a game. If taken too far, this can eventually impact the believability of the game, but if done with care in an intelligent fashion, we can achieve tremendous performance savings with little to no effect on the believability perceived by the player.

Character Importance and Priority Calculation

The process of scheduling and dispatching, as well as the process of capability scaling or reduction both use a measure of a non player character's importance as a factor that ultimately affects both resource allocation and performance. Capability scaling or reduction uses a measure of importance directly, adjusting the capabilities of a character accordingly. Scheduling and dispatching use importance in the form of a priority in determining which characters are run in each game tick. Below, we examine how each measure is computed.

The importance of a non player character is determined by a collection of factors, with one calculating the importance, i, of a character as:

$$i = (\alpha f + \beta d + \gamma r + \delta c) / 4$$

where α, β, γ, and δ are weights between 0 and 1.0 to tune and balance the equation. The factor f is a measure of player focus on the character, which takes into consideration the distance between the player and the character, whether the character is within range of the player's senses, and the strength of relationships between the player and the character. The factor d is a designer-imposed measure of importance of the character, usually with respect to the story of the game. The factor r is a measure of importance defined by the currently active roles

of the character in question, as some roles in the game are inherently more important than others. Lastly, the factor *c* is a measure of importance that comes from character interactions. If a given character is involved with other, more important characters, their own importance might require a boost to put the characters on more equal footing. (For example, if the two are fighting each other, an inherently more important character could enjoy an unfair advantage over less important characters because the seemingly more important character has more capabilities and better access to computational resources.) The factors *f*, *d*, *r*, and *c* all range between 0 and 1.0 and so after scaling, the importance of a character also lies between 0 and 1.0.

With this in mind, the priority, *pri,* of a non player character can be computed as follows:

$$pri = \varepsilon i + \zeta s - \eta rc + \theta p$$

where ε, ζ, η, and θ are also weights between 0 and 1.0 to tune and balance the equation. (Carefully setting of these weights can also result in various scheduling policies, such as fair, least-slack-time, and so on (Rankin, 2009).) The factor *i* is the importance of the character as defined above. The factor *s* is a starvation factor that increases at a certain rate for each game tick that the character is not run; by doing this, even an unimportant character's priority will eventually exceed the most important character, allowing the unimportant character to run and avoid starvation. The factor *rc* is a run counter used to ensure that a single character is not over-allocated update cycles despite its importance. Lastly, the factor *p* is a progress measure that approaches 1.0 as the character approaches completion of its task at hand, to allow scheduling to clear out near-complete tasks from characters. All of factors *i*, *s*, *rc*, and *p* are normalized to between 0 and 1.0.

If desired, we can add a fifth factor to priority calculations to reflect the amount of capability reduction being applied to a particular non player character. Doing so may be reasonable since a character with its capabilities reduced by data or processing reductions will require fewer computational resources and therefore can cope with its schedule being reduced as well. Ordinarily, this would be accomplished using the importance factor *i,* as a low importance would trigger both capability and schedule reduction simultaneously. If importance and capability reduction were not so closely linked, a separate factor indicating reduction would then be necessary. (This can occur, for example, when there is a very large number of non player characters needing to be managed; in such a case, even the capabilities of fairly important characters would need reduction despite their importance in order to maintain game performance at an acceptable level.)

PROTOTYPE IMPLEMENTATION

As a proof of concept, we started with the development of a foundation framework for believable non player characters. This foundation was designed to be extended with modules for character scheduling and dispatching, as well as capability scaling or reduction, as discussed earlier in this paper. This prototype was developed for the various Microsoft Windows platforms using a combination of managed and unmanaged C++ using Microsoft Visual Studio 2008 as a development environment.

At the core of this prototype is a character system based on the high-level decision making process shown in Figure 2. Character state is composed of a psychosocial model integrating aspects of personality, emotions, relationships, roles, beliefs, desires, intentions, and coping. The personality model is derived from Reiss' theory of basic desires (Reiss 2004), as this approach presents personality in a fashion well suited to goal selection and consequences of actions. The emotion model selected is based on Ekman's universal emotion model (Ekman et al. 1972), a veritable standard in this area. Roles were developed using role theory from (Guye-Vuilleme and Thalmann 2001) as a basis. Aspects of appraisal and coping were adapted from (Gratch and Marsella 2004), while goal selection and planning/action selection were driven by standard utility based processes. A further discussion of the non player character system used as a foundation in this work can be found in (Acton 2009).

A scheduler and dispatcher module was added to the prototype to allocate computational resources to non player characters from the character system. This was based on a simple serial sort and search algorithm to determine the next characters to run based on priorities as described in the previous section. Capability scaling or reduction was implemented with multiple levels of reduction. A character running with full capabilities uses the complete character system described above. The first level of reduction uses rudimentary partial planning in which planning is carried out over several update cycles, with actions selected from partial plans in earlier update cycles while the current cycle continues to refine the plan. The second level of reduction uses full appraisal, coping, and goal selection capabilities, but then uses default plans and actions associated with goals selected, instead of carrying out a full planning/action selection stage. Finally, the third level of reduction uses randomization to select a goal and select a plan and actions capable of achieving this goal. While this is not the most realistic of approaches, it can still be appropriate for non critical characters in the game.

To assess the operation and performance of the prototype system, detailed logs are collected. These logs show all non player character state and activity at each tick of simulated game time, and contain performance information related to the scheduling and capability level of each character in the system. These logs are valuable to experimentation with the prototype system, as discussed at length in the next section of this paper.

RESULTS AND EXPERIENCES TO DATE

To assess the effectiveness of our approach to scalable believable non player characters, we conducted a series of experiments using our prototype system. In this section, we discuss highlights of our results. A complete presentation of experimental results and experiences can be found in (Rankin 2009).

Experimental Environment and Configuration

All experimentation was executed on an Intel Core 2 Duo system with a clock speed of 2.0Ghz and 4.0GB of RAM. The 64-bit variant of Windows Vista was used as the system's operating system. This configuration provided more than enough power for the experimentation we conducted.

The prototype system was configured to use the psychosocial model described in the previous section, with characters having access to 4 roles, 5 goals, and 8 actions during processing. While a typical game would have more possibilities open to its characters, this configuration on its own was sufficient to demonstrate the effectiveness of our approach. Time in the system was simulated so that 4 characters could run each game tick, there were 30 milliseconds between game ticks, and each action consumed one tick for execution. While actions would ordinarily take longer and have varied lengths in reality, this accelerated experiments and simplified analyses, as it was easier to confirm that factors such as importance were being properly handled by the system. Lastly, for simplicity and balancing, all weights used in calculating importance and priority were set to 1.0, except during starvation experiments. It is possible that better (or worse) results could be obtained through the fine-tuning of these weights. Additional experiments are currently under way, and others are planned to explore these and other issues more fully in the future.

Initial Experiments

Prior to more rigorous experimentation, initial testing was conducted to assess the basic operation of our prototype system. From this, we were able to verify:

- Equal fixed importance and priority resulted in an even distribution of resources to characters and equal opportunity for execution.
- Increased importance and priority translated into an increase in resource allocations to characters and a corresponding increase in execution time.
- Starvation of characters with low importance was effectively prevented by our approach to scheduling, and would be a serious issue if these measures were disabled or not provided in the first place.
- Characters with reduced capabilities required fewer resources to execute than characters with full capabilities intact.
- The prototype system could handle several characters of varying importance well, adjusting scheduling and capabilities accordingly without difficulty.

While these tests verified the correct operation of the prototype system, we still needed to assess the improved performance and scalability enabled by our approach. This is accomplished through experimentation outlined in the next section.

Stress Testing

To assess performance and scaling improvements, we used the prototype system to manage hundred of characters simultaneously. In these experiments, we executed three scenarios with 100, 200, and 800 characters respectively. Each scenario was itself run three times, once with all characters at full capability, once with all characters at the second level of reduction (as described in the previous section), and once with all characters at the third level of reduction.

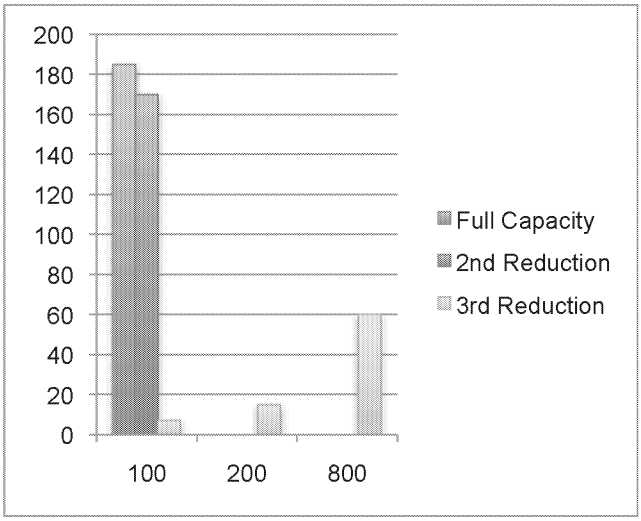


Figure 6: Character Stress Testing

Results from this experimentation are shown in Figure 6, measuring the time to completion of 200 game ticks in seconds. With 100 characters executing, performance under full capacity suffered greatly. There was a slight improvement under the second reduction, but performance was still unacceptable. (The small difference achieved with this reduction is because even under full capacity, the planner is somewhat primitive and incomplete. With a complete planner, full capacity characters would suffer worse, and there would be a bigger improvement achieved through this reduction.) With the third reduction, performance improvements were substantial. As the number of characters increased, only the third reduction characters were able to complete. While their time to completion increased, performance was still improved dramatically through this reduction.

It is important to note that while full capacity and slightly reduced characters had performance issues, the system would never be expected to support this many at a time. Through dynamic adjustments to capabilities, only a few would run at these capability levels at a time, depending on the game, with the others reduced further. This experiment was to solely demonstrate performance improvements through our approach.

From these results, we can see great improvements in the performance delivered by our approach to scalable believable non player characters. We are able to deliver a collection of characters that can adapt to various computational requirements through proper scheduling and capability adjustment.

CONCLUSIONS AND FUTURE WORK

This paper introduced a scalable approach to believable non player characters in modern video games. Through a combination of importance determination, prioritized scheduling and dispatching, and capability scaling or reduction, we can adjust the level of functioning of characters to adhere to computational constraints while maintaining believability. Experimental results with our prototype system have been both positive and quite promising.

In the future, there are many avenues for continued work. We plan to continue experimentation and further tune, scale, and explore the capabilities of our approach. We will continue the development of our prototype approach, adding both more functionality to our characters and additional capability reduction techniques. Lastly, we plan to embed our approach within a complete game or game engine to fully assess both its performance and its sustained believability through extensive user testing.

REFERENCES

Acton, G. 2009. *Playing the Role: Towards an Action Selection Architecture for Believable Behaviour in Non Player Characters and Interactive Agents*. Masters Thesis, Department of Computer Science, The University of Western Ontario.

Bailey, C. and Katchabaw, M. 2008. "An Emergent Framework For Realistic Psychosocial Behaviour In Non Player Characters". *Proceedings of FuturePlay 2008*. (Toronto, Canada, November 2008.)

Baille-de Byl, P. 2004. *Programming Believable Characters In Games*. Charles River Media.

Baker, A. 1998. "A Survey of Factory Control Algorithms that Can Be Implemented in a Multi-agent Heterarchy: Dispatching, Scheduling, and Pull". *Journal of Manufacturing Systems*, *Volume 17, Number 4*. Elsevier.

Cass, S. 2002. "Mind Games". *Appeared in IEEE Spectrum*, 39(12).

Bates J., Loyall A., and Reilly W. 1994. "An Architecture for Action, Emotion, and Social Behavior." *Lecture Notes in Computer Science, 830:55-68*.

Champandard, A. 2004. *AI Game Development: Synthetic Creatures with Learning and Reactive Behaviours*. New Riders.

Dias, J. and Paiva, A. 2005. "Feeling and Reasoning: A Computational Model for Emotional Characters". *Lecture Notes in Computer Science, 3808*.

Ekman, P., Friesen, W., and Ellsworth, P. 1972. Emotion in the Human Face: Guidelines for Research and an Integration of Findings. Pergamon.

Funge, J.D. 2004. *Artificial Intelligence For Computer Games*. A K Peters.

Gratch, J. and Marsella, S. 2004. "A Domain-Independent Framework for Modeling Emotion". *Cognitive Systems Research, 5(4).* (December 2004).

Guye-Vuilleme, A., and Thalmann, D. 2001. "A High-level Architecture For Believable Social Agents". *VR Journal, 5*.

Haupt, R. 1989. "A Survey of Priority Rule-based Scheduling". *OR Spectrum, 11(1).*

Holthaus, O. and Rajendran, C. 1997. "New Dispatching Rules for Scheduling in a Job Shop - An Experimental Study". *The International Journal of Advanced Manufacturing Technology, 13(2).*

Imbert, R., De Antonio, A., and De Informatica, F. 2005. "COGNITIVA: A Context Independent Cognitive Architecture for Agents Combining Rational and Emotional Behaviours". *In 5th. WSEAS Int. Conf. on Multimedia, Internet and Video Technologies.* (Corfu, Greece, 2005.)

Lawson, G. 2003. "Stop Relying On Cognitive Science In Game Design - Use Social Science". *Accessed June 2010 from http://www.gamasutra.com/php-bin/letter_display.php?letter_id=647.*

Livingstone, D. 2006. "Turing's Test And Believable AI In Games". *Computers in Entertainment (CIE), 4(1).*

Loyall, A. 1997. *Believable Agents: Building Interactive Personalities.* PhD Dissertation, Stanford University.

Orkin, J. 2004. "Symbolic Representation of Game World State: Toward Real-Time Planning in Games". *In Proceedings of the AAAI Workshop on Challenges in Game Artificial Intelligence.*

Pettré, J., de Heras Ciechomski, P., Maïm, J., Yersin, B., Laumond, J., and Thalmann, D. 2006. "Real-Time Navigating Crowds: Scalable Simulation and Rendering". *Computer Animation and Virtual World (CAVW) Journal - CASA 2006 Special Issue.*

Prendinger, H., and Ishizuka, M. 2001. "Social Role Awareness In Animated Agents". *Proceedings of the International Conference on Autonomous Agents.*

Ramamritham, K. and Stankovic, J. 1994. "Scheduling Algorithms and Operating Systems Support for Real-time Systems". *Proceedings of the IEEE, 82(1).*

Rana, O. and Stout, K. 2000. "What is Scalability in Multi-agent Systems?" *Proceedings of the Fourth International Conference on Autonomous Agents*. (Catalonia, Spain, June 2000.)

Rankin, A. 2009. *Scalability and Performance of Affective Multi-Agent Systems.* Masters Thesis, Department of Computer Science, The University of Western Ontario.

Reilly, W. and Bates, J. 1992. *Building Emotional Agents.* Technical Report CMU-CS-92-143, School of Computer Science, Carnegie Mellon University. (Pittsburgh, PA, May 1992.)

Reiss, S. 2004. "Multifaceted Nature of Intrinsic Motivation: The Theory of 16 Basic Desires". *Review of General Psychology*, 8(3).

Rizzo, P., Veloso, M., Miceli, M., and Cesta, A. 1997. "Personality-driven Social Behavior In Believable Agents". *Proceedings of the AAAI Fall Symposium on Socially Intelligent Agents*.

Roberts, D. and Isbell, C. 2007. "Desiderata For Managers Of Interactive Experiences: A Survey Of Recent Advances In Drama Management". *In the Proceedings of the First Workshop on Agent-Based Systems for Human Learning and Entertainment.*

Sung, M., Gleicher, M., and Chenney, S. 2004. "Scalable Behaviors for Crowd Simulation". *Computer Graphics Forum, Vol. 23, No. 3.*

Sweetser, P. 2008. *Emergence in Games*. Charles River Media, Game Development Series.

Tanenbaum, A. 2008. *Modern Operating Systems*, Third Edition. Prentice Hall.

Wright, I. and Marshall, J. 2000, "Egocentric AI Processing for Computer Entertainment: A Real-time Process Manager for Games". *Proceedings of the First International Conference on Intelligent Games and Simulation (GAME-ON 2000).* (London, United Kingdom, November 2000.)

You J. and Katchabaw, M. "A Flexible Multi-Model Approach to Psychosocial Integration in Non Player Characters in Modern Video Games". *Proceedings of FuturePlay 2010.* (Vancouver, Canada, May 2010.)

GAME SIMULATION AND GRAPHICS

BALLISTIC DAMAGE MODELS AND THEIR AFFECTS ON GAME PLAY

Tom Feltwell
University of Lincoln, UK
E-mail: tfeltwell@lincoln.ac.uk

KEYWORDS

Ballistics, Half-Life 2, damage model, engine modifications, level design, physics modifications

ABSTRACT

This paper presents a set of damage models based on ballistic simulation and a "post-flight" model of bullet energy. Three different damage models are simulated in Half-Life 2, and a modified ray-tracing method calculates the damage inflicted by a bullet. The first model uses a reciprocal time of flight function. The second implements a model of low viscous friction, whilst the third uses supersonic projectile drag. Our objective is to create a more believable and engaging player experience by more accurately simulating the behaviour of real weapon systems: our evaluation investigates the effect on user experience, and perceived weapon behaviour. This is conducted by testing players with different combinations of game levels and damage models. The results show that all models significantly disrupt the balance of gameplay, and the way in which players use different weapons.

INTRODUCTION

Accurate simulation of firearm weapons has been largely overlooked in the majority of modern games: instead, somewhat *ad hoc* damage calculations are used and balanced by hand. This has traditionally been driven by designers' need for configurable gameplay, and an easy-to-understand method of representation. The origins can be traced back to early games such as Wolfenstein 3D, where players had health rated between 0% and 100%. Weapons within such games inflict a relative percentage of damage, against the player's current health level. This is the simplistic principle behind damage models used in the vast majority of current first person shooter (FPS) games.

Recently, a small number of developers have attempted to implement more "realistic" models, such as those used in *Red Orchestra* (Tripwire Interactive, 2010) and the *Armed Assault* series (Bohemia Interactive, 2009). These games do use ballistic models, and are implemented more as simulations, making game play more technical and less accessible and engaging for casual players.

There is currently little "middle–ground" between simulation and the heuristic models still used in most FPS games. Our premise is that ballistic models can in fact be used to bridge the gap between simulation and game play by delivering a more interesting and engaging experience (than *ad hoc* models), whilst retaining enjoyable and accessible game play. To this end we implemented a set of three simplified ballistic models within the Half-life 2 FPS game, and investigate their effects on game play, and player experience. Our models use a small number of ballistic parameters that can be easily tuned to create a realistic, yet accessible output.

BALLISTIC MODELLING

Ballistics is a specialised branch of physics, the study of which has been broadly divided into three areas: "interior", "exterior" and "terminal" (McCoy, 1999). These respectively deal with the launch of the projectile inside the barrel, the atmospheric flight of the projectile and the impact of the projectile on target. In this project we examine aspects of exterior and terminal ballistic.

The main physical forces acting on a bullet in flight (exterior) are gravity and fluid dynamic drag. The dynamic behaviour of a bullet is further affected by spin imparted by the barrel rifling (during the interior phase) which helps to stabilise flight and maintain a higher flight velocity. The study of fluid dynamic drag is fragmented into different models applicable to different motion/media profiles, and we investigate different drag models in our implementation.

In the 19th century a set of standard projectile shape profiles (which can be scaled to the size/calibre of any projectile) were developed along with an associated model of fluid dynamic drag and "ballistic coefficient". There are 7 profile shapes representing commonly used projectile shapes (McCoy, 1999). Newtonian physics also provides a method for calculating drag with Stokes' equation (Bourg, 2002). This equation allows a general calculation of drag for any object, including projectiles. Of these two methods, the G models are used most commonly to perform practical ballistic calculations for firearms, as they provide robust estimates using relatively simple calculations. More complex Newtonian models are used by physicists and other researchers interested in developing more sophisticated models (Heard, 2008).

PROJECTILES IN HALF-LIFE 2

The Half-Life 2 game engine uses two different methods to process weapon projectiles. The first is used to model large projectile munitions, such as rockets, which are visible to the player and have a relatively long flight time. Such projectiles need to exhibit believable projectile behaviour over a flight time of several seconds, and so are processed using a simple Newtonian dynamic model, which incorporates the effect of gravity. The game engine stores a list of current objects of this type on a small stack, and each is updated at each game frame with a new position, orientation, and velocity.

This method is effective and appropriate for objects such as rockets, but is impractical for processing bullet projectiles: mainly because there are many more bullets being fired than rockets, so processing bullets in this way would impose a large computational overhead. This is not a problem however, as bullets are too small and fast to be seen, and so a simpler approach to processing is used. This second method uses a *hit-scan* test, to project a ray (called a *TraceLine)* from the muzzle of the weapon through the game world (Valve, 2009). Figure (1) demonstrates the TraceLine method along with a realistic bullet trajectory.

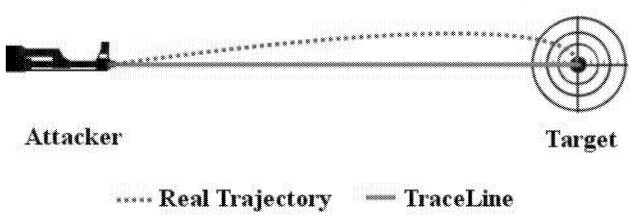


Figure 1: Comparison of TraceLine and Real Trajectories

The bullet is deemed to have hit the object first intersected by the TraceLine. There are no exterior effects on this hit-scan test as it is executed with one update of the game world. This means that the hit-scan method cannot easily be used in combination with ballistic affects such as viscosity and gravity. However, it does lend itself well to calculating the distance the bullet would have flown in the absence of these. Half-Life 2 already utilises the hit-scan test for other tasks and there is functionality already in place to allow the length of the ray to be extracted.

A standard method called *FireBullets* calls the hit-scan test for every weapon, drawing the information of the weapon from a data structure called *AmmoDef* (Valve Corporation, 2009). This structure stores the damage values for NPC and Player damage, along with physics impulse and tracer type. Every weapon in the game has an *AmmoDef* object defined, detailing its characteristics. The ray for the hit-scan is drawn from the attacker's direction and the first entity hit is stored in the *TraceLine* pointer. To apply damage to this entity, the *DispatchTraceAttack* method is called and given a variable called *flActualDamage*. The value of this variable is determined by checking whether the target entity is a player or an NPC, and selecting the correct damage value accordingly. This damage is applied instantaneously.

OUR BALLISTIC MODELS

As mentioned, there are a number of different ballistic models in use: we have selected three for implementation in this study. These require parameterisation, and we have thus gathered real specifications for the weapons used in Half-Life 2. These were identified as a Glock 17, .356 S&W Magnum, Spas-12 Shotgun, and H&K MP7 machine pistol (Jones and White, 2008). Corresponding specifications, such as muzzle velocity and bullet weight, are published by ammunition manufacturers. This presents a dilemma as there is significant variation between manufacturers. A European manufacturer was used as the source for the majority of parameters used in our implementation (Sellier & Bellot, 2010). The ammunition for the H&K MP7 is unique to the *RUAG Ammotec* aerospace defence company, and their product specifications were used in this case (RUAG Ammotec, 2010). Relevant specific data used for particular models is described in later sections.

The instantaneous nature of the hit-scan test implies that the projectile cannot easily be affected during flight without incurring significant computational overhead. The creators of Half-Life 2, Valve, distribute the source code for free, although the core engine code is not included: this is a limiting factor on modifying the hit-scan test function. Hence, our ballistic models are applied post-flight. The ray used in the hit-scan test has a method called *length()* which returns the distance travelled in game world units (1 unit = 19mm). Combining this value with the muzzle velocity v_0 of the projectile, the time of flight t_{of} can be calculated using Equation (1). Note that this is an estimate, as it does not account for decreased velocity during flight.

$$t_{of} = \text{length} / v_0 \quad (1)$$

The kinetic energy (KE) of the projectile at impact was also calculated, using the standard Equation (2), where m is the mass of the projectile and v_1 is the velocity at impact. This value was used to moderate the different damage models presented in this paper.

$$KE = \frac{1}{2} m v_1^2 \quad (2)$$

The damage models consider energy loss through air viscosity; however other potential sources of energy loss should also be considered (e.g. heat). To simulate additional loss, we decided that the final calculation of kinetic energy should be scaled to a coefficient of restitution. It should be noted that in all models, velocity is not represented as a vector, due to limitations in the game engine; therefore we only consider the speed component.

Model One

It is reasonable to conjecture that the damage caused by a bullet impact is dependent on the range to the target. Our first proposed model encodes this idea in a very simple way, such that the effect of fluid dynamic drag is uniform across all bullet types, and the impact velocity is inversely proportional to the time of flight. The equation for model one is defined by Equation (3).

$$v_1 = 1/t_{of} \quad (3)$$

The time of flight is inversely proportional to the distance flown, and so the model encodes the desired characteristic of decreasing damage affect as range increases.

Model Two

Model two employs a more sophisticated physics model, as it utilises the concept of fluid dynamic drag to calculate the energy of the bullet:

$$v(t) = v_0 \exp\left(\frac{-C_d}{m} t_{of}\right) \quad (4)$$

Equation (4) calculates the velocity of the projectile at time t, corresponding to the time of impact. v_0 is the muzzle velocity of the projectile, C_d is the drag coefficient of the projectile, m the mass of the projectile and t_{of} the time of flight. The drag coefficient was sourced from pre-calculated tables, which give values of 0.295 for a bullet, 0.500 for a lead ball (Douglas, 2005). Equation (4) is derived from the standard model of fluid dynamic drag in a low viscosity fluid, illustrated in Equation (5).

$$\underline{F}_{vl} = -C_d\underline{V} \quad (5)$$

Model Three

This final model uses a model of fluid dynamic drag presented by Kolbe (Kolbe, 2000), and applicable to projectiles travelling at supersonic velocity.

$$v(s)=v_0+\frac{(A-Bv_0)S}{Cv_0} \quad (6)$$

Equation (6) expresses the velocity of the bullet at specific range, s, relative to the initial velocity, v_0. The range of the bullet is related directly to the time of flight. C is the ballistic coefficient. Kolbe proposes the use of experimentally defined coefficients A and B, depending on the bullet's "ogive" and "tail" configuration. The ogive describes the nose shape of the bullet, which may be one of two categories: a tangent ogive – the intersection of the body of the bullet and the radius of the ogive blend together at a tangent point; and a sectant ogive - the intersection of the ogive and bullet body does not flow together smoothly and is thus a sectant of the arc (Lilja, 2002). Figure (2) illustrates these types of nose design, along with the two tail types. The configuration of each munition type was established through visual analysis, and Kolbe presents coefficients for all seven "G models". The final value required for Equation (6) is the ballistic coefficient, which is pre-calculated by the manufacturers. Note that the ballistic coefficient is the inverse to the drag coefficient: it defines how well the bullet overcomes air resistance.

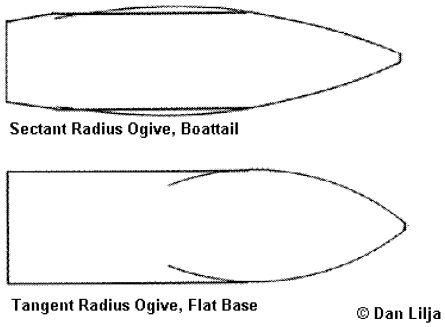


Figure 2: Sectant and Tangent Ogive Bullet Design (Lilja, 2002)

Some difficulty was encountered trying to define parameters for lead shot as the model presented by Kolbe is designed for G model bullets. In this instance coefficients A and B were selected from the most similar bullet shape, that of the *Rounded Nose, Flat Base* (Kolbe, 2000). The ballistic coefficient was sourced from The High Road shooting forum (The High Road, 2007).

IMPLEMENTATION

The initial step was to incorporate all the required ballistic information into the engine. The projectile weight and muzzle velocity were added to the *AmmoDef* of each weapon, with new methods added for declaring these variables. Coefficients specific to the damage model, such as the drag coefficient for model two, were coded directly into the *FireBullets* method. This handles the "firing" of weapons using the hit-scan test, so this is where the damage models were best implemented. The ammunition type is checked, the selected ballistic model run, damage calculated, assigned to target and statistics are recorded.

The first key point relating to the modifications is the checking of the current ammunition type. The *FireBullets* method is used by any entity firing a hit-scan based weapon, therefore the weapon/ammunition type needs to be extracted directly. Selection of ballistic model is controlled by a *ConVar* (Console Variable). These can be initialised and then modified during game play using the developer console. A new ConVar "drag_function" was added; its value indicating which damage model to execute. Zero was used to represent the standard damage model, and the new models are represented by a corresponding index.

Damage is calculated by the current damage model. Figure (3) shows the code implementation for calculating kinetic energy. The statement checks to ensure the projectile has not stopped, as v_1 represents the final velocity calculated by the damage model. If the projectile has no velocity ($v_1 = 0$) then there will be no damage applied.

```
if(v1 > 0){
   energy = (M /2) * pow(v1,2);
   energy *= 0.7;  //coeff. restitution
   flCalculatedDamage = energy/factor; }
else
   flCalculatedDamage = 0;
```

Figure 3: Calculation of Kinetic Energy

Quantitative and qualitative gameplay data was gathered for participating players. Statistics were logged to a text file which could be read and updated accordingly. Two statistics were recorded during testing: the number of bullets fired by the player, and the amount of damage received by the player. A method called *IsPlayer*, which returns true or false depending on what the hit-scan ray is intersecting, is used to verify each statistic log. In order to evaluate the proposed models, four short game levels were created for Half-Life 2. The settings and scenery were selected to be similar to the original game, and the levels were designed to engage the player in different modes of combat (long/short range, etc). Four testing conditions were used; a control condition (standard Half-Life 2 *ad hoc* model), damage model one, damage model two and damage model three. Each participant played a different level for each test condition, and participants were blocked in pairs for each combination of level and condition.

The players were timed and the number of deaths experienced per level was manually recorded. A short interview was carried out between each level asking perceived "difficulty rating", opinions of each weapon and game play tactics. All participants had previously played Half-Life 2, although level of ability varied. The results set totalled 11 players. Testing took place in a computer laboratory within the University of Lincoln. Data was obtained from each of the three damage models to represent how they affected the effectiveness of over range.

RESULTS

Figure (4) demonstrates the control (standard) damage model, along with the three newly implemented damage models for the pistol. It is interesting to note that models two and three have higher initial damage than the control, implying that the pistol is more powerful than was represented originally in the game, as these use ballistic information to ascertain damage. Model one presents a short steep curve, in keeping with the expectations for the reciprocal function. Model two provides a long shallow curve that would be expected for a projectile travelling through a viscous fluid. Finally, model three presents a shallow sloped line, with only a small variation between damage at 0 (zero) meter range and damage at 20 meter range. There are some variances across different weapons, but these are omitted for clarity.

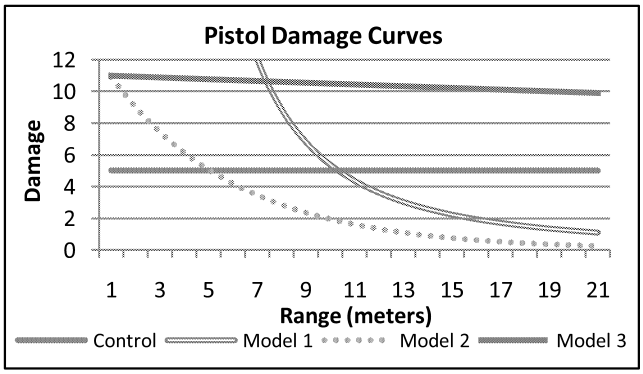


Figure 4: Damage Curves per Model for Pistol

Table 1: Average Level Completion Time per Condition

Condition:	Control	Model 1	Model 2	Model 3
Avg. Time (Seconds):	2.26	4.16	3.59	3.19

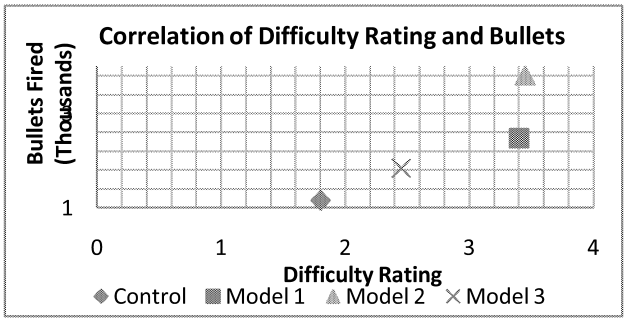


Figure 5: Correlation of Difficulty Rating and Total Bullets Fired

The average time to complete a level can be seen in Table (1). Model one took the longest time to complete; it can be reasoned that the higher difficulty, high number of bullets fired and large number of deaths shown in Figures (5) and (6) cause this. Model one has the steepest damage curve and from this we can gather that players were being damaged more at close range, and thus dying more often. Players found themselves changing tactics and the way they played the game: "*I stuck to...listening, waiting for them to rush past a corner*" and *"...due to the power [of the weapons] I just rushed through*". It was observed that once players died in this damage model, they attempted to damage opponents from long range. This was unsuccessful due to the steep damage curve and therefore meant an increased number of bullets were fired.

Model two was rated most difficult, and this can be attributed to the short shallow damage curve of this model. Players commented: "*the guns just didn't kill anything*", and in Figure (4) it can be seen that damage quickly drops to a small amount as range increases. It was observed that players were using the majority of the weapons as they quickly depleted their ammunition. Interestingly, the two models with similar damage curves, see Figure (4), were rated as being the most difficult. The cause for this is the fact each model has such drastically different damage when compared to the control. Model one produces a huge amount of damage at close range and tails off quickly at medium range, whereas model two produces a low amount of damage unless at extremely close range. Both of these situations provide equally difficult for the player, thus receiving a similar difficulty rating.

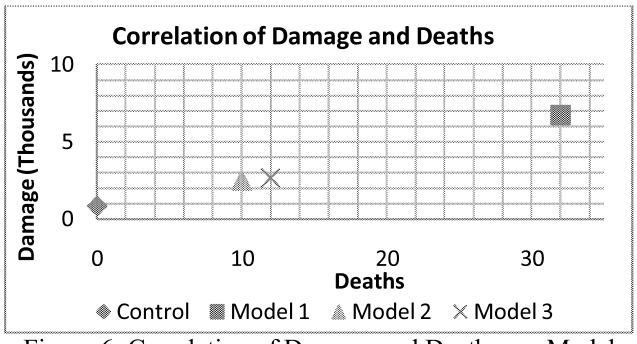


Figure 6: Correlation of Damage and Deaths per Model

Model three has the closest difficulty rating to the control test, and also the most similar completion time. Figure (6) shows that players died more often in model three than model two, but they rated it easier. Players noticed: "*Guns seemed a lot more powerful from a distance*" and another noted "*I was getting shot [from range] and it wasn't hurting that much*". The damage curves, see Figure (4), show that model three is the most similar to the control, with a gentle shallow slope. Players experienced more deaths due to the increased power (increased 6 points from control) but weapons responded in keeping with their expectations and thus, were more able to adapt to the changes

CONCLUSIONS

Each of the three damage models presented provides a distinctly different gameplay profile from the *ad hoc* damage model in Half-life 2. Ballistic information is required for the calculation of these models, although each model has its own limitations. Model one provided a difficult experience for players, although it may not be considered accessible due to the high number of deaths incurred. The damage curve of model two created weapons that were weak over range, making players rate it the most difficult. It is questionable whether this model would continue to be accessible and enjoyable to other players. Model three provides an improved challenge over the control condition whilst feedback from players is positive. It is harder than the control condition due to increased damage of weapons, but would require further testing to see how this affects the accessibility.

Calculating the drag on a projectile once it has flown creates middle ground between full ballistic modelling and *ad hoc* damage models in games. The damage is calculated using the presented models only once for each bullet, which is a drastic reduction in calculations compared to real-time ballistic modelling. Although there are a number of limitations, the approximated damage models present a good trade off between full ballistic modelling and *ad hoc* damage modelling.

When framing the results in terms of Mechanics, Dynamics and Aesthetics, model one changes the dynamics of the game, as players are forced to encounter enemies at close range, due to steep damage curve. Playing tactics changed as resultant game play was more skill based, meaning accessibility is reduced. The damage is exponentially higher than the control damage at close range and therefore it only takes one or two shots for the player to be killed. This puts a bigger emphasis on skills such as quick reactions, spatial awareness and caution, as players must use more skill to avoid being killed by the enemy. The effects on aesthetics are obvious as players rated it difficult and a large number of deaths were encountered. Model two changes the dynamics also as weapons are not powerful over range, creating problems with running out of ammunition and forcing players to use levels to their advantage. This is a different aesthetic than was originally intended, as it puts more emphasis on tactical skill, rather than "arcade fun". Model three provides the most similar dynamics to the original game, namely because the damage curve was very similar to the control damage model. Aesthically, although weapons are more powerful, this model is considered to have the most similarity with the original game, demonstrated through difficulty rating, damage curve and player feedback.

It can be debated whether a realistic damage model makes the game more fun and more challenging. The results show that the damage model that bears the closest resemblance to full realism (model three) is rated closest to the control condition, with regards to difficulty. The increased damage of the weapons can be seen to create more of a challenge, due to more deaths incurred, but whether this makes the playing experience more fun is subjective to the player. There are implications for level design when using any of these damage models. As previously mentioned, players changed their tactics when faced with high damage close up, and levels could be designed to take advantage of this fact incorporating non-linear elements to help a player strategise.

Evidently, damage models and weapon handling play a large part in how a player perceives the game environment. There are many other metrics that could be studied to give a better insight: studying AI vs AI combat, recording player movements through a level and working out mean attacker to target range would all yield interesting results. Increasing the sample size exponentially would also give better results that could be generalised better. This work also could be progressed into a kinetic energy-based damage system, whereby all enemies are damaged realistically completely replacing the *ad hoc* damage system.

REFERENCES

Bohemia Interactive, 2009. *ArmA: Manual.*Online: http://community.bistudio.com/wiki/ArmA:_Manual

Bourg, D. M. 2002. *Physics for Games Developers.* O'Reilly Media Incorporated, California, 101-120, 165-166.

Douglas, J. F.; J. M. Gasiorek; J. A. Swaffield; and L. B. Jack. 2005. *Fluid mechanics*. Pearson Education, Harlow, Essex, 943.

Heard, B. J. 2008. *Handbook of Firearms and Ballistics*. John Wiley & Sons Limited, Oxford. 1-143.

Higher Education Statistic Agency. 2008. *Table 0a – All students by institution, mode of study, level of study, gender and domicile, 2006/07.* Online: http://www.hesa.ac.uk/dox/dataTables/studentsAndQualifiers/download/institution0607.xls

Jones, R. and A. White. 2008. *Jane's Guns Recognition Guide.* Harper Collins, London, 15-350.

Kolbe, G. 2000. *A Ballistics Handbook*. Pisces Press, Newcastleton, 302-308.

Lilja, D. 2002. *Calculating Bullet Weights*. Online: http://www.riflebarrels.com/articles/bullets_ballastics/bullet_weights.htm

McCoy, R. L. 1999. *Modern Exterior Ballistics: The Launch and Flight Dynamics of Symmetric Projectiles*. Schiffer Publishing Limited, Pennsylvania, 10-17.

RUAG Ammotec. 2010. *4.6 mm x 30 SINTOX® Steel Monolith (DM31) Product Details*. Online: http://www.ruag.com/de/Ammotec/Armee_und_Behoerden/_x0034_.6mm/Factsheets_PDF/4.6x30_Penetrator_MK_II.pdf

Sellier & Bellot. 2010. *Pistol and Revolver Cartridges*. Online: http://www.sellier-bellot.cz/ammunition-map.php

The High Road. 2007. *Shot velocity loss.* Online: http://www.thehighroad.org/archive/index.php/t-300545.html

Tripwire Interactive, 2010. *Red Orchestra Ballistics*. Online: http://forums.tripwireinteractive.com/showthread.php?p=561451

Valve Corporation. 2009. *TraceLine – Valve Developer Community.* Online: http://developer.valvesoftware.com/wiki/Using_TraceLines

Valve Corporation. 2009. *FireBullets() – Valve Developer Community. Online: http://developer.valvesoftware.com/wiki/FireBullets%28%29*

Towards An Exaggeration Machine

Ken Newman PhD
Academy of Digital Entertainment
University of Applied Sciences Breda (NHTV)
Breda
Netherlands
E-mail: newman.k@nhtv.nl

KEYWORDS
Exaggeration, animation principles, procedural animation.

ABSTRACT

In game animation and indeed in all forms of animation the exaggeration of movement is a necessary and time consuming activity of the professional animator. The question of how traditional animation principles can be synthesized using procedural algorithms is complex. A traditional animator makes many decisions intuitively about timing, shape and style to achieve their desired result. This paper considers exaggeration in terms of these three dimensions – timing, shape and style, reviewing the relevant research in each and attempting to pull together a broad range of parameters under these three dimensions which would be needed to effectively synthesize movement exaggeration informed by traditional animation principles.

IMPROVING ON REALITY

Frank Thomas veteran animator from the golden age of Disney animation is quoted as saying *our characters obey all the laws of physics, except when it's funnier not to*. John Lasseter's seminal paper (Lasseter, 1987) articulates not only the relevance of traditional animation principles to the emerging computer animation processes, but also the problem of the animator's time and skill level required to creatively exaggerate movement. Another factor that has further eroded the adherence to traditional animation principles since Lasseter's original paper is the extensive use of motion capture (mo-cap) as the starting point for character animation especially in game animation.
Animation producers who use mocap as the starting point can easily slip into a habit of assuming without questioning that the most appropriate and believable movement for their brief is a cleaned-up hyper-realistic version of the natural motion-captured movement. There are several problems with this assumption. The movements of a full-size human figure, when reduced down to a few centimeters on a screen, do not preserve the subtle properties of weight force and strength. They need to be exaggerated just to appear normal. Other problems are to do with expectation and human perception. Just as martial arts films have developed exaggerated expectations that combat moves will be sped up and whooshing sound fx added to every punch - so raw mo-cap movement can be very unsatisfying to a contemporary viewer. In a game animation environment the lessons of exaggeration from the old masters are often forgotten or ignored, and in 2010 (Citters, 2010) considers his appeal for a return to animation principles rather than slavish adherence to realism to be a renegade approach.

While there are a number of fruitful areas of research around the topic of procedural exaggeration of movement, the work tends to focus on a single part of the problem in precise mathematical isolation. This paper is intended to act as a positioning paper to both review existing work and also propose an integration of the different work and future work into some kind of useable "exaggeration machine".

WHAT TO EXAGGERATE

The advocates of traditional animation principles are often not the people who are pushing the boundaries of procedural animation research, and to be fair, some of the animation principles really do not lend themselves to procedural solutions. *Character Appeal*, principle number 11, (Lasseter, 1987), for example is a principle so bound up in human ideas of beauty, culture, time and place, that is difficult to imagine procedures that could ever hope to implement this principle. So which of the traditional animation principles are appropriate for procedural solutions? I suggest three dimensions to procedural exaggeration – Timing, Shape and Style – which together encompass the traditional animation principles which do lend themselves to procedural solutions. Timing incorporates timing, anticipation and easing. Shape incorporates squash and stretch, and Style incorporates visual staging, animation icons and metaphors.

These three dimensions will be discussed with a brief overview of the current state of research in each of the dimensions at the time of writing.

TIMING

Previous work has been done in identifying the features of a movement to be used for exaggerating intensity or effort by comparing human subjects doing the same action with different levels of intensity (Davis & Kannappan, 2002). In this study the live subject is used to help identify which

parts of a movement are appropriate for procedural exaggeration.

In a 2008 study by the author of how live actors perform exaggeration, 10 subjects were asked to perform three levels of exaggeration – normal, medium and extreme - of three discrete actions: a tennis serve, a sneeze, and a shock reaction. Each performance should be not longer than 10 seconds. For the purposes of this study a discrete action was a single directional primary movement which has an identifiable beginning, trajectory and end. Each action began and ended with a neutral stance. The sequences were videoed and the timing of each action was broken down into anticipation, the action trajectory and the ease-out/return to neutral.

Figure 1 shows the average timing distribution for all subjects and actions broken into the three parts (represented by the light, dark, light bars) of each action – anticipation, action trajectory and ease out/return.

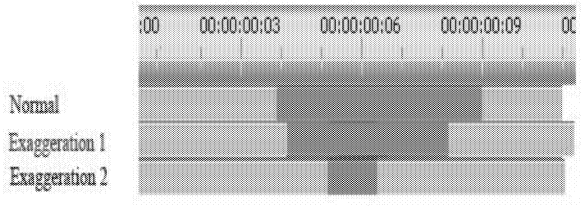


Figure 1: Timing Averages for Normal, Exaggeration 1(Medium), and Exaggeration 2 (Extreme)

In all cases the subjects intuitively increased the anticipation and ease out time while reducing the action time. This is consistent with what we see in traditional cartoon animation and therefore suggests that in a discrete action, exaggeration means reducing the duration of the action trajectory relative to the anticipation and ease out durations. In a traditional animation we might see extreme examples of exaggerated timing when, for example, a character runs off-stage by simply striking an anticipation pose and then vanishing in a couple of frames of directional motion blur, an appropriate sound fx and a small cloud of dust.

To some extent timing of discrete actions can be controlled procedurally with easing formulas and work has been done on essential easing formulas(Parent, 2007), motion-blending algorithms (Stocker, 2006),

SHAPE

From the earliest days of photography and cinematography there was an awareness of the tendency of shapes to deform according to the mass the shape represents, the rigidity of that mass and the physical forces acting on the mass (weight, acceleration).

From Muybridges famous photographic studies of horses running it was possible to study the extent of the shape deformation in nature. The Disney animators articulated shape deformation in the principles of stretch and squash. In moments of extreme exaggeration for example living figures might begin to behave *like a sock filled with custard* and deformed appropriately.

Work has been done on procedural methods for shape deformation based on traditional squash and stretch (Chenney, Pingel, Iverson, & Szymanski, 2002) and more recently using *time-shifted deformations* on each joint of a rigged character (Kwon & Lee, 2009). In these studies formulas are offered for controlling shape deformation based on trajectory and velocity. Chenney et al also offer formulas for collisions and coming to rest.

There is also work in the deformation of 2D shapes which may have some relevance to movement exaggeration (Weber & Gotsman, 2010), and (Karni, Freedman, & Gotsman, 2009).

STYLE

The style dimension is intended to cover animation effects which arise more from human perception and existing animation conventions than from physical laws. Fairly standard examples of stylistic exaggeration in traditional animation include:

1. dust clouds to indicate acceleration or decceleration.
2. motion blur lines to indicate extreme movement.
3. Splat shapes to indicate collision.
4. Arrow quivering behaviour on collision
5. Yo-yo behaviour on coming to rest.

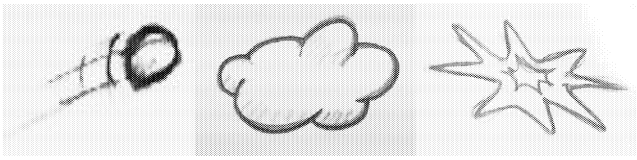


Figure 2: Some Examples of Stylistic Exaggeration

A key factor in the style dimension is in the introduction of sound fx. Traditional animators know intuitively that screeching tyres, arrow quivering, and impact slaps will always make movement seem more exaggerated and this is supported by studies on the influence of sound on animation smoothness (Mastoropoulou, Debattista, Chalmers, & Troscianko, 2005).

Aside from the human perception studies there has not been much work in developing stylistic exaggeration algorithms. The closest are perhaps studies in movement styles in facial, cloth, (Hughes, et al., 2007) and hair movement (Chang, Jin, & Yu, 2002).
One approach to the problem of hair movement is to use a database of hair movement in wind, head turns, acceleration etc (Sugisaki, Kazama, Morishima, Tanaka, & Sato, 2006). Although this is not a totally procedural approach the possibility exists for isolating parametric properties of the movement data and drawing on these to

create at least a partially procedural solution. This approach may also yield results if applied to a broader range of style effects.

SUMMARY

This paper, like others before, begins with the assumption that there are real advantages in using traditional animation principles in contempary computer game animation. In practice game animation often begins with mo-cap and the time and cost involved in implementing exaggeration according to animation principles can be prohibitive. Procedural exaggeration may provide the solution to this problem, and in many areas extensive work has already been done, though mostly the studies have been necessarily narrow focused on a particular problem. My proposal is to model exaggeration processes in three very broad dimensions Timing, Shape and Style. These three dimensions appear to me not simply logical groupings of the relevant animation principles which stand a chance of being parameterized for procedural synthesis, but also reflects a more complete picture of the ways traditional animators are intuitively introducing exaggeration into their work.

The review of current studies in each of these dimensions locates some fruitful studies in the Timing and Space but less serious research activity in the Style dimension. This appears to present an opportunity future research in at least two areas – firstly some serious studies in procedural generation of stylistic exaggeration and secondly in the area of unifying the existing work in all three dimensions into a single exaggeration machine.

REFERENCES

Chang, J. T., Jin, J., & Yu, Y. (2002). *A practical model for hair mutual interactions*. Paper presented at the Proceedings of the 2002 ACM SIGGRAPH/Eurographics symposium on Computer animation.

Chenney, S., Pingel, M., Iverson, R., & Szymanski, M. (2002). *Simulating cartoon style animation*. Paper presented at the Proceedings of the 2nd international symposium on Non-photorealistic animation and rendering.

Citters, D. V. (2010). Animation and technology: a renegade approach. *SIGGRAPH Comput. Graph., 44*(3), 1-4.

Davis, J. W., & Kannappan, V. S. (2002). *Expressive features for movement exaggeration*. Paper presented at the ACM SIGGRAPH 2002 conference abstracts and applications.

Hughes, C. J., Grzeszczuk, R., Sifakis, E., Kim, D., Kumar, S., Selle, A. P., et al. (2007). *Physical simulation for animation and visual effects: parallelization and characterization for chip multiprocessors*. Paper presented at the Proceedings of the 34th annual international symposium on Computer architecture.

Karni, Z., Freedman, D., & Gotsman, C. (2009). *Energy-based image deformation*. Paper presented at the Proceedings of the Symposium on Geometry Processing.

Kwon, J.-Y., & Lee, I.-K. (2009). *The squash-and-stretch filter for character animation*. Paper presented at the ACM SIGGRAPH ASIA 2009 Posters.

Lasseter, J. (1987). *Principles of traditional animation applied to 3D computer animation*. Paper presented at the Proceedings of the 14th annual conference on Computer graphics and interactive techniques.

Mastoropoulou, G., Debattista, K., Chalmers, A., & Troscianko, T. (2005). *The influence of sound effects on the perceived smoothness of rendered animations*. Paper presented at the Proceedings of the 2nd symposium on Applied perception in graphics and visualization.

Parent, R. (2007). *Computer Animation, Second Edition: Algorithms and Techniques*: Morgan Kaufmann Publishers Inc.

Stocker, H. (2006). *Linear filters: animating objects in a flexible and pleasing way*. Paper presented at the Proceedings of the eleventh international conference on 3D web technology.

Sugisaki, E., Kazama, Y., Morishima, S., Tanaka, N., & Sato, A. (2006). *Anime hair motion design from animation database*. Paper presented at the Proceedings of the 2006 international conference on Game research and development.

Weber, O., & Gotsman, C. (2010). *Controllable conformal maps for shape deformation and interpolation*. Paper presented at the ACM SIGGRAPH 2010 papers.

GAME DESIGN

COLORS AND EMOTIONS IN VIDEOGAMES

Evi Joosten, Giel van Lankveld, and Pieter Spronck
Tilburg University / TiCC
P.O. Box 90153, NL-5000 LE Tilburg, The Netherlands
E-mail: evijoosten@gmail.com, g.lankveld@uvt.nl, p.spronck@uvt.nl

KEYWORDS
Emotions, videogames, Self-Assessment Manikin.

ABSTRACT

People experience emotions when playing videogames, and these emotions are a main reason for playing. In our research, we examine whether colors can be used in videogames to elicit specific emotions. In an experiment we used a videogame in which four different colors, associated with four specific emotions, were used in four different conditions. After each condition we measured the players' emotional responses by means of a Self-Assessment Manikin questionnaire. We found that the color red evoked a highly-aroused, negative emotional response, while the color yellow evoked a positive emotional response. These results were significantly different from the emotional responses measured for other colors. Furthermore, we found that inexperienced players showed much more explicit reactions to colors than experienced players. We conclude that the use of colors is a suitable method for game designers to elicit specific emotional responses from the players.

INTRODUCTION

"Emotions" are feelings, caused by persons or events, that are experienced over a short period of time and that can change rapidly (Robbins and Judge 2008). Experiencing emotions tends to be the main reason for people to play videogames (Ravaja et al 2004). Therefore, manipulating emotions is a game designer's prime concern, and playing a game becomes an enjoyable experience if players experience emotions that they find satisfying.

Psychological research has shown that music and color affect people's emotions. There is evidence that this holds in particular for videogames (Livingstone and Brown 2005). Stark et al (1982) found that colors influence emotions experienced during play of a gambling game. Wolfson and Case (2000) found a link between the colors red and blue, and emotions experienced while playing a "breakout"-like game. Previous research on the effect of colors on emotions in videogames was, however, generally limited in three ways: (1) only few colors were examined, (2) rather simple games were used, and (3) the adaptation of colors in order to influence emotions was considered to be outside the scope of the research.

In the present research we examine to what extent colors can influence a player's emotions in a relatively complex role-playing game. We do not measure emotions directly, but instead focus on emotional responses, i.e., a player's arousal and enjoyment experienced during game playing.

We first provide background information on research into emotions, emotional responses, emotions in video games, and emotions and colors. We then outline our experimental setup and describe and discuss results.

BACKGROUND

This section discusses how emotions can be measured, what the most salient emotions in video games are, and to what extent emotions can be evoked by the use of colors.

Measuring emotions

Theories of emotions state that humans are evolutionary endowed with a limited set of basic emotions, namely anger, fear, disgust, happiness, sadness, and surprise (Ekman 1993). Each basic emotion is independent of the others in its behavioral, psychological, and physiological manifestations, and each arises from activation within unique neural pathways of the central nervous system (Posner et al 2005). One should therefore be able to measure basic emotions by examining facial expressions and physiological responses. At present, however, insufficient empirical foundation exists for defining which emotions are basic and how they correlate to what can be observed (Ortony and Turner 1990).

Emotions can be measured indirectly in terms of emotional responses. Studies have repeatedly yielded two-dimensional models of emotional responses (Larsen and Diener 1992). In these models all affective states (emotions) are understood to arise from common, overlapping, neuropsychological systems (Posner et al 2005). One often-used model is the 'circumplex model of affect' (Russell 1980). It characterizes emotions in terms of two dimensions of emotional responses, namely (1) arousal and (2) valence. *Arousal* is the physiological and psychological state of being proactive (activation) or reactive (deactivation) to stimuli. *Valence* is an intrinsic positive (pleasant) or negative (unpleasant) feeling that is evoked by an event, object, or situation. Russell posits that each affective state is the consequence of a linear combination of these two independent dimensions (Figure 1). The present research measures emotions based on emotional responses (arousal and valence), derived from the circumplex model of affect.

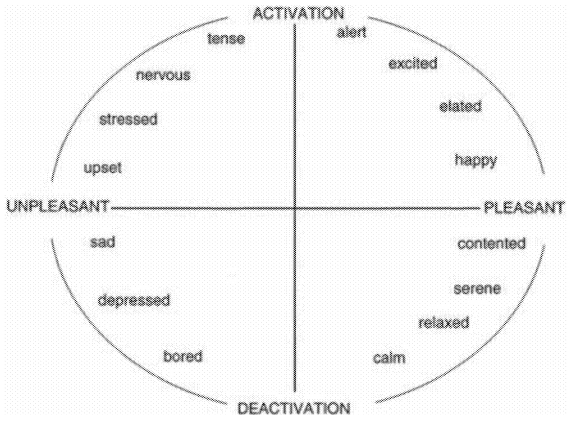


Figure 1: The circumplex model of affect (Russell 1980) with valence on the horizontal and arousal on the vertical axis.

Salient emotions in videogames

Ravaja et al (2004) examined whether there are reliable differences in the emotional response patterns elicited by four videogames with diverse characteristics, namely: *Tetris*, *Super Monkey Ball 2*, *Monkey Bowling 2*, and *James Bond 007: NightFire*. Emotions elicited by the games were defined by arousal and valence in terms of six affective states, namely: fear, anger, relaxation, pleasure, joy, and depression. They showed that arousal and valence are two important factors in enhancing emotions, especially in *James Bond 007: NightFire*. They demonstrated that arousal and valence are suitable measurements to identify emotions in a videogame.

Perron (2005) attempted to characterize some prototypical videogame emotions. Based on film theory he identified the following seven emotions: interest, enjoyment, worry, fear, surprise, anger, and frustration. All these emotions score high on the arousal level. Therefore, there is evidence that videogames generate highly-aroused emotions.

Effect of color on emotions

Color is a high predictor for emotions (D'Andrade 1974). El-Nasr et al (2006) measured the impact of changing color (in terms of saturation, brightness, and warmth) and contrast on tension felt in videogames. They reported a significant effect. Their results suggest that the use of lighting patterns attaches players emotionally to a game.

Wolfson and Case (2000) examined the effects of the colors red and blue on players' emotions. They used five simple videogames where the color of the screen was manipulated. The results suggested that participants in the red group were more aroused than participants in the blue group. Therefore they found effects on arousal by manipulating color in videogames, though limited to the colors red and blue. Furthermore, the simple games they used can be considered less arousing compared to the more complex videogames that are common today.

Plutchik (2001), Oberascher and Gallmetzer (2003), and Valdez (1994) stated that every basic emotion can be linked to a color. Plutchik (2001) linked colors to basic emotions as shown in Table 1. Oberascher and Gallmetzer (2003) confirmed Plutchik's findings. Of the emotions used in Plutchik's research, the highly-aroused ones are surprise, fear, joy, and anger. These should be prevalent in videogames. Therefore our research focuses on their associated colors: light blue, dark green, yellow, and red.

Emotion	Color
Surprise	*Light blue*
Fear	*Dark green*
Acceptance	Light green
Joy	*Yellow*
Anticipation	Orange
Anger	*Red*
Disgust	Purple
Sadness	Dark blue

Table 1: Emotions and corresponding colors as identified by Plutchik (2001). The colors and emotions in italics are those used in the present research.

EXPERIMENTAL SETUP

For the purpose of this research, we designed and built a videogame. In this game background colors were manipulated. The emotional responses of players were measured through a questionnaire, and analyzed. This section describes the participants, the questionnaire, the game, and the analysis method, respectively.

Participants

Students of the School of Humanities of Tilburg University were approached to participate in our experiment. A total of 68 students signed up, 60 of which were used during the analysis (the remaining 8 students either did not show up, or failed to complete the game). 25 of the participants were male, 34 female, and one did not specify gender. Their age ranged from 18 to 31 years, with a mean age of 21.5 years.

Questionnaire

To measure players' emotional responses a picture-oriented instrument called Self-Assessment Manikin (SAM) was used (Lang 1980). The SAM, displayed in Figure 2, has been used to effectively measure emotional responses in a variety of situations, including reactions to videos (which can be compared to videogames). It is an easy, nonverbal (and therefore language-independent) method for quickly assessing people's reports of emotional responses (Bradley and Lang 1994). By asking participants to indicate which image from a row of pictures best approaches their current emotional state, the SAM directly assesses the valence, arousal, and dominance associated with a response to an object or event. The arousal and valence dimensions are as discussed in the Background section. The dominance dimension is not related to an emotional response, but is used for indexing the relationship of control that exists between the perceiver and the perceived situation. Scores ranged from 1 to 9 (participants could also indicate a value between two neighboring pictures). For the first row, 9 indicated "most aroused" (left-most picture). For the second row, 9 indicated "most pleasant" (left-most picture). For the third row, 9 indicated "most in control" (right-most picture).

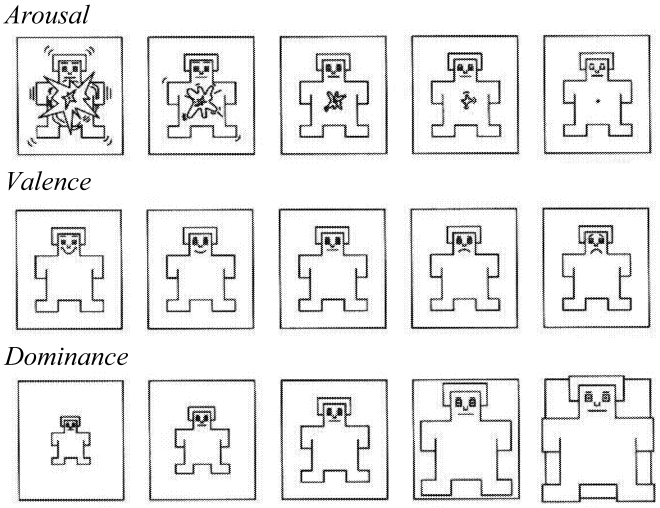


Figure 2: Self-Assessment Manikin.

Game

For the experiment we built a game as a Neverwinter Nights module using BioWare's Aurora toolset. The game starts with a short introduction that introduces the participant to the game controls. After that he encounters five conditions (situations). In each condition the player enters a different room, in which he has an interaction with a non-player character (NPC), and must collect an item, which he needs to open the next room. When all five items are collected, the participant can open a treasure chest and "win" the game. After leaving each of the rooms, the game asks him to fill out the SAM questionnaire. After the game is won, he is asked to fill out a short questionnaire, which includes a question on his experience with videogames.

The visual characteristics of the five conditions do not change in the game, apart from the ambient light color. The first condition is the control condition in which no manipulation takes place. In the other four conditions the color is manipulated. Color is therefore our independent variable. In order to prevent carry-over effects of the response set, the order of the colors of the four manipulated conditions are counterbalanced completely. As there are 24 different orderings for the four colors used in the experiment, our 60 participants were sufficient to test each ordering two or three times.

By manipulating the conditions' colors our aim was to stimulate feelings of surprise, fear, joy, and anger, which, according to Plutchik (2001), are associated with the colors light blue, dark green, yellow, and red (see Table 1). These feelings all should score high on the arousal dimension. However, whereas surprise and joy (light blue and yellow) should score high on the valence dimension, fear and anger (dark green and red) should score low on the valence dimension.

Analysis

To analyse the results of the measured emotional responses, a one-way ANOVA, an independent-samples T-test, and descriptive statistics were performed using SPSS 16.0. The scores on the SAM were the three dependent variables: arousal, valence, and dominance. Descriptive statistics of these variables were performed over the control condition, the rooms, and the four colors.

Descriptive statistics were also used to measure the number of participants, the mean score of the SAM dimensions, and the standard deviation for the four rooms, the control condition, and the four colors. A manipulation check was performed to analyze the differences between the rooms (conditions) by a one-way ANOVA. To test whether there is a main effect for color (including the control condition) on the arousal, valence, and dominance scores a one-way ANOVA was performed. We also examined whether experience or inexperience with videogames had an effect on emotional responses using one-way ANOVA tests.

RESULTS

In this section we discuss the SAM scores for the conditions (rooms), for the colors (and thus the emotions), and for the differences between experienced and inexperienced participants, respectively. Note that exact numerical details of the results are given by Joosten (2010).

SAM-scores for the rooms

The mean scores on the SAM for the four manipulated rooms are presented in Figure 3. From the results we conclude that a significant effect for the rooms was found on the valence and dominance scores. Scores for valence increased with the room numbers, with significant differences between rooms 1 and 4. Scores for dominance increased with the room numbers, with significant differences between rooms 1 and 3, rooms 1 and 4, and rooms 2 and 4. This means that, on average, a player's experience of pleasure and feeling of control increased while the game progressed.

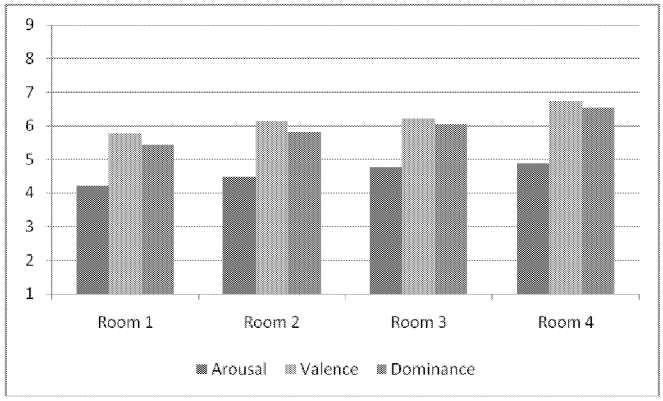


Figure 3: Mean scores on arousal, valence, and dominance for the four manipulated rooms.

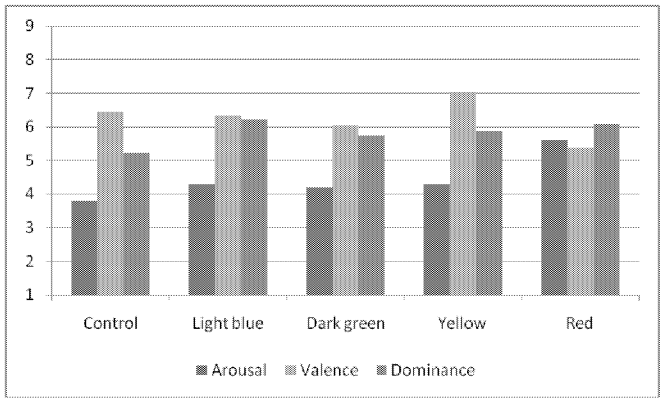


Figure 4: Mean scores on arousal, valence, and dominance for the control condition and the colors.

SAM-scores for the color conditions

The mean scores on the SAM for the four color conditions are presented in Figure 4. From the results we conclude that, on average, arousal scores for the color red were significantly higher than for any of the other colors or the control condition. There were no significant differences on arousal for the other colors. On average, valence scores for the color yellow were significantly higher than for the colors dark green and red, and for the control condition. There was no significant difference between the valence scores for yellow and light blue. Neither was there a significant difference for light blue and either dark green or red, or the control condition. The valence score for red, however, was

significantly lower than those for yellow, light blue, or the control condition. For dominance no significant effects were found for any of the colors. This means that, on average, more so than any other color used in the experiment, the color red arouses a player, but is experienced as negative. The color yellow, more than any other color used in the experiment, is experienced as positive. These results are statistically significant.

SAM-scores and experience

When examining our data, we hypothesized that there were differences in scores between participants who considered themselves experienced videogame players, and participants who considered themselves inexperienced in this area. We therefore calculated the mean scores on the SAM for the four color conditions for these two groups separately. The results are presented in Figures 5 and 6.

From the results we conclude the following: For the colors dark green, red, and the control condition the 38 inexperienced participants scored significantly lower on valence than the 21 experienced players (one participant did not answer the question on experience), i.e., they derived less pleasure from these conditions. For the colors yellow, red, and the control condition the dominance scores of inexperienced participants were also significantly lower than for experienced participants, i.e., they felt less in control in these conditions.

For experienced participants we find no significant differences between the scores on any of the colors for any of the dimensions. In contrast, for inexperienced participants we find three significant effects. First, the arousal score for the color red is significantly higher than the scores for any of the other colors, or the control conditions. One might even call the score 'surprisingly high.' Second, the valence score for the color yellow is significantly higher than the scores for the colors dark green, red, or the control condition. There was no significant difference with the color light blue, although the score for yellow is still quite a bit higher than the score for light blue. Third, the valence score for red was significantly lower than all the other colors or the control condition. For dominance, no significant effects are found.

The somewhat surprising conclusion that we must draw is that the effects we found were mainly the result of the answers given by the inexperienced participants.

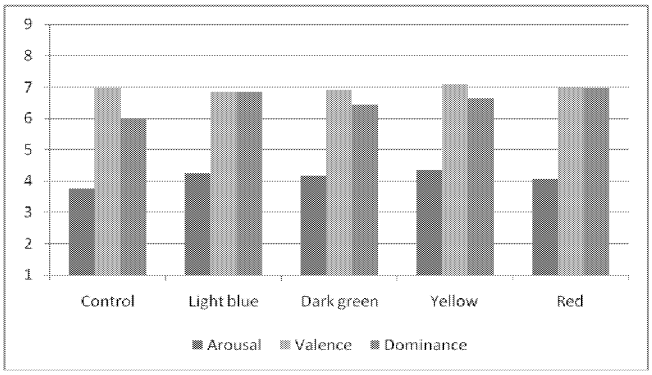


Figure 5: Mean scores on arousal, valence, and dominance of experienced videogame players for the control condition and the colors.

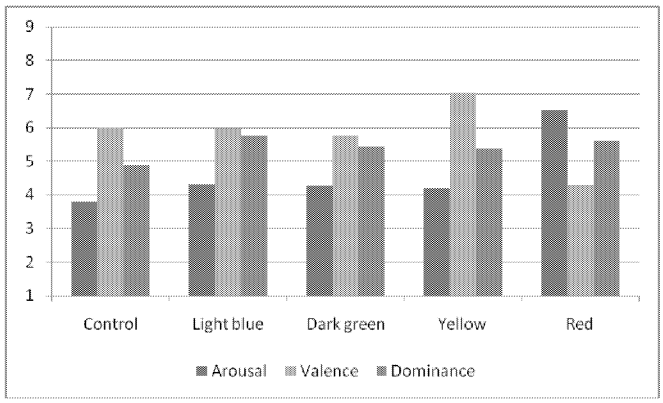


Figure 6: Mean scores on arousal, valence, and dominance of inexperienced videogame players for the control condition and the colors.

DISCUSSION

In this section we discuss our findings on the effects of the rooms, the colors, and player experience on emotional responses.

Effects of the rooms on emotional responses

Results on the differences between the rooms showed an effect on the valence and dominance dimensions. Differences on the valence scores between the rooms may be due to the enjoyment participants felt on winning the game. Participants felt more enjoyment in the last condition than in the first condition. Holbrook et al (1984) state that performance is an important factor for enjoyment in videogames.

The reason for the differences on the dominance scores between the rooms may be due to the fact that the participants got more experienced during the game, therefore navigation became easier (Griffith et al 1983) and participants felt more in control. This explanation is supported by the fact that experienced participants showed significantly higher dominance scores than inexperienced participants.

Effects of the colors on emotional responses

While we found effects on emotional responses for the colors yellow and red, we failed to find effects for the colors dark green and light blue. We provide three possible reasons for this lack of effect.

First, the lack of effect may be due to the environment used to measure emotional responses. The natural environments used by Plutchik (2001), who identified the relationship between colors and emotional responses, differ from ours. It is very well possible that color manipulation in a videogame leads to different emotional response effects than in a natural environment.

Second, the lack of effect may be due to the limited options for colour selection in the Aurora toolset. The colors in the light blue and dark green condition were less intense than the colours in the red and yellow condition. Valdez and Mehrabian (1994) stated that colors of high intensity elicit stronger emotional effects in humans than those of low intensity. Furthermore, the color of the light blue condition

was almost identical to the color of the control condition, and the color dark green made the condition very hazy, which might have a stronger effect on how a player experiences the game than just the color effect.

Third, the lack of effect may be due to the fact that in this experiment only the background light of a condition was manipulated. The objects in the condition and the condition itself were in standard colors.

Effects of experience on emotional responses

Emotional response effects for videogame experience were found on the valence and the dominance scores. In general experienced participants scored higher on these dimensions than inexperienced players. The higher scores for the experienced participants may be due to the perceived complexity of the videogame. Players who perceive less complexity during a videogame feel more enjoyment and have higher feelings of dominance (Holbrook et al 1984).

Emotional response effects for color were only found for the inexperienced participants. The reason for the lack of emotional response effects of color for the experienced participants may be due to the fact that they are familiar with videogame situations. They know what to expect and how to navigate through the game; their experience guides them more than the colors they see. Jørgensen (2008) stated that videogame players who are unfamiliar with a videogame situation use visual aspects (including color) of the videogame to make sense of the situation. Therefore, they take more note of visual cues, even subconsciously.

A further explanation for the lack of emotional response effects for experienced participants may be the time they spent in the conditions. In general, they were very fast in playing the game. Landers and Boutcher (1993) found that the duration of a videogame is an important factor in eliciting emotional responses.

CONCLUSION

We investigated to what extent the use of colors in a relatively complex videogame may influence a player's emotional responses. We investigated to what extent the use of colors in a relatively complex videogame may influence a player's emotional responses. We found significant effects on emotional responses for the colors red and yellow. According to Plutchik (2001), the color red is associated with a feeling of anger, which evokes a highly-aroused, negative emotional response, and the color yellow is associated with a feeling of joy, which evokes a highly-aroused, positive emotional response. Indeed, in our experiment we found that red elicited a highly-aroused, negative emotional response, and yellow elicited a positive emotional response. We found these color effects prevalent mainly with inexperienced videogame players. We conclude that game designers can employ certain colors to manipulate player's emotions, specifically those of novice players.

REFERENCES

Bradley, M.M. and Lang, P.J. 1994. "Measuring emotion: The Self-Assessment Manikin (SAM) and the semantic differential." *Journal of Experimental Psychiatry & Behavior Therapy*, 25, 49-59.

D'Andrade, R., and Egan, M. 1974. "The Color of Emotions." *American Ethnologist*, 1(1), 49-63.

Ekman, P. 1993. "Facial Expression and Emotion." *American Psychologist*, 48, 384-392.

El-Nasr, M., Niedenthal, S., Knez, I., Almeida, P., and Zupko, J. 2006. "Dynamic lighting for tension in games." *Game Studies*, 7(1).

Griffith, J.L., Voloschin, P., Gibb, G.D., and Bailey, J.R. 1983. "Differences in eye-hand motor coordination of video-game users and non-users." *Perceptual and Motor Skills*, 57(1), 155-158.

Holbrook, M.B., Chestnut, R.W., Oliva, T.A., and Greenleaf, E.A. 1984. "Play as a Consumption Experience: The Roles of Emotions, Performance, and Personality in the Enjoyment of Games." *Journal of Consumer Research*, 11(2), 728-739

Joosten, E. 2010. *The Emotions of Colour in Videogames*. MA thesis, Tilburg University, Tilburg, The Netherlands.

Jørgensen, K. 2008. "Audio and gameplay: an analysis of PvP battlegrounds in World of Warcraft." *Game Studies*, 8(2).

Landers, D.M. and Boutcher, S.H. 1993. "Arousal-performance relationships." *Applied sport psychology, Personal growth to peak performance* (J.M. Williams, ed.), 2nd edition, Mayfield Publishing Company, Palo Alto, CA.

Lang, P.J. 1980. "Behavioral treatment and bio-behavioral assessment: Computer applications." *Technology in mental health care delivery systems*, 119-167, Ablex, Norwood, NY.

Larsen, R.J. and Diener, E. 1992. "Promises and problems with the circumplex model of emotion." *Review of personality and social psychology*, 13, 25-59.

Livingstone, S.R. and Brown, A.R. 2005. "Dynamic response: real-time adaptation for music emotion." *Proceedings of the Second Australasian Conference on Interactive Entertainment*, 105-111.

Oberascher L. and Gallmetzer M. 2003. "Colour and emotion." *Proceedings of AIC 2003 Bangkok: Color Communication Management*, 370-374.

Ortony, A., and Turner, T.J. 1990. "What's basic about basic emotions?" *Psychology Review*, 97, 315-331.

Perron, B. 2005. "A cognitive psychological approach to gameplay emotions." *Conference Proceedings of DiGRA'05, Changing Views: Worlds in Play*, Digital Games Research Association.

Plutchik, R. (2001). The nature of emotions. *American Scientist*, 89, 344-350.

Posner, J., Russell, J.A., & Peterson, B.S. 2005. "The circumplex model of affect: An integrative approach to affective neuroscience, cognitive development, and psychopathology." *Development and Psychopathology*, 17, 715-734.

Ravaja, N., Salminen, M., Holopainen, J., Saari, T., and Laarni, J.J.A. 2004. "Emotional response patterns and sense of presence during videogames: potential criterion variables for game design." *Proceedings of the Third Nordic Conference on Human-Computer Interaction*, 339-347.

Robbins, S.P. and Judge 2008. *Organizational Behavior*. Prentice Hall International, Upper Saddle River, NJ.

Russell, J.A. 1980. "A circumplex model of affect." *Journal of Personality and Social Psychology*, 39, 1161-1178.

Stark, G.M., Saunders, D.M., and Wookey, P.E. 1982. "Differential effects of red and blue coloured lighting on gambling behavior." *Current Psychological Research*, 2, 95-100.

Valdez, P. and Mehrabian, A. 1994. "Effects of colour on emotions." *Journal of Experimental Psychology*, 123, 394-409.

Wolfson, S. and Case, G. 2000. "The effects of sound and colour on responses to a computer game." *Interacting with computers*, 13(2), 183-192.

EMOTION ASSESSMENT IN GAME PLAYING

L.J.M. Rothkrantz
Delft University of Technology
Mekelweg 4
2628 CD, Delft
The Netherlands
+31152787504

R. Janssen
Delft University of Technology
Mekelweg 4
2628 CD, Delft
The Netherlands

D. Datcu
The Netherlands
Defence Academy
Nieuwe Diep 8
1781 AC, Den Helder
The Netherlands

M.C. Popa
Delft University of Technology
Mekelweg 4
2628 CD, Delft
The Netherlands

KEYWORDS
Emotion recognition, Eye blink detection, EEG analysis, experiments.

ABSTRACT

At TUDelft there is a project running on the assessment of the emotional state of human operators. We used computer games as a simulation environment. Emotions are usual assessed by analysis of facial expressions and voice analysis. But unfortunately in general game players don't show their emotion in an overt way. In this paper we also report about assessment of emotions via eye-blink detection using camera's and EEG analysis. In a car simulation environment respondents have to drive different tracks under different stress conditions. We were able to assess the amount of stress by eye-blink detection. We report about the experimental design and results of analysis.

INTRODUCTION

Emotions play an important role in our daily life. They are part of our nonverbal behavior such as facial expressions, body posture or way of speaking. Emotions stimulate humans to perform some actions. This is also the case with game players. A player plays games because he expects to be the winner, is able to play roles which he can't play in real life or expects to have a lot of fun, or has to solve challenging problems. So a game player should be in a positive/active emotional state. If he starts loosing or the game becomes boring he will loose his interest and drops of. Game designers have to take care of the expected emotional state of their target group. In case it is possible to assess the emotional state of the player real time a challenging option is to design adaptive games. If a game player losses his interest because he is loosing all the time, the game player should be stimulated for example by lowering the game level, or start a new scenario or whatever. From the other side if the player gets bored, the level of the game should be raised or challenging scenarios should be introduced.
There is another interesting option for game designers if the emotional state of the player can be assessed. In current games the player meets emotional characters. In case a character shows extreme angriness then maybe it is not wise to attack him. But the emotional state of the player is hidden for the character, so he has no idea if he frightens his opponent, or his opponent is fooling him. So the character gets partial feedback of his behavior. Maybe that is one of the reasons of success of first person shooter games, the goal is simple, you have to kill as much of your opponents by just shooting and the feedback is clear, you hit or miss. If emotions are involved, games which are more similar to real life human interaction can be designed.
There is a relation between the emotional state of a person and his eye blink rate. Certain emotions can cause a person to blink much more rapidly than usual. Emotions such as fear and nervousness are likely to increase eye blinking as a way to clear the image in order to maintain the clearest vision possible. Charles Darwin observes that in stressful, threatening situation we want to have a clear picture of the situation and that is why we open our eyes and increase our blink rate (Darwin 1859).
At this moment nonverbal assessment of emotions has been realized by analysis of recorded facial expressions and recorded speech. One of the problems is that game players usually don't express their emotions by facial expressions or speech. Even some experienced Quake players play without any emotions. As soon as emotions are part of the game, players will show emotions either to show their emotional state or to fool the opponent. So one of our research questions was if game players have emotions, but hide their emotions. Next is it possible to assess emotions in a noninvasive way. In this paper we use brain computing via EEG analysis. The problem is that the emotional centre is located in the deeper and older parts of the brain, brainstem and amygdale for example. The question is if we are able to assess activities in the emotional centre, deep in the brain via EEG analysis via sensors attached to the skull. Finally we will use an infra-red camera to assess the blink rates.
In the next section we will present related work, next we present our models. Then we will report on our experiments and finally we will present the results.

RELATED WORK

Ever since the arrival of computers, people have been trying to detect and track faces on digital images. One of the best known solutions for face detection is the Viola Jones algorithm (Viola and Jones 2001). This technique uses a tree hierarchy to quickly scan an image for specifically designed features. Implemented on a conventional desktop computer, face detection works at 15 frames per second. Many improvements have been proposed and implemented

for the algorithm, resulting in techniques with even higher frame rates. The Viola&Jones algorithm is so wide spread that it even made it's way into Intel's OpenCV library (Intel 2000) for Computer Vision as a standard implementation for face detection. Although this method provides good results, the Face Detection problem is far from solved. Phenomena like occlusion, head posture, position, and facial hair can still be difficult to overcome.

A lot of research is already done in recognizing emotions. For example, research has been done to make computers recognize emotion from speech (Dellaert et al. 1996), facial expressions (Cowie et al. 2001) or a fusion of both methods (Busso et al. 2004). Another method of measuring human emotion is using physiological signals (Haag et al. 2004). These physiological signals are available at any moment, and it has been shown that emotional markers are present in electroencephalograms (EEG), and can be deduced from heartbeat rates and skin conductivity. Using these physiological signals also has the advantage that they can be hardly deceived by voluntary control and are available all the time, without needing any further action of the user. The disadvantage of using these signals is that the user has to wear some measurement equipment. This equipment could be very simple when measuring only heartbeat, but EEG measuring devices tend to be more demanding. This method using EEG signals can however still use a lot of improvement. Earlier research has shown that these types of human computer interfaces do not perform very well. They are able to recognize emotions correctly from around 60-70% of the samples given when recognizing three emotions (Kim et al. 2004). Choppin (Choppin 2000) suggests there is a lot of improvement possible when looking at different features or using other methods or techniques. A fusion of different methods could be another solution (Kim et al. 2004).

A lot of effort has gone into creating fully automatic drowsiness detectors. The main problem in classifying drowsiness is how to measure the entity. EEG scanners can monitor brain waves which can give an indication of the level of fatigue. Different researchers have different hypothesis as to what is the indicator for drowsiness. Some argue that the disappearance of Alpha activity in favor of Delta activity is viewed as crucial (Kim et al. 2004). While others mark the increase of power in the Alpha-Theta band as important. While trying to find the exact discriminator for fatigue, others have moved on using other techniques, while EEG scans are not very practical to apply in field situations. Those other efforts mainly focus on oculomotoric parameters. Blink frequency and duration remain the best examined oculomotoric indicators for current alertness state and drivers' ability to react to environmental stimuli (Choppin 2000). It is well known that fatigue is associated with increased blink frequency.

Eye-blink frequency and duration remain the best examined oculomotoric indicators for current alertness state and drivers' ability to react to environmental stimuli. It is well known that fatigue is associated with increased blink frequency. Blink and increased blink rate can also be an indicator for increased emotional activity.

Thus, a lot of research is devoted to extract the meaning of increased blinks, blink duration and saccades. Hargutt (Hargutt 2003) concluded that a state of light fatigue was indicated by an increased blink rate alone whereas the transition from severe sleepiness was accompanied by increased blink duration. Some argue, that blink frequency and blink duration have to be considered as two independent factors in blink behavior. A lid closure that lasts for more than 500 milliseconds and covers the pupil for that period is usually described as a micro sleep. However, lid closure alone is insufficient to detect severe sleepiness for all people.

TOOLS

Electroencephalography (EEG) uses the electrical activity of the neurons inside the brain. When the neurons are active, they produce an electrical potential. The combination of this electrical potential of groups of neurons can be measured outside the skull, which is done by EEG. Because there is some tissue and even the skull itself between the neurons and the electrodes, it is not possible to measure the exact location of the activity. Figure 1 shows an example of a cap for measuring EEG data.

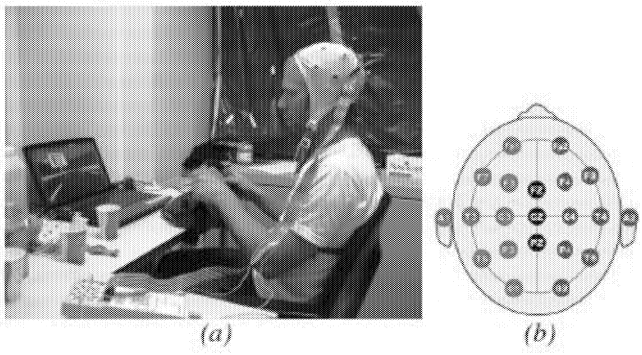


Figure 1: Respondent with brain cap using TORCS car simulator

For EEG measurements an array of electrodes is placed on the scalp. Many caps available use 19 electrodes, although the number of caps using more electrodes is rising. The electrodes are placed according to the international 10-20 system. This system extracts results from different research easily comparable. The brain activity can be coarsely divided into different classes. This division follows the way the activity is produced by the brain.

TORC

In order to perform the tests, a set of tools have been used. To mimic an in car environment, TORCS is used as a driving simulator along with a steering wheel (TORCS - The Open Racing Car Simulator). Furthermore, to be able to extract the number of blinks during a race, we chose to use an EEG scanner during some of the tests (EEG Scanner & TruScan – see figure 2). At the TU Delft the EEG scanner has been used in different experiments along the way, and

has proven that it can be indeed a valuable asset, given that it's used right. Once the EEG data and the output of the processor is acquired, all data has to be compared to draw any conclusions. Primarily MATLAB will be used for that task. Next we provide a description of these tools and platforms.

The Open Racing Car Simulator, or TORCS (see figure 3), is an open source race game developed for racing but for research as well. TU Delft has extended this game with races and billboards, making this platform suitable for performing such a test. Two different tracks have been used for testing.

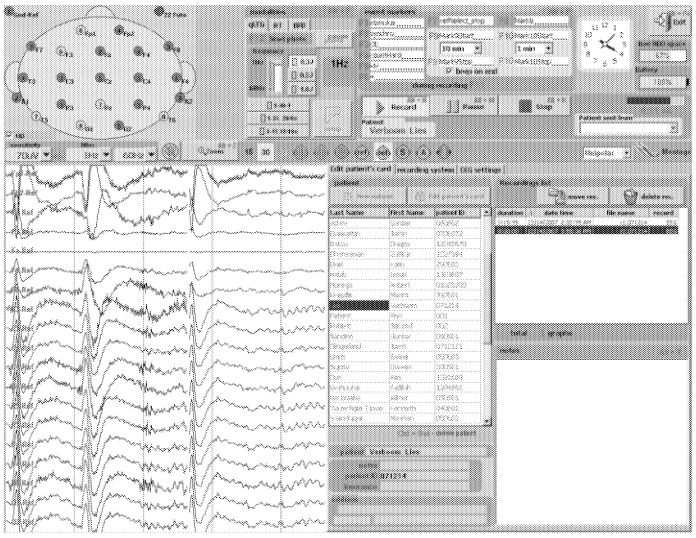


Figure 2: Tool TruScan to measure EEG signals

Figure 3: Screen of car simulator TORCS

Most of the tests where performed at the "Road Track 2", displayed in figure 4. Details of the billboards are shown in figure 5. This track is chosen because of it's approximate length (in minutes).

Figure 4: Racetrack and example of billboards

We wanted to have recordings of around 2,5 minutes, as to make sure the test subjects actually blink during the race and on the other hand we do not have too much data to analyze.

Along the track we place some billboards. Our assumption was that a relaxed driver observed the billboards which generate an increase in the blink rate or a typical P300 signal in the EEG recordings.

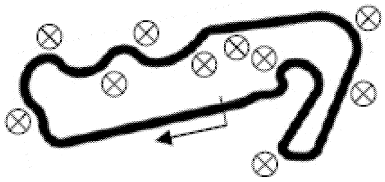


Figure 5: Race track, markers indicate the place of the billboards

ARCHITECTURE

Our system is composed of different modules. We will discuss next the different modules.

Eye blink detection

After capturing the data from video recordings we analyze every frame. We first locate the face using the Viola&Jones algorithm. Next we localize the eyes, eyebrows, and mouth in the face using the histogram algorithm. For every horizontal and vertical line we compute the grayness of adjacent pixels. If the difference in grayness passes some given threshold we raise the count by one (see figure 6). Let us consider some examples. If a horizontal line passes the area above the eyebrow the threshold will not be passed because the area has uniform grayness. But of the horizontal line passes the area of the eye we observe a lot of differences in grayness because of the hairs, color of the pupil, white area of the eye etc. The same holds for vertical lines. From the shape of the histograms we are able to find the location of the eyes, mouth, nostrils, etc. (see figure 7).

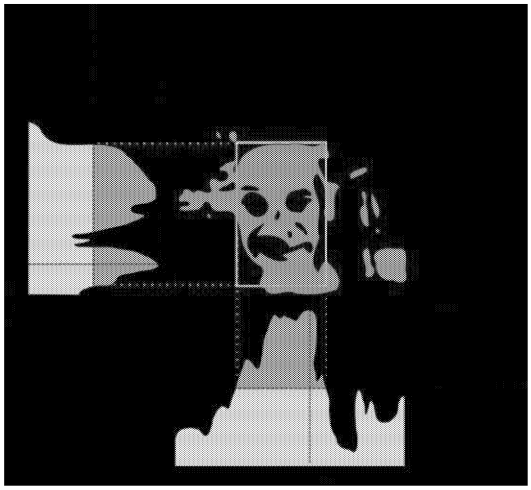


Figure 6: Fingerprints of histograms based on differences in grayness

Once we localized the eyes, we trained feed forward Neural Network using back propagation to find fudicial points on the contour of the eyes, such as the corners of the eyes and intersection of the eyeball with the lower and upper eyelid. Next we fit the inner contour of the eye via the localized fudicial points (see figure 8). The area of the inner contour is an indicator if the eyes are pen, half open, or closed.
In this way we are able to detect eye-blinks. To remove false positives and negatives we compare the results of adjacent frames. The used algorithms are relative simple. The main reason to use them was the following: our application requires real time processing. The chosen algorithm uses mainly matrix/vector computation. The software has been implemented on a special designed chip. The execution time is about 50 times faster than usual, so real time processing is possible.

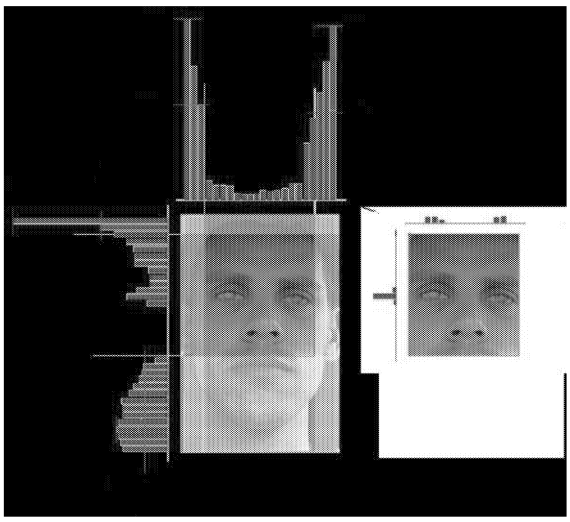


Figure 7: Location of the eyes

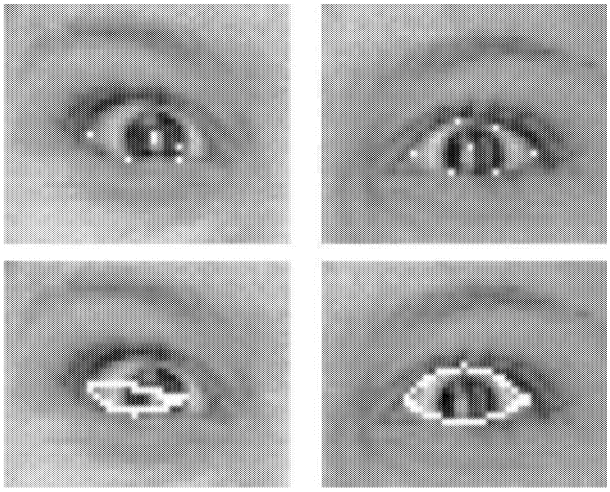


Figure 8: Location of fiducial points along the contours of the eyes

Eye blink detection using EEG analysis

The EEG signals of respondents are captured by the EEG cap and analyzed using special software implemented in TruScan software or MATLAB.
Eye blinks generate characteristic eye blinks patterns. From the location of the eye blinks graphs, we were able to compute the eye blink frequency.

Facial expression analysis

The detection of fiducial points as described in a previous section has a high error rate. We developed a more reliable method using the Active Appearance Model - AAM. The AAM model uses not only the geometrical characteristics of the face but also texture properties. We defined a model of the contours of eyes, eyebrows, and mouth as displayed in figure 9. After training the model on a set of annotated data of facial expressions we were able to find the fiducial points in a reliable way. The model can be used in two ways. First, it can be used to analyze single frames containing a picture of a face. Secondly, regarding the videos, we analyze single frames and relate the positions of the points as displayed in figure 10. The graphs of the tracked fiducial points are characteristic for different emotions. In this way we are able to label the facial expressions.

Figure 9: Face model with contours of eyes, eyebrows, mouth and chin

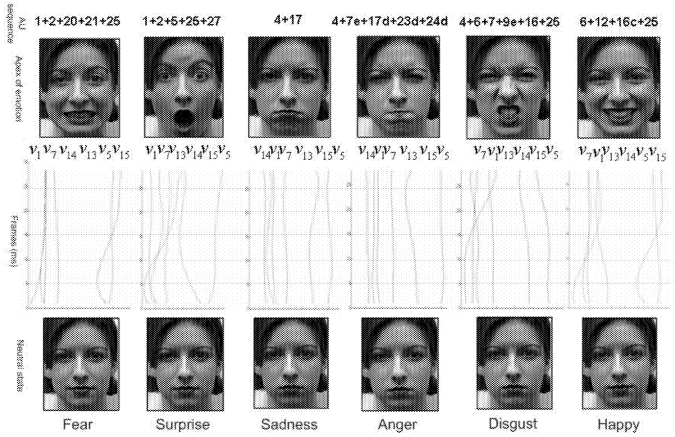


Figure 10: Tracks of fiducial points

EXPERIMENTS AND RESULTS

We selected 13 students form Delft University of Technology to take part in our experiments. We considered two experimental conditions. In the free race conditions we took recordings of the face of the respondents using a video camera and in the EEG mode we took EEG recordings. Students were requested to play the race game TORC under several experimental conditions. They had to drive carefully or as fast as possible. The track was with or without billboards. In all cases we measured the amount of eye-blinks under the two experimental conditions. In table 1 we display the result.

Table 1: Results of detecting eye blinks

		Test 1	Test 2
	code name	Free Race	EEG Race
Stats	Participants #	13	13
	Male %	92.5	92.5
	Female %	7.5	7.5
	Facial hair %	15.4	15.4
	Glasses %	30.7	30.7
	avg. blinks #	19.31	12.77
	avg. detection rate %	50.37	45.71
Details	total blinks	251	166
	false positive #	110	157
	false negative #	36	93
	blinks detected #	112	60

In figure 11 we present the results of experiments in localizing the face and eye-region.

We were able to detect the face and the eye-region with high accuracy. We were able to detect the eye-blinks in half of the cases. We missed a lot of eye-blinks because of the following reasons:

- we are able to detect eye-blinks in the frontal case. If the head of respondents is rotated the systems fails. We assumed that the respondents were always looking to the screen but this assumption was not true. In many cases the respondents look away to the tester or to the keyboard or whatever.
- sometimes the eye was occluded by hands.

Detect	Approach	Recognition Rate	
		little movement	normal movement
Face	color	99.00%	99.00%
	shape	81.00%	56.00%
	movements	67.00%	86.00%
Detect	**Approach**	**Recognition Rate**	
		little movement	normal movement
Eye region	Side rectangles	85.00%	59.00%
	Prediction	87.00%	82.00%

Figure 11: Results of localization of the face and eye region

CONCLUSIONS AND FUTURE WORK

We are satisfied about the modules of the system localizing the face and facial areas. But if respondents move the position of their heads our system for eye-blink detection fails. We can solve this problem by attaching the camera to the head. But this is not what we want.

Next future we will train our eye-blink detection system for many orientation and posture of the head/face.

Our system for detection of facial expressions has a high recognition score.

So we can conclude that we are able to detect the emotional state of the player by analysis of nonverbal behavior, but there is still room for improvement.

REFERENCES

Busso, C.; Z. Deng; S. Yildirim; M. Bulut; C.M. Lee; A. Kazemzadeh; S. Lee; U. Neumann; and S. Narayanan. 2004. “Analysis of emotion recognition using facial expressions, speech and multimodal information”. In *ICMI 04: Proceedings of the 6th international conference on Multimodal interfaces*. New York, NY, USA, ACM Press. 205-211.

Choppin A. 2000. “EEG-based human interface for disabled individuals: Emotion expression with neural networks”. Master's thesis, Tokyo Institure of Technology.

Cowie, R.; E.D. Cowie; N. Taspatsoulis; G. Votsis; S. Kollias; W. Fellenz; and J.G. Taylor. 2001. “Emotion recognition in human-computer interaction”. In *Signal Processing Magazine*. IEEE, 18-32.

Darwin C.R. 1859. “On the origin of the species”. John Murray.

Dellaert, F.; T. Polzin; and A. Waibel. 1996. “Recognizing emotion in speech”. In *ICSLP 96*. Proceedings, volume 3, 1970-1973.

Haag, A.; S. Goronzy; P. Schaich; and J. Williams. 2004. “Emotion recognition using bio-sensors: First steps towards an automatic system”. *Lecture Notes in Computer Science*, 3068:36-48.

Hargutt. 2003. "Das Lidschlagverhalten als Indikator für Aufmerksamkeits- und Müdigkeitsprozesse bei Arbeitshandlungen". Düsseldorf:VDI Verlag.

Intel. 2000. "Intel's OpenCV Library" http://opencv.willowgarage.com/wiki/" - Website.

Kim K.H.; S.W. Bang; and S.R. Kim. 2004. "Emotion recognition system using short term monitoring of physiological signals". *Medical and Biological Engineering and Computing*, 42:419-427.

Viola P. and M. Jones. 2001. "Robust Real-time Object Detection". In *Second International workshop on statistical and computational theories of vision*.

BIOGRAPHIE

LEON ROTHKRANTZ is a Professor of Computer Science at the Netherlands Defence Academy and at Delft University of Technology. He has a MSc degree in Mathematics but also in Psychology. He received his PhD in Mathematics from the University of Amsterdam. His research interests are multimodal communication, artificial intelligence, signal processing, and sensor technology.

REMCO JANSSEN studied computer science at the Delft University of Technology and received his MSc degree at the same university in 2010.

DRAGOS DATCU received his PhD degree in Computer Science at the Delft University of Technology in 2009 and is currently working as a PostDoc at the Netherlands Defence Academy. His research interests are in emotion recognition, artificial intelligence, and 3D facial modeling.

MIRELA POPA is a PhD student at Delft University of Technology. She has a PDEng degree in Software Technology and a MSc degree in Algorithms and Data Structures. She is mainly interested in multimodal fusion, behavior assessment, and emotion analysis.

INVOLVING PLAYER EXPERIENCE IN DYNAMICALLY GENERATED MISSIONS AND GAME SPACES

Sander Bakkes and Joris Dormans
Amsterdam University of Applied Sciences
Computer Science Dept., section Game Development / CREATE-IT Applied Research
P.O. Box 1025, NL-1000 BA Amsterdam, The Netherlands
e-mail: {s.c.j.bakkes, j.dormans}@hva.nl

ABSTRACT

This paper investigates strategies to generate levels for action-adventure games. This genre relies more strongly on well-designed levels than rule-driven genres such as strategy or role-playing games for which procedural level generation has been successful in the past. The approach outlined by this paper distinguishes between missions and spaces as separate structures that need to be generated in two individual steps. It discusses the merits of different types of generative grammars for each individual step in the process. Notably, the approach acknowledges that the online generation of levels needs to be tailored strictly to the actual experience of a player. Therefore, the approach incorporates techniques to establish and exploit player models in actual play.

INTRODUCTION

In the domain of video games, procedurally generated content is considered to be of great importance to the computer-game development in the present and in the future; both offline, for making the game development process more efficient (design of content such as environments and animations now consume a major part of the development budget for most commercial games), and online, for enabling new types of games based on player-adapted content [1]. In fact, games with procedurally generated content have been around for some time, though in a severely limited form.

The classic example of this type of game is *Rogue*, an old *Dungeons & Dragons* style ASCII dungeon-crawling game which levels are generated every time the player starts a new game. The typical approach of these games can be classified as a brute-force random algorithm that is tailored to the purpose of generating level structures that function for the type of game. Although these algorithms have a proven track-record for the creation of (relatively uncomplicated) rogue-like games, the gameplay their output supports is rather limited.

What is more, Kate Compton and Michael Mateas point out that generating levels for a generally relatively complex action platform game is more difficult as level design is a far more critical aspect of that type of game [2]. Action-adventures rely on level design principles that result in enjoyable exploration, flow and narrative structure, too. As it turns out, these principles are difficult to implement with the algorithms commonly encountered in rogue-like games. The algorithms generally cannot express these principles as they mostly operate on a larger scale than the scale of individual dungeon rooms and corridors. In order to generate game levels informed by such principles we need to turn to a method that does operate on the scale on which these principles reside. This method is the use of *generative grammars*. However, even with the use of generative grammars, generating good levels is still very hard. Levels often have a random feel to it and tend to lack overall structure. To search simply for a single generative grammar to tackle all these problems is not sufficient. Well-designed levels generally have two, instead of one structures; a level generally consists of a mission and a space.

This paper suggests that both missions and spaces are best generated separately using types of generative grammars that suit the particular needs of each structure. As outlined in the final sections of this paper, the route presented here is to generate missions first and subsequently generate spaces to accommodate these missions. We acknowledge that the online generation of levels needs to be tailored strictly to the actual experience of a player. Therefore, the approach incorporates techniques to establish and exploit player models in actual play.

MISSIONS AND SPACES

In a detailed study of the level design of the Forest Temple level of *The Legend of Zelda: The Twilight Princess*, conducted by the authors and described in more detail elsewhere [3], two different structures emerge that both describe the level. First, there is the geometrical lay-out of the level: the space. Level space can be abstracted into a network of nodes and edges to represent rooms and their connections. Second, there is the series of tasks the player needs to complete in order to get to the end of the level: the actual mission. The mission can be represented by a directed graph indicating which tasks are made available by the completion of a pre-

ceding task. The mission dictates a logical order for the completion of the tasks, which is independent of the geometric lay-out. As can been seen in Figure 1, the mission can be mapped to the game space. In this case certain parts of the space and the mission are isomorphic. In particular, in the first section of the level mission and space correspond rather closely. Isomorphisms between mission and space is frequently encountered in many games, but the differences between the two structures are often just as important.

Level space accommodates the mission and the mission is mapped onto the space, but otherwise the two are independent of each other. The same mission can be mapped to many different spaces, and one space can support multiple different missions. The principles that govern the design of both structures also differ. A linear mission, in which all tasks can only be completed in a single, fixed order, can be mapped onto a non-linear spatial configuration. Likewise, a non-linear mission featuring many parallel challenges and alternative options, can be mapped on to a strictly linear space, resulting in the player having to travel back and forth a lot.

Some qualities of a level can ultimately be attributed to its mission while others are a function of its space. For example, in *Zelda* levels, and indeed in many Nintendo games, it is a common strategy to gradually train the player in the available moves and techniques. In addition, numerous missions follow a Hollywood-like structure, that can be attributed to Joseph Campbell's monomyth (*cf.* [4]).

The spatial qualities of the Forest Temple are distinct. Its basic layout follows a hub-and-spoke layout that provides easy access to many parts of the temple. The boomerang acts as key to many locks that can be encountered right from the beginning. Once it is obtained extra rooms in the temple are unlocked for the player to explore, a structure frequently found in adventure games [5].

GENERATIVE GRAMMARS

Before detailing our approach to dynamically generated missions and game spaces, we introduce the concept of generative grammars, and provide a discussion on the advantages and applications of generative grammars.

Concept. Generative grammars originate in linguistics where they are used as a model to describe sets of linguistic phrases. In theory, a generative grammar can be created that is able to produce all correct phrases of a language. A generative grammar typically consists of an alphabet and a set of rules. The alphabet is a set of symbols the grammar works with. The rules employ rewrite operations: a rule specifies what symbol can be replaced by what other symbols to form a new string. For example: a rule in a grammar might specify that in a string of symbols, symbol 'S' can be replaced by the symbols 'ab'. This rule would normally be written down as 'S → ab'. Generative grammars typically replace the symbol (or group of symbols) on the left-hand side of the arrow with a symbol or group of symbols on the right-hand side. Therefore, it is common to refer to the symbols to be replaced as the left-hand side of the rule and to refer to the new symbols as the right-hand side. Some symbols in the alphabet can never be replaced because there are no rules that specify their replacement. These symbols are called terminals and the convention is to represent them with lowercase characters. The symbols 'a' and 'b' in the last example are terminals. Non-terminals have rules that specify their replacement and are conventionally represented by uppercase characters. The symbol 'S' from the previous rules is an example. For a grammar that describes natural language sentences, terminal symbols might be words, whereas non-terminal symbols represent functional word groups, such as noun-phrases and verb-phrases. The denominator 'S' is often used for a grammar's start symbol. A generative grammar needs at least one symbol to replace; it cannot start from nothing. Therefore, a complete generative grammar also specifies a start symbol.

Grammars like these are used in computer science to create language and code parsers; they are designed to analyse and classify language. Moreover, grammars are suited for predicting, and generating automatically language phrases. We utilise grammers for this latter purpose. It is easy to see that simple rules can produce quite interesting results especially when the rules allow for recursion: when the rules produce non-terminal symbols that can directly or indirectly result in the application of the same rule recursively. The rule 'S → abS' is an example of a recursive rule and will produce endless strings of ab's. The rule 'S → aSb' is another example and generates a string of a's followed by an equal number of b's. Generative grammars developed for natural languages are capable of capturing concepts that transcend the level of individual words, such as argument construction and rhetoric, which suggests that generative grammars developed for games should be able to capture higher level design principles that lead to interesting levels at both micro and macro scopes.

Generative grammars can be used to describe games when the alphabet of the grammar consists of a series of symbols to represent game specific concepts, and the rules define sensible ways in which these concepts can be combined to create well-formed levels. A grammar that describes the possible levels of an adventure game, for example, might include the terminal symbols 'key', 'lock', 'room', 'monster', 'treasure'. While the rules for that grammar might include:

1. Dungeon → Obstacle + treasure
2. Obstacle → key + Obstacle + lock + Obstacle
3. Obstacle → monster + Obstacle
4. Obstacle → room

In this case, when multiple rules specify possible replacements for the same non-terminal symbol, only one rule will be selected. This can be done (pseudo-)randomly.

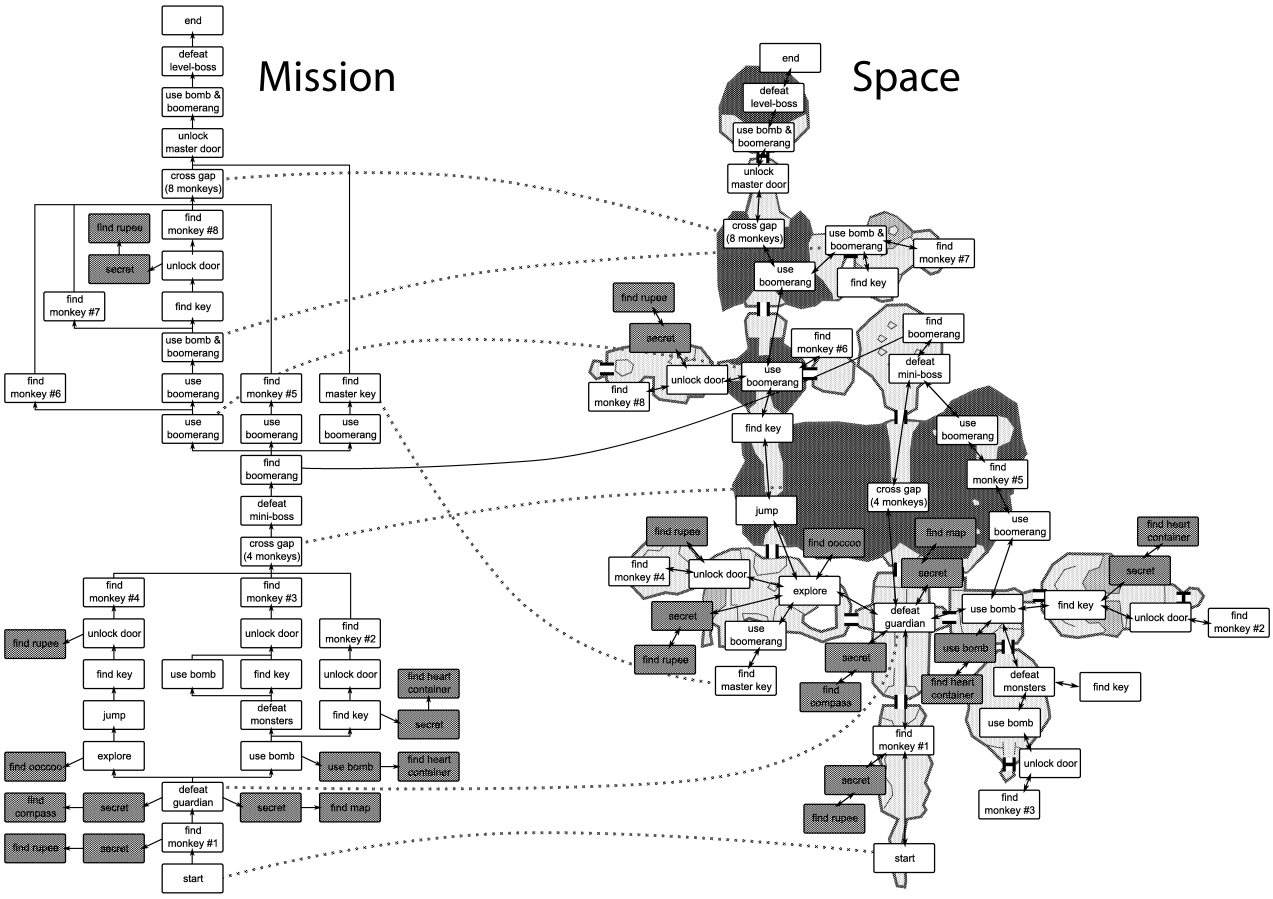


Figure 1: Mission and space in the Forest Temple Level of *The Legend of Zelda: The Twilight Princess*.

The rules can generate a wide variety of strings including:

1. key + monster + room + lock + monster + room + treasure
2. key + monster + key + room + lock + monster + room + lock + room + treasure
3. room + treasure
4. monster + monster + monster + monster + room + treasure

The strings produced by the grammar discussed above are not all suited for a game level. For instance, string 3 is too short (and uninteresting) even in the limited example above. The problem stems not from generative grammars as such, but from quality of the rules that are used in the example. In fact, generative grammar can easily counter these problems by creating rules that capture level design principles better, such as:

1. Dungeon → Obstacle + Obstacle + Obstacle + Obstacle + treasure
2. Dungeon → Threshold Guardian + Obstacle + Mini-Boss + reward + Obstacle + Level-Boss + treasure.

Where rule 1 incorporates the idea that a dungeon needs to have a minimal length to be interesting at all, and rule 2 directly incorporates a three act story structure like the one described for Forest Temple level of *Zelda: The Twilight Princess* above.

Advantages and applications. Generative grammars can be used in different ways to produce content for games. Game experts and designers can produce a grammar to generate content for a particular game. Drafting such a grammar would by no means be an easy task, but the initial effort vastly outweighs the ease by which new content can be generated or adjusted. Furthermore, grammars and procedurally content can be used to aid the designer by automating some, but not all, design tasks. This approach was taken by Epic Games for the generation of buildings and large urban landscapes. It proved to be very versatile as it allowed designers to quickly regenerate previous sections with the same constraints but with new rule sets without having to redo a whole section by hand [6]. Finally, it would be possible to grow grammars using evolutionary algorithms that select successful content from a test environment. The grammars presented in this paper were all drafted using the first method. Evolutionary grammars, although a tantalizing concept, are beyond the scope of the material presented here. Relevant applications of generative grammars can also be found in Lindermayer Systems (L-Systems), for instance for the procedural generation of city models [7].

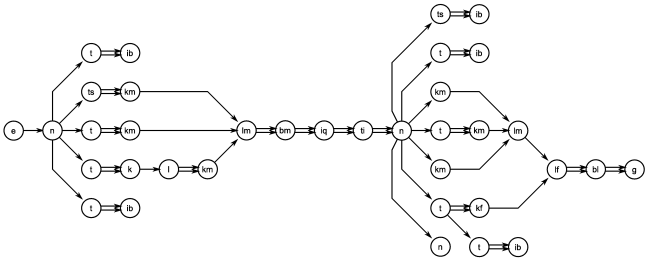


Figure 3: Example of a generated mission, based on the rules given in Figure 2.

GRAPH GRAMMAR TO GENERATE MISSIONS

Graph grammars are a specialized form of generative grammars that produce graphs consisting of edges and nodes, instead of producing strings. In relation with level generation, graph grammars are discussed by David Adams in his 2002 thesis *Automatic Generation of Dungeons for Computer Games* [8]. In a graph grammar, one or several nodes and interconnecting edges can be replaced by a new structure of nodes and edges. For examples we refer the reader to [3].

Graph grammars are well suited to generate missions, as missions are best expressed as nonlinear graphs. It would need an alphabet that consists of different tasks, including challenges and rewards. Figure 2 proposes several rules to generate a mission structured similarly as the mission of the forest temple. Figure 3 shows sample output of the graph grammar. We note that this grammar includes two types of edges, represented by single arrows and double arrows; different types of edges is a feature that can be found in other graph grammars. In this case, the double edges indicate a tight coupling between the subordinate node and its super-ordinate: this means that the subordinate must be placed behind the superordinate in the generated space. It is specific to the implementation described in this paper. A normal edge represents a loose coupling and indicates that the subordinate can be placed anywhere. This information is very important for the space generation algorithm (see section *Generating Space from Mission*).

SHAPE GRAMMAR TO GENERATE SPACE

Shape grammars are most useful to generate space. Shape grammars have been around since the early 1970s after they were first described by George Stiny and James Gips [9]. Shape grammars shapes are replaced by new shapes following rewrite rules similar to those of generative grammar and graph grammar. Special markers are used to identify starting points and to help orientate (and sometimes scale) the new shapes.

For example, imagine a shape grammar, which alphabet consists of three symbols: 'a wall', 'open space' and a 'connection' (see Figure 4a). In this grammar only the 'connection' is a non-terminal symbol, which has a square marker with a triangle indicating its orientation. The grey marker on the right-hand side of a shape grammar rule as represented here, indicates where the original shape was, and what its orientation was. We can design rules that determine that a connection can be replaced by a wall (effectively closing the connection), a short piece of corridor, or a T-fork (see Figure 4b). The construction depicted in Figure 4c is a possible output of these rules, provided that the start symbol also was a connection, and given that at every iteration a random connection was selected to be replaced.

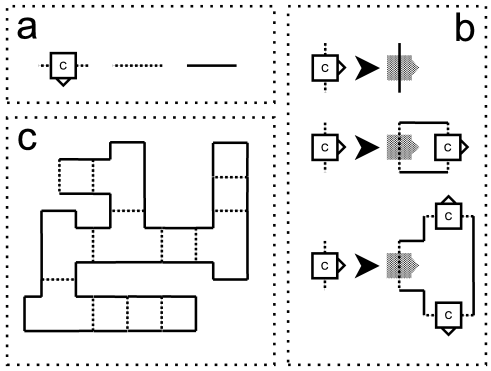


Figure 4: Shape grammar a) alphabet, b) rules, and c) output.

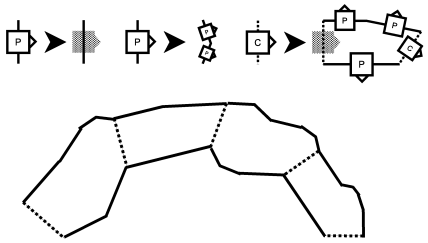


Figure 5: Recursive shape rules and its output.

Shape grammars, like any generative grammar can include recursion. Recursion is a good way to introduce more variation in the resulting shapes. An interesting possibility is that hereby it provides shapes to grow in a certain direction. In this case the implementation of the grammar should allow the right-hand side to be resized to match the size of the growing shape. For instance, the rules in Figure 5 are recursive and the shapes these rules produces will have a more natural (fractal) feel.

GENERATING SPACE FROM MISSION

In order to use a shape grammar to generate a space from a generated mission a few adjustments need to be made to the shape grammar. The terminal symbols in

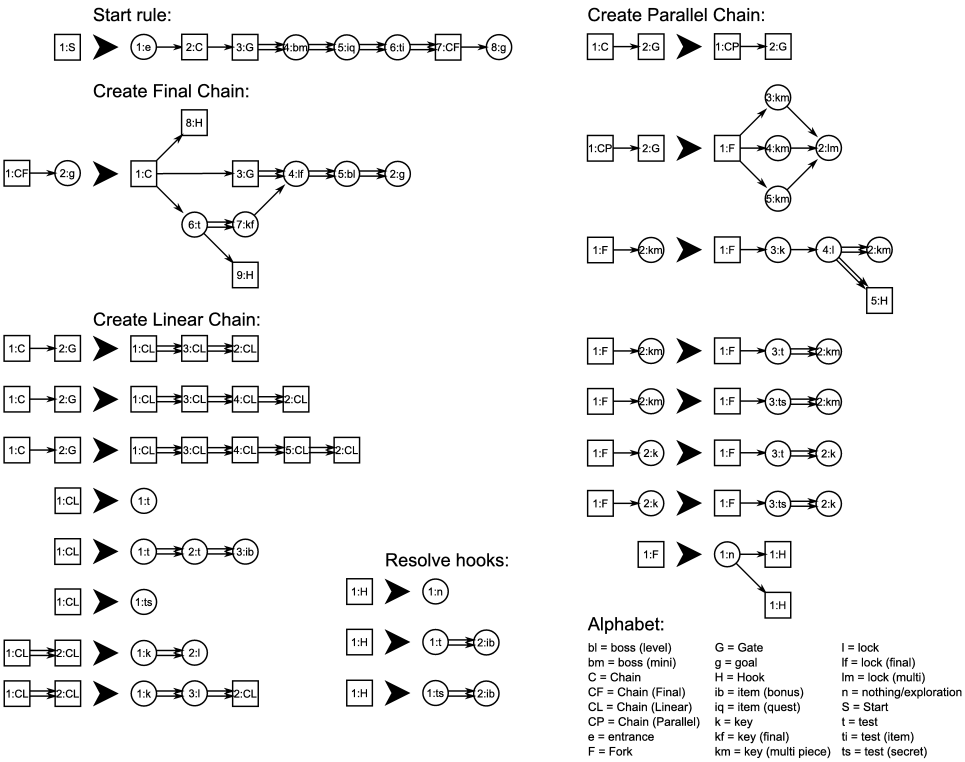


Figure 2: Example of rules to generate a mission.

the mission need to function as building instructions for the shape grammar. To achieve this, each rule in the shape grammar is associated with a terminal symbol form in the mission grammar. The prototype that implements the shape grammar first finds the next symbol in the mission, looks for rules that implement that symbol, selects one pseudo-randomly based on their relative weight, then looks for possible locations where the rule could be applied, and finally selects one location pseudo-randomly based on their relative fitness (one location might be more suitable than another).[1] The algorithm stores a reference to the mission symbol for which each element was generated, allowing the algorithm to implement the tight coupling as dictated by the mission. This prevents the algorithm from placing keys and items at random locations instead of behind, e.g., tests or locks as specified by the mission. The shape grammar is further extended by dynamic parameters that influence the rule selection. These parameters are used to create progressive difficulty or to shift between different 'registers'. For example the grammar can increase the chance of selecting rules with more difficult obstacles with every step, and switch from a register that causes it to build many traps to a register that causes it to include many monsters.

In the prototype application supporting this research,

[1]The prototype in which we incorporate the ideas proposed in this paper, is available online, at http://www.jorisdormans.nl/dungeoncrawl

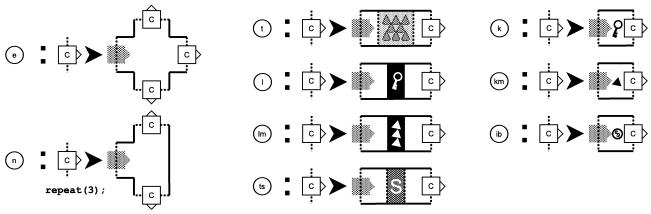


Figure 6: Shape grammar rules to generate missions.

rules can have commands associated with them. These commands are executed either before or after the application of a rule. These commands, among other functions, facilitate dynamic rule weights and progressive difficulty.

Once the complete mission is accounted for, the shape grammar will continue to iterate until all non-terminals are replaced with terminal symbols using a set of rules designed to finalize the space (or, alternatively, to grow some additional branches). Figure 6 lists several rules for a shape grammar constructed in this way. Figure 7 illustrates a few iterations in the construction of a level based on the first part of the mission presented in Figure 3 above.

In practise, it is not be very difficult to generate maps that can accommodate multiple missions. Missions can be blended, with the generator alternating between missions when selecting the next task to accommodate on

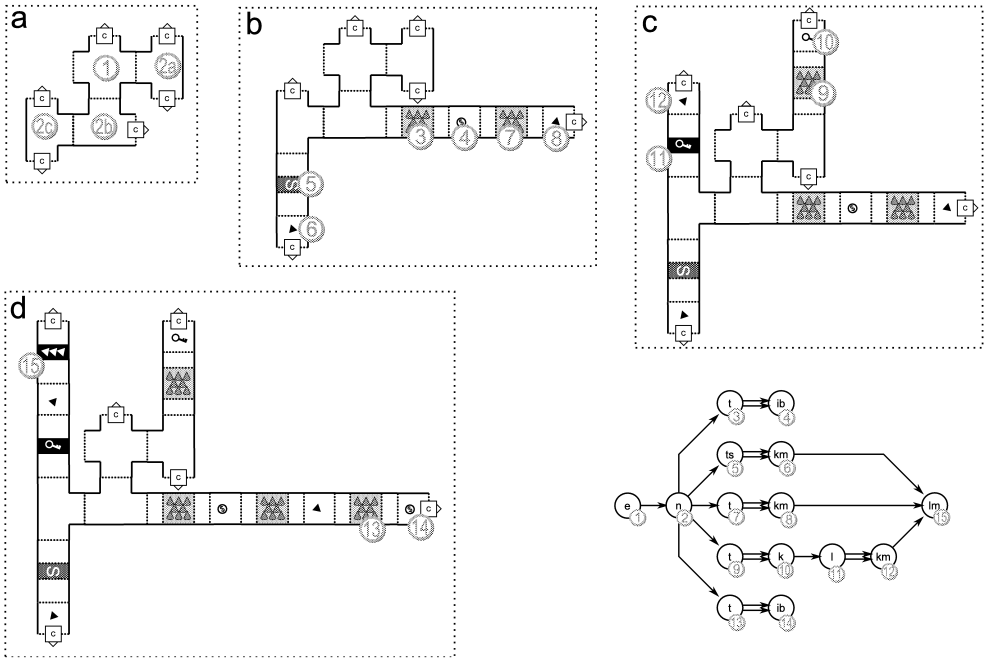


Figure 7: Space generation using the rules from Figure 6 and part of the mission from Figure 3 (depicted in the lower-right), over four iterations.

the map. Alternatively, a second mission is used as building instructions after the first mission has been completely accounted for.

ESTABLISHING PLAYER MODELS

The generation techniques discussed in this paper can also be employed to generate (or adapt) levels during play, allowing for the opportunity to *let behaviour of the player impact the generation*. To this end, techniques for player modelling need to be incorporated.

Player modelling is an important research area in game playing. It concerns establishing models of the player (discussed in this section), and exploiting the models in actual play (discussed in the section that follows). In general, a player model is an abstracted description of a player or of a player's behaviour in a game. In the case the models concern not the human player, but instead an opponent player, we speak of 'opponent modelling'. The goal of opponent modelling is to raise the playing strength of the (computer-controlled) player by allowing it to adapt to its opponent and exploit his weaknesses [10]. On the other hand, the goal of player modelling generally is to steer the game towards a predictably high player satisfaction [11], on the basis of modelled behaviour of the human player.

Player modelling is of increasing importance in modern video games [12]. Houlette [13] discussed the challenges of player modelling in video-game environments, and suggested some possible implementations of player modelling. A challenge for player modelling in video games is that models of the player have to be established (1) in game environments that generally are relatively realistic and relatively complex, (2) with typically little time for observation, and (3) often with only partial observability of the environment. Once player models are established, classification of the player has to be performed in real time. As by approximation only twenty percent of computing resources are available to the game AI [14], only computationally inexpensive approaches to player modelling are suitable for incorporation in the game.

For the domain of modern video games, we deem three approaches applicable to player modelling, namely (1) action modelling, (2) preference modelling, and (3) player profiling.

Action modelling. In video games, a common approach to establishing models of the player is by modelling the *actions* of the player [15]. It has been shown that it is possible to create a model of the actions that players tend to take in particular game situations [13]. Although such models are generally quite useful, the actual action that a player takes in a particular game situation will depend not only on the situation but also on the player's overall game preferences (e.g., adopting a particular strategy). For instance, in one play of the game a player may want to win by purely military means whereas in another they may decide to aim for cultural dominance. In general, a player's preference is not easily determined on the basis of only the current game situation, and hence purely action-based models tend to be of limited applicability [15]. What is more, the interest of game developers generally goes out to de-

termining a player's actual game experience, which, by implication, is even less easily determined solely on the basis of action-based models.

Preference modelling. An alternative, more advanced approach is to model the *preferences* of players, instead of the actions resulting from those preferences. This preference-based approach is viable [15] and identifies the model of a player by analysing the player's choices in predefined game situation. In the preference-based approach, player modelling can be seen as a classification problem, where a player is classified as one of a number of available models based on data that is collected during the game. Behaviour of related game AI is established based on the classification of the player. Modelling of preferences may be viewed as similar to approaches that regard known player models (1) as stereotypes, and (2) as an abstraction of observations [16]. As a means to generalise over observed game actions, a preference-based approach may be of interest to game developers.

Player profiling. A recent development with regard to player modelling, is to establish automatically psychologically verified *player profiles*. Player profiling has gathered substantial research interest (*cf.*, e.g., [1]). Van Lankveld [17] states that the major differences between player modelling (by means of action modelling, or preference modelling) and player profiling, lie in the features that are modelled. That is, player modelling generally attempts to model the player's playing style (e.g., playing defensively), while player profiling attempts to model traits of the player's personality (e.g., extraversion). The models produced by player profiling are readily applicable in any situation where conventional personality models can be used. In addition, player profiling is supported by a large body of psychological knowledge.

EXPLOITING PLAYER MODELS

Here, we propose three approaches for exploiting player models in the context of generative grammars, namely for (1) space adaptation, (2) mission adaptation, and (3) difficulty scaling.

Space adaptation. A natural starting point for exploiting player models, is to allow the space in which the game is played to grow in response to the actual behaviour of the player. Firstly, after observing the player for some time, features within the established player model may indicate that it is interesting to transform (gradually) the game surroundings. For instance, from open to confined spaces, from linear to more organic environments, and from easily maneuverable corridors to intricate mazes. Secondly, varied gameplay may be provided by also allowing events that take place within certain game spaces (e.g., particular rooms) to respond to the player's previous behaviour. For instance, if the player already has encountered and fought many monsters, the rules that would generate more monsters might decrease in weight while rules that would generate obstacles of a different type might increase in weight. Or, inversely, when the player model indicate that the player enjoys combating monsters (for example because he goes after every monster he can find), the game may confront him with more and tougher creatures. The possibilities of a feedback loop between the actual performance of the player and the online generation of the game, are many.

Mission adaptation. An interesting alternative to space adaptation, is to allow the game's mission to grow in response to the actual behaviour of the player. A strategy in this regard, which we regard as promising, would be to generate a mission that still has some non-terminals in its structure before constructing the space. The replacement of these non-terminals should occur during play, and should be informed by the performance of the player directly or indirectly. The space could either grow in response to the changes in the mission, or already have accommodated all possibilities. This could quite literally lead to an implementation of an interactive structure that Marie-Laure Ryan calls a fractal story; where a story keeps offering more and more detail as the player turns his attention to certain parts of the story [18].

Difficulty scaling. Player models may potentially be applied to adapt automatically the challenge that a game poses to the skills of a human player. This is called difficulty scaling [19], or alternatively, challenge balancing [20]. When applied to game dynamics, difficulty scaling aims usually at achieving a "balanced game", i.e., a game wherein the human player is neither challenged too little, nor challenged too much.

In most games, the only implemented means of difficulty scaling is typically provided by a *difficulty setting*, i.e., a discrete parameter that determines how difficult the game will be. The purpose of a difficulty setting is to allow both novice and experienced players to enjoy the appropriate challenge that the game offers. Usually the parameter affects plain in-game properties of the game opponents, such as their physical strength. Only in exceptional cases the parameter influences the strategy of the opponents. Consequently, even on a "hard" difficulty setting, opponents may exhibit inferior behaviour, despite their, for instance, high physical strength. Because the challenge provided by a game is typically multifaceted, it is tough for the player to estimate reliably the particular difficulty level that is appropriate for himself. Furthermore, generally only a limited set of discrete difficulty settings is available (e.g., easy, normal, and hard). This entails that the available difficulty settings are not fine-tuned to be appropriate for each player.

In recent years, researchers have developed advanced techniques for difficulty scaling of games. Demasi and Cruz [21] used coevolutionary algorithms to train game characters that best fit the challenge level of a human

player. Hunicke and Chapman [22] explored difficulty scaling by controlling the game environment (i.e., controlling the number of weapons and power-ups available to a player). Spronck *et al.* [19] investigated three methods to adapt the difficulty of a game by adjusting automatically weights assigned to possible game strategies. In related work, Yannakakis and Hallam [23] provided a qualitative and quantitative method for measuring player entertainment in real time.

The proposed mission and space grammars combined with player models offer a straightforward means to difficulty scaling. Generally, we may prefer to use the grammars for the purpose of generating the most challenging game environment possible. Analogously, the grammars can be applied for the purpose of obtaining (and maintaining) a predefined target in the provided challenge. For instance, the grammars could exploit information of the established player models, to generate dynamically an easily maneuverable environment with appropriately few powerful opponents, instead of a complex environment with frustratingly many powerful opponents.

CONCLUSIONS

The levels of action adventure games, and numerous other games, are double structures consisting of both a space and a mission. When generating levels for this genre procedurally, it is best to break down the generation process in two steps. Generative graph grammars are suited to generate missions. They are capable of generating non-linear structures which for games of exploration are preferred over linear structures. At the same time they can also capture the larger structures required for a well-formed game experience. Once a mission is generated an extended form of shape grammar can be used to grow a space that can accommodate the generated mission. This requires some modifications to the common implementation of shape grammars. The most important modification is the association of a rule in the shape grammar with a terminal symbol in the grammar used to generate the mission.

Breaking down the process into these two steps allows us to capitalize on the strengths of each type of grammar. With a well-designed set of rules and the clever use of recursion, this method can be employed to generate interesting and varied levels that are fun to explore and offer a complete experience. Furthermore, these techniques can be used to generate levels on the fly, allowing the game to respond to the player behaviour. By means of establishing and exploiting player models, generative grammars can be used to scale dynamically the difficulty level to the player, and adapt the game space and game mission online, while the game is still being played, to ensure the game experience is tailored to the individual player. This opens up opportunities for gaming and interactive storytelling that hitherto have hardly been examined.

REFERENCES

[1] Pedersen, C., Togelius, J., Yannakakis, G.: Modeling Player Experience for Content Creation. IEEE Transactions on Computational Intelligence and AI in Games **1**(2) (2009) 121–133

[2] Compton, K., Mateas, M.: Procedural level design for platform games. In: Proceedings of the AIIDE. (2006)

[3] Dormans, J.: Adventures in level design: generating missions and spaces for action adventure games. In: Proc. of the 2010 Works. on Proc. Content Generation in Games. (2010) 1–8

[4] Vogler, C.: The writer's journey: Mythic structure for writers. Michael Wiese Productions (1998)

[5] Ashmore, C., Nitsche, M.: The Quest in a Generated World. In: Proc. 2007 DiGRA Conference: Situated Play. (2007) 503–509

[6] Golding, J.: Building blocks: Artist driven procedural buildings (2010) Presentation at GDC 10, San Fran., CA.

[7] Parish, Y., Müller, P.: Procedural modeling of cities. In: Proceedings of the 28th annual conference on Computer graphics and interactive techniques. (2001) 301–308

[8] Adams, D.: Automatic Generation of Dungeons for Computer Games. University of Sheffield Dissertation (2002)

[9] Stiny, G., Gips, J.: Shape grammars and the generative specification of painting and sculpture. Information processing **71** (1972) 1460–1465

[10] Carmel, D., Markovitch, S.: Learning models of opponent's strategy in game playing. In: Proceedings of AAAI Fall Symposium on Games: Planning and Learning. (1993) 140–147

[11] Van den Herik, H.J., Donkers, H.H.L.M., Spronck, P.H.M.: Opponent modelling and commercial games. In: Proceedings of the CIG'05. (2005) 15–25

[12] Fürnkranz, J.: Recent advances in machine learning and game playing. ÖGAI-Journal **26**(2) (2007) 19–28

[13] Houlette, R.: Player modeling for adaptive games. In: AI Game Programming Wisdom 2. (2004) 557–566

[14] Millington, I.: Artificial Intelligence for Games. Morgan Kaufmann Publishers, San Francisco, California, USA (2006)

[15] Donkers, H.H.L.M., Spronck, P.H.M.: Preference-based player modeling. In Rabin, S., ed.: AI Game Programming Wisdom 3. (2006) 647–659

[16] Denzinger, J., Hamdan, J.: Improving modeling of other agents using tentative stereotypes and compactification of observations. In: Proceedings of the IAT 2004. (2004) 106–112

[17] Lankveld, G., Schreurs, S., Spronck, P.: Psychologically verified player modelling. In: Proceedings of the GAMEON'2009. (2009) 12–19

[18] Ryan, M.: Narrative as virtual reality: Immersion and interactivity in literature and electronic media. Johns Hopkins University Press Baltimore, MD, USA (2001)

[19] Spronck, P.H.M., Sprinkhuizen-Kuyper, I.G., Postma, E.O.: Difficulty scaling of game AI. In: Proceedings of the GAMEON'2004. (2004) 33–37

[20] Olesen, J.K., Yannakakis, G.N., Hallam, J.: Real-time challenge balance in an RTS game using rtNEAT. In: Proceedings of the CIG'08. (2008) 87–94

[21] Demasi, P., Cruz, A.J.de.O.: Online coevolution for action games. International Journal of Intelligent Games and Simulation **2**(3) (2002) 80–88

[22] Hunicke, R., Chapman, V.: AI for dynamic difficulty adjustment in games. In: AAAI Workshop on Challenges in Game Artificial Intelligence. (2004) 91–96

[23] Yannakakis, G.N., Hallam, J.: Towards optimizing entertainment in computer games. Applied Artificial Intelligence **21**(10) (2007) 933–971

REAL-TIME LOAD BALANCING OF AN INTERACTIVE MULTIPLAYER GAME SERVER

James Munro and Patrick Dickinson
Lincoln School of Computer Science
University of Lincoln
Brayford Pool
Lincoln, LN6 7TS
United Kingdom
{jmunro,pdickinson}@lincoln.ac.uk

KEYWORDS

Game Architecture, Concurrent Programming, Multiplayer Games, Scalability.

ABSTRACT

We investigate optimal load balancing strategies for a scalable parallel game server architecture. Our work develops from an existing multi-threaded implementation of the QuakeWorld game server: we investigate the comparative efficiency of different load-balancing algorithms, and determine how different metrics can be used to analyse performance. We find that achieving optimal performance is a trade-off between achieving an even workload distribution whilst minimising intra-frame wait time, and that a combined set of metrics is required to understand how load balancing affects performance.

INTRODUCTION AND EXISTING WORK

The introduction of multi-core CPUs has lead to interest in concurrent processing architectures for video game engines and servers. In this paper we consider the optimisation of load balancing for a parallel game server which scales to a multi-core CPU. Some existing work has been conducted in this area; however, the issue of load balancing has not been examined in detail. Moreover, parallel server architectures are inherently complex, and existing research has not considered how performance is best quantified: we will show that a set of relevant metrics are required to fully analyse algorithm performance. These represent our main contributions, together with a re-evaluation of results presented by existing research.

The Quake series of games engines has often been used as a test-bed for work on parallel game architectures, due to its commercial success and availability as an open-source project. Abdelkhalek et al. (2003) began a study of the QuakeWorld server, which developed into a multi-threaded implementation using a lock-based approach to synchronise access to shared resources (Abdelkhalek and Bilas, 2004). The study was successful in increasing the maximum number of clients that can simultaneously interact in the game world.

Cordeiro et al. (2007) introduced a lock-less strategy which avoids manual synchronisation. It is this approach that forms the basis of our own study, and is described in more detail later in this paper. Alternative approaches employ multiple computers to increase capacity (Müller et al., 2007; Ploss et al., 2008; Cronin et al., 2002; Alexandre et al., 2009). These works address the same issue of synchronised access to data, with the additional complexity of sharing data across a network. More recent work has investigated the use of transactional memory to abstract the complexity of synchronisation (Zyulkyarov et al., 2009; Gajinov et al., 2009; Lupei et al., 2010). However, results show that software-based implementations are not yet capable of supporting real-world game servers.

STARTING POINT FOR OUR WORK

Our work develops from the QuakeWorld server architecture proposed by Cordeiro et al. (2007): their design makes use of the *clone()* system call within the Linux kernel. This creates a new execution thread (forced to run on a discrete CPU core) that exploits copy-on-write (COW) semantics, reducing the complexity of thread synchronisation across shared data structures (Daniel P. Bovet, 2006). There is no need for application-level mutual exclusions, and the system can run in a lock-less fashion.

After incoming packets have been received, the server performs a pre-processing step which creates “connected” groups of entities (entities which *may possibly* interact in the current frame) based on a radius of 256 game units. Distinct groups are physically separated in the current frame, and so may be processed in parallel. These groups are queued for execution across the available threads, based on a Longest Processing Time First (LPTF) scheduling algorithm (Coffman and Sethi, 1976) which is intended to produce an even load balance. At this point the server invokes the *clone()* system call, and supplementary threads are launched to execute their allotted groups.

Each thread processes its allotted entities, and selectively synchronises data into a shared structure. Any modified global or shared data is discarded, and not propagated back into the coordinator thread unless manually specified. The main coordinator thread *must wait* at this point for the execution of the supplementary threads before beginning the next

frame. After synchronisation, the supplementary threads are destroyed in parallel, while the coordinator thread completes execution of the frame.

LOAD BALANCING ALGORITHMS

Our initial interest is in determining an optimal load balancing algorithm to support the architecture proposed by Cordeiro et al. (2007). To this end we consider a set of candidate algorithms which are analysed in our experimental work presented later in this paper. In this section we proceed to describe a set of candidate algorithms and their anticipated behaviour and performance.

Longest Processing Time First (LPTF) takes the pre-processed list of entity workgroups and sorts them in order of descending size. For each group, it identifies the thread with the least *weight* (defined as the number of entities within a group) assigned to it and then allocates it the largest group before moving onto the next. This is repeated until all workgroups have been assigned to a thread. In Cordeiro *et al.*'s implementation, the main coordinator thread is given an initial *weight penalty* of 1 which results in less weight being subsequently assigned to it. This is justified as an offset against additional tasks that the main thread must perform. Part of our experimental work investigates whether this decision actually provides an optimal balance.

Our initial analysis suggests that LPTF produces an approximately even workload distribution. However, a significant imbalance can occur on the second thread, caused by a large congregation of clients that cannot be split into smaller groups. This large group is typically assigned to the second thread because of the penalty given to the coordinator thread.

Shortest Processing Time First (SPTF) is essentially the opposite of LPTF. It takes the list of workgroups and sorts them in order of ascending size. For each group, it identifies the thread with the least weight, and assigns the smallest group in the list. For consistency we carry over the penalty given to the coordinator thread in our implementation. We expected that this algorithm would perform worse than LPTF, but provide a reasonable base line performance.

Round-Robin (RR) is the simplest of the balancing algorithms we have considered. It does not perform sorting and simply iterates the list of workgroups and assigns each to an alternating thread. We anticipated that this algorithm would exhibit "worst case" performance, due to the resulting unpredictable distribution; however, we are interested in to what extent the reduced algorithm complexity offsets this.

Sorted Round-Robin (SRR) is an extension of the RR algorithm. It sorts the workgroup list in order of descending size as with LPTF, and then iterates through the list. We were unsure of how well this algorithm would perform in the QuakeWorld server.

Evaluation Metrics

The previous work by Cordeiro et al. (2007) relies on only one or two metrics to quantify server performance. One of our premises is that, in fact, a range of metrics are needed to fully understand the behaviour of a concurrent architecture. In our experimental work we use four separate metrics, and re-visit the results presented by Cordeiro *et al*. We proceed to describe our metrics.

Frames-per-second (FPS) provides us with a general perception of performance. FPS is calculated by dividing the number of executed frames by the total elapsed run-time.

Server throughput is measured in response-packets-per-second (RPP/s). This is a measure of the server's ability to respond to client requests in a timely fashion. Ideal server performance would see a 1:1 ratio with incoming request-packets-per-second. A decrease in server throughput as more clients join the server indicates that the server has become saturated and gameplay performance is degrading.

Intra-frame wait time (IFWT) is a measure of how long the main server thread spends waiting for the supporting threads to complete their tasks. As IFWT increases, server performance decreases as the load balancing algorithm is performing sub-optimally. We also expect that the less time spent waiting increases the overall time spent executing in parallel.

Workload distribution provides us with a visual representation of how evenly the load balancing algorithm distributes work between the available threads. We would expect a more even distribution to produce better overall server performance, as this would increase the contribution of each of the supplementary threads. This metric is based on the total accumulated entity workgroup weight processed by each thread for the lifetime of its execution,

EXPERIMENTS

For our experiments we used seven networked Dell XPS 630i computers. Six of the machines ran Windows 7, and acted as QuakeWorld clients; the remaining machine ran Ubuntu Linux 9.10 32-bit edition (kernel 2.6.31). Initial experiments showed that our network bandwidth was not sufficient to operate the server at full capacity, and so the computational complexity of processing a single client was artificially increased. We achieved this by adding a fixed delay (300μs) within the entity processing code: this simulates an increased processing load per player, thus reducing the total client count at which the server becomes saturated. This setup maintains an accurate representation of the effects of load balancing, because the artificial delay is distributed across the available threads (on discrete processor cores) proportionally. We also used the automated bot clients implemented by Cordeiro *et al.* with minor modifications to execute it on our Windows clients. Further details are given in Cordeiro et al. (2007). As a result of the low processing requirements of the QuakeWorld client we are able to run up to 32 clients

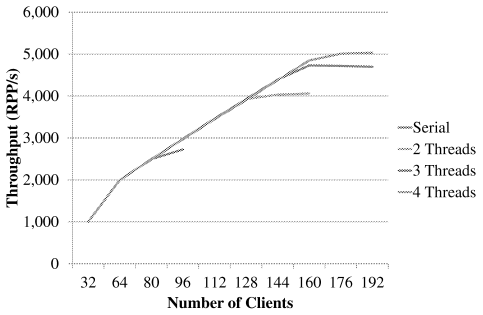


Figure 1: Server Throughput for LPTF with 1-4 Threads

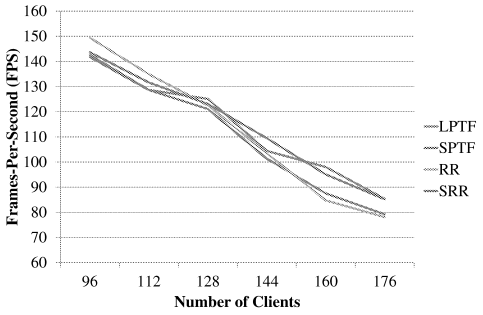


Figure 2: FPS for all Algorithms Across all Client Counts

per-machine.

For each of the load balancing algorithms we analysed the server performance for a range of different client counts. We then used the resulting data, in conjunction with our set of metrics, to compare algorithm performance and identify how many simultaneously connected clients each algorithm could support. For each algorithm we started with 96 clients, and increased this in increments of 32 to a total of 192. We performed five experiments for each specific client count, such that the results for each algorithm comprise 35 individual experiments. We repeated this for for two, three, and four CPU cores.

Results

Initially we examine the benefit of the parallel server over the serial implementation. In Figure 1 we present the server throughput for the LPTF algorithm with 2-4 threads. The serial server peaks at around 96 clients. It is also clear that using multiple threads increases the maximum capacity of the server up-to and including 4 threads. Interestingly, Cordeiro *et al.*'s analysis, based only on an FPS metric, suggests that using more than 3 threads does not improve performance. However, our analysis of server throughput shows that this is not the case, underlining the need for multiple evaluation metrics. For brevity we only present results for the 4-threaded version of the server from this point onwards.

FPS

Figure 2 shows that the measured FPS degrades as the connected client count increases. However, all algorithms (including SRR) exhibit very similar performance. In accordance with our expectations, the SPTF and RR algorithms do not perform as well as LPTF; however, their impact on performance in terms of FPS is minimal. This result, combined with the following results for other metrics, suggest that FPS is not a highly discriminative measure of performance.

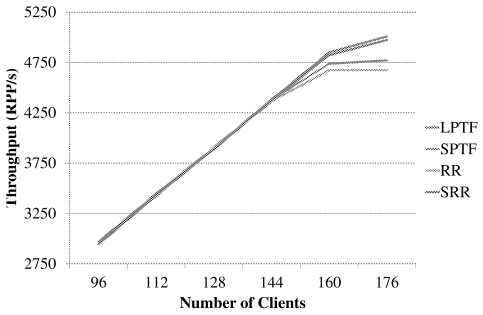


Figure 3: Server throughput for all 4 algorithms

Server Throughput

As previously discussed, server throughput is an indication of the overall performance of the server from the end-user's perspective. At the point where the server is no longer able to respond to incoming client requests, gameplay will noticeably deteriorate. In Figure 3 we can see this measurement for the four tested algorithms. As expected the LPTF algorithm causes the server to saturate at a higher point than the other algorithms. SPTF and RR do not perform as well and the server performance peaks at a lower client count. The interesting result here is that SRR performs only slightly worse than LPTF despite having a very different overall workload distribution properties (see following section). A better understanding of these performance similarities requires a more detailed analysis of the algorithms, on a per-frame basis.

Workload Distribution

Figure 4 shows the distribution for the original LPTF algorithm. These results correlate directly with results from the study by Cordeiro *et al.* Due to the penalty given to the coordinator thread, the second thread receives the greatest accumulation of weight. Figures 4 and 5 clearly demonstrate that the accumulated workload distribution is not a reliable measure of algorithm performance. The overall balance appears to be better than that achieved by LPTF, yet the performance in terms of FPS throughput and server throughput seen in Figures 3 and 2 undermine this. Based on this measure alone we would expect this to be the best performing algorithm. A similar balance is evident in Figure 6 for the RR algorithm which exhibits similar overall performance characteristics to SPTF.

The most varied workload distribution was produced by the SRR algorithm, seen in Figure 7. Based on expected workload distribution, we anticipated a poor performance; however, in terms of server throughput seen in Figure 3 this is not the case. An analysis of the the IFWT data is required to understand why this is the case.

Our results in this section show that the workload distribution metric introduced by Cordeiro et al. (2007) is not an accurate

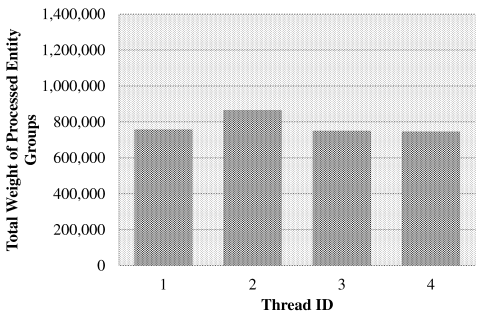


Figure 4: Workload Distribution for LPTF with 176 Clients

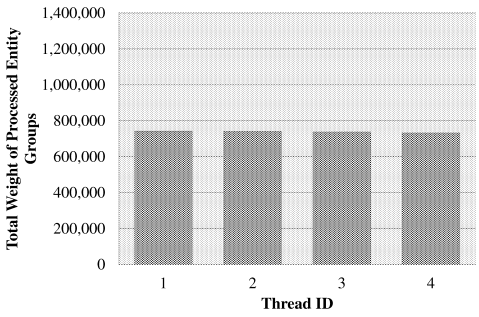


Figure 5: Workload Distribution for SPTF with 176 Clients

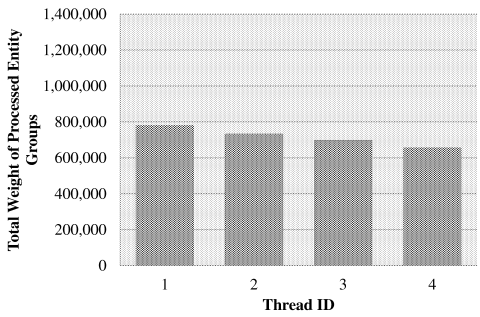


Figure 6: Workload Distribution for RR with 176 Clients

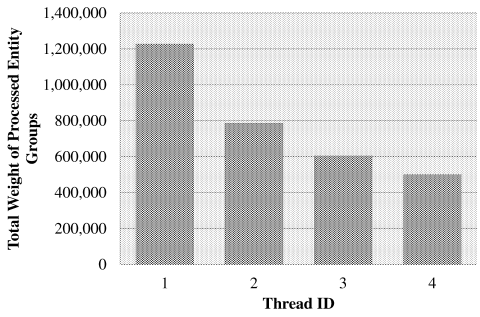


Figure 7: Workload Distribution for SRR with 176 Clients

representation of the true load balancing performance: this metric is based on the total accumulated weight distributed to each of the server's threads.

IFWT

We predicted that as the number of connected clients increased, IFWT would increase for all algorithms. We can see from Figure 8 that this is generally the case, with SRR being a glaring exception. Unexpectedly the IFWT actually decreases for each incremental client count. We can attribute this to the large workgroups caused by congregations of clients: as the number of clients increases, so does the likelihood of congregations occurring. In the SRR algorithm this large group will always be assigned to the coordinator thread. In Figure 7 we showed that the workload distribution is ladder-like in appearance. As the congregation of clients increases, the time the coordinator thread spends waiting decreases at the expense of reduced workload distributed to supplementary threads.

Per-Frame Analysis

We needed to perform further experiments to expose the true behaviour of each algorithm. We collected the raw entity workgroup data before any load balancing had been applied once-per-second for 3 minutes: this provided us with 180 samples of data. We then ran offline experiments on the data using the LPTF, SPTF and SRR algorithms. Initially, we analysed the average workload distribution for all algorithms. The results correlated with the workload distribution data presented in Figures 4, 5 and 7 and did not identify any clear differences. We have omitted the chart for space reasons.

In order to understand whether each algorithm consistently produced the same workload distribution we need to consider the standard deviation, seen in Figure 9. The deviation for SPTF is the highest across all threads and provides us with an explanation as to why total accumulated and average workload distribution are not a true measure of the algorithms performance. We conclude that SPTF produces a varied workload distribution and does not consistently produce similar results. LPTF is the most consistent algorithm which reinforces the highest server throughput seen in Figure 3.

CONCLUSIONS

Our first objective was to determine an optimal load balancing strategy for a parallel game server. Our experiments have shown that looking at the total accumulated workload distribution is not an accurate representation of the frame-by-frame performance of each algorithm. Using a range of metrics we have determined that consistently producing the same balance on a per-frame basis is a critical factor. In addition, we have shown that a set of metrics is required to achieve a detailed understanding of the server behaviour, and that previous works in this area have failed to fully appreciate this.

It remains unclear how the performance of the server will be effected once the IFWT reaches zero whilst using the SRR algorithm. We conjecture that the performance would begin to degrade, due to an inefficient workload distribution across the supplementary threads, reducing the overall time that the server spends in concurrent execution. Due to the increasing IFWT whilst using the LPTF algorithm, we have also shown that imposing a penalty on the coordinator thread is an inefficient strategy. Due to large congregations of clients the second thread has to deal with an especially large task which

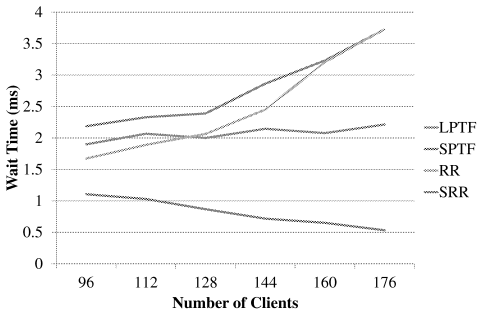


Figure 8: Intra-Frame Wait Time for all Algorithms

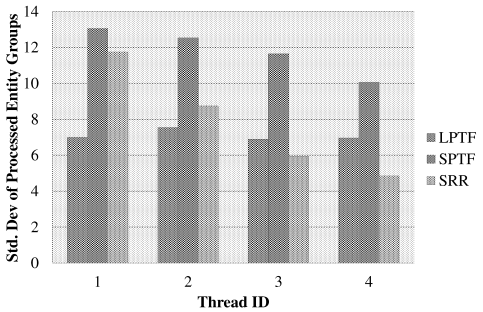


Figure 9: Standard Deviation of Workload Distribution for LPTF, SPTF and RR over 180 Frames

results in the coordinator thread spending more time waiting for the supplementary threads to finish.

Earlier in this paper we noted our intention to determine whether omitting the sorting step of the algorithms could provide a boost in performance. Our results clearly show that the RR algorithm exhibits much poorer performance than SRR as a result of skipping this step. From this we can conclude that sorting the list of workgroups by weight is a critical factor for producing a more consistent workload distribution. It still may be possible to bypass this step in situations where the list is already partially sorted, but this is not the case with the QuakeWorld server.

FUTURE WORK

We suggest that further research into the performance of the server on machines with more native cores would provide interesting results. Additionally, we propose that adjusting the way in which workgroups are constructed may provide further insight into how load balancing strategies may transpose to different game types. Using a smaller radius of effect (such as 128 units) may produce more manageable workgroups at the cost of altered game semantics.

Finally, we feel that this version of the server may be a good candidate for experimenting with a “task stealing” strategy. Currently, when a thread completes its designated workload it synchronises its processed data, and is no longer participating in parallel execution. It may be possible for an additional step to be added in which the thread re-examines the list of workgroups and ’steals’ work from the tail of another thread.

ACKNOWLEDGEMENTS

The authors of this paper would like to thank Daniel Cordeiro and Ahmed Abdelkhalek for kindly sharing the source-code of their QuakeWorld research projects.

REFERENCES

Abdelkhalek, A., Bilas, A., 2004. Parallelization and Performance of Interactive Multiplayer Game Servers. *Parallel and Distributed Processing Symposium, International* 1, 72a.

Abdelkhalek, A., Bilas, A., Moshovos, A., 2003. Behavior and Performance of Interactive Multi-Player Game Servers. *Cluster Computing* 6 (4), 355–366.

Alexandre, R., Prata, P., Gomes, A., 2009. A Grid Infrastructure for Online Games. In: *ICIS ’09: Proceedings of the 2nd International Conference on Interaction Sciences*. ACM, New York, NY, USA, pp. 670–673.

Coffman, Jr., E. G., Sethi, R., 1976. A Generalized Bound on LPT Sequencing. In: *SIGMETRICS ’76: Proceedings of the 1976 ACM SIGMETRICS conference on Computer performance modeling measurement and evaluation*. ACM, New York, NY, USA, pp. 306–310.

Cordeiro, D., Goldman, A., da Silva, D., 2007. Load Balancing on an Interactive Multiplayer Game Server. In: Kermarrec, A.-M., Boug, L., Priol, T. (Eds.), *Euro-Par 2007 Parallel Processing*. Vol. 4641 of Lecture Notes in Computer Science. Springer Berlin / Heidelberg, pp. 184–194.

Cronin, E., Filstrup, B., Kurc, A. R., Jamin, S., 2002. An Efficient Synchronization Mechanism for Mirrored Game Architectures. In: *NetGames ’02: Proceedings of the 1st workshop on Network and system support for games*. ACM, New York, NY, USA, pp. 67–73.

Daniel P. Bovet, M. C., 2006. Understanding the Linux Kernel, third edition Edition. O’Reilly, pages 115–118.

Gajinov, V., Zyulkyarov, F., Unsal, O. S., Cristal, A., Ayguade, E., Harris, T., Valero, M., 2009. QuakeTM: Parallelizing a Complex Sequential Application Using Transactional Memory. In: *ICS ’09: Proceedings of the 23rd international conference on Supercomputing*. ACM, New York, NY, USA, pp. 126–135.

Lupei, D., Simion, B., Pinto, D., Misler, M., Burcea, M., Krick, W., Amza, C., 2010. Transactional Memory Support for Scalable and Transparent Parallelization of Multiplayer Games. In: *EuroSys ’10: Proceedings of the 5th European conference on Computer systems*. ACM, New York, NY, USA, pp. 41–54.

Müller, J., Gorlatch, S., Schröter, T., Fischer, S., 2007. Scaling Multiplayer Online Games Using Proxy-Server Replication: a Case Study of Quake 2. In: *HPDC ’07: Proceedings of the 16th international symposium on High performance distributed computing*. ACM, New York, NY, USA, pp. 219–220.

Ploss, A., Wichmann, S., Glinka, F., Gorlatch, S., 2008. From a Single to Multi-Server Online Game: a Quake 3 Case Study using RTF. In: *ACE ’08: Proceedings of the 2008 International Conference on Advances in Computer Entertainment Technology*. ACM, New York, NY, USA, pp. 83–90.

Zyulkyarov, F., Gajinov, V., Unsal, O. S., Cristal, A., Ayguadé, E., Harris, T., Valero, M., 2009. Atomic Quake: Using Transactional Memory in an Interactive Multiplayer Game Server. In: *PPoPP ’09: Proceedings of the 14th ACM SIGPLAN symposium on Principles and practice of parallel programming*. ACM, New York, NY, USA, pp. 25–34.

NARRATIVE MEMORY IN HYPERFICTION AND GAMES

Helena Barbas

IEMo and DEP/FCSH and CENTRIA
Universidade Nova de Lisboa, Portugal,
e-mail: {hebarbas}@fcsh.unl.pt

KEYWORDS

Storytelling; narratology; memory; hyperfiction; serious games; digital humanities;

ABSTRACT

The aim of this paper is to address some theoretical issues concerning the narrative practice in cyberspace. From a narratological perspective it intends to clarify the functioning of time and space in storytelling. For that purpose it traces the concept(s) of memory inherited from rhetoric; the use of memory as a narrative device in traditional accounts; the adaptations imposed by hyperfiction. Using practical examples (including two Portuguese case studies - *InStory* 2006, and *Noon* 2007) it will show how narrative memory strategies can be helpful in game literacy. The main purpose is to contribute to serious game research and (trans)literary studies.

INTRODUCTION

The narrative-memory relationship is bidirectional. Memory is the precondition for engendering a narrative as well as narrative is crucial for the agility of the faculty of memory (Rose 2003). All narratives are inherently put together through the use of recollection: «As already observed, there is no such thing as memory of the present while present, for the present is object only of perception, and the future, of expectation, but the object of memory is the past. All memory, therefore, implies a time elapsed» says Aristotle *On Memory and Reminiscence* (449b24).

Narration is intrinsic to the human condition. We do tell ourselves our everyday stories, daily fabulating our lives - transforming the events experienced into our own personal novel. From a psychoanalytical point of view, all (auto)biographies are self told stories (Robert 1972).

The present concepts of memory are inherited from Antiquity, namely the ones registered in the *Rhetoric Ad Herennium* (Caplan 1964). In Classical Rhetoric Memory is a discipline and an art. The system it uses - the «method of *loci* [places]» - has been overestimated or under-rated. The renaissance re-interpretations of *ad Herennium* transform the «*loci* method» either in a encyclopedic scheme to arrive to the knowledge of God; or in a mere technique for learning «by rote» poems, lists of events, names. The modern contemporary approaches are interested in its architectural patterns or in its visual splendour that allows to play with images and text. None explored its inherent narrative features.

Presently, the notion of memory is recovering its theatrical dimension, and the notional frame of the concept has been enlarged to a never imagined scope in the mediatised sphere. The «method of *loci*» can become useful if up-dated with the new proposals from the cognitive sciences, and Digital Games Based Learning (DGBL) lessons (Blunt 2005). The narrative-memory relationship becomes fundamental when research proves that people do remember better when content is furnished through storytelling (Furman 2007) - or a gameplay (Blunt 2010).

A SHORT HISTORY OF MEMORY

In Hesiod's (c.700 b.C.) *Theogony* (vv. 53-62) Mnemosyne was the personification of Memory. She was a titaness, daughter of Gaia and Uranus. During nine nights Zeus fathered in her the nine Muses - all twins and born at the same time (Evelyn-White 1914). These muses were multitasking and globalized - inspiring all men and all possible kinds of human activities.

From Antiquity through the Middle Ages, Memory was a major discipline of Rhetoric. Had as its first syllabus the *Rhetorica ad Herennium* (c. 86-82 b.C.) falsely attributed to Cicero (106-43 b.C). Its four books on the «theory of public speaking», an aide to lawyers and orators (in an illiterate society), were the source for all posterior treatises antique and medieval, up to our modern «5 point essay».

Memory as an «Art»

The invention of the «Art of Memory» was attributed to the poet Simonides of Ceos (556-486 b. C.), the first to use the association of images with mental places (*loci*) as a mnemonic device (Edmonds 1924). Its initial structure and principles are registered for the first time in *ad Herennium* (*Book* III, 16-24).

By the «method of *loci*» the practitioner devises the matter (a topic, facts, a narration); chooses an architectural layout of some physical space (his house, a building, a desert - or his own body) and distributes his information/images by those discrete *loci*. When desiring to remember, he can mentally walk through these *loci*, and reclaim the items as necessary.

This art attained its excellence with Giulio Camillo (ca. 1480–1544) who, in his book *L'Idea del Teatro* (inspired by Vitruvius *De Architectura*, 25 b.C.), transformed the whole cosmos into a huge mythological *locus*, from where all the knowledge could be retrieved: «this high and incomparable distribution not only keeps hold of the entrusted things, words and arts, but also shelters the information for our own needs so that we can easily find it; and so, it will even provide true wisdom» (Camillo 1554). This was adapted by several scholars to their disciplines - Ramon Lull (1232-1315), Leon Batista Alberti (1404-1472), Paolo Lomazzo

(1538-1600), Giordano Bruno (1548-1600), Tomaso Campanella (1568-1639) - each of them contributing to enrich the several steps of the process. Ignatius Loyola (1491-1556) made it the source of his catholic *Spiritual Exercises* (1521-3), and it became fundamental to the Jesuitical pedagogy model (i.e., Claudio Acquaviva's *Ratio Studiorum*, 1599).

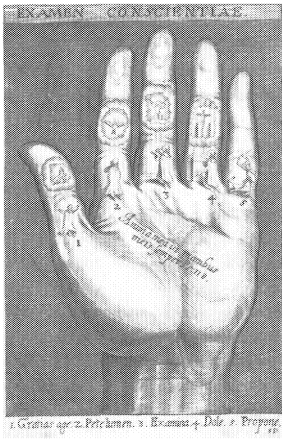


Figure 1. Engraving of a Jesuitical mnemonic for the daily conscience exam (c.1588).

At the same time, this scholastic form of knowledge is associated with less orthodox subjects such as Astrology, Alchemy and Kabala. The image engraved for Robert Fludd's (1574-1637) "Ars memoriae" (1619), is said to have inspired the project of Shakespeare's Globe Theatre.

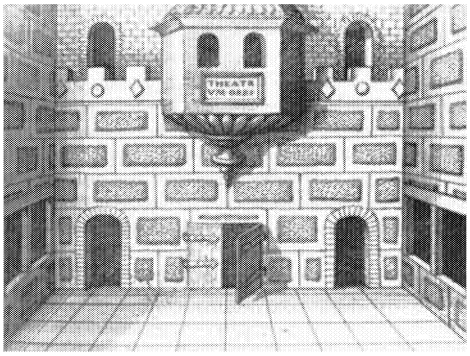


Figure 2. Robert Fludd - *Theatrum Orbi*, "Ars memoriae", engr. John Theodore De Bry, 1619;

Besides architects and philosophers, the «Art of Memory» was also practiced by painters and sculptors. Its very last rules are encoded in Cesare Ripa's *Iconologia* (Ripa 1593). These figures were used as ready-made «*topoi*»/images to express emotions, or concepts:

and were accompanied by a caption:

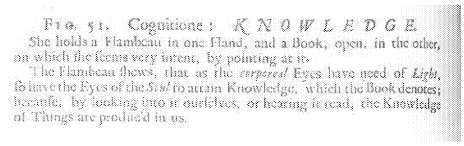


FIG. 51. Cognitione: KNOWLEDGE.
She holds a Flambeau in one Hand, and a Book, open, in the other, on which she seems very intent, by pointing at it.
The Flambeau shews, that as the *corporeal* Eyes have need of *Light*, so have the Eyes of the *Soul* to attain Knowledge, which the Book denotes; because, by looking into it ourselves, or hearing it read, the Knowledge of Things are produc'd in us.

Figure 4. Cesare Ripa - *Knowledge*-1593;

The «art of memory» in cyberspace

The historical studies of Paolo Rossi (1960) and Frances Yates (1966) were responsible for a renewed interest in the rhetorical Art of Memory. The theme, plus the visual impact and architectural potential of the memory theatres were most propitious to seduce cyber artists and theorists.

The «Simonides Effect» was explored at M.I.T.: «a study recalling the ancient principle of using spatial cueing as an aid to performance and memory» (Bolt 1979); computers were addressed as "Theatre" (Laurel 1991); Camillo's theories fundament the project «Computers as Theatre of Memory» (Matussek 1999-2004). More recently the account of the evolution of these practices is given concerning contemporary cyber performances (La Rocca 2008).

The above approaches are focused either on understanding architectural patterns, using the «Art» as a model for data mining (in its prosthetic dimension); or on its optical magnificence, playing with images and text. Although it is recognized that these “architectural mnemonics” are based on the use of narrative structures for the improvement of the ability to memorise (Sapir 2006), and that (again from an architectural perspective) the 'images' to be memorized can be events (Yilmaz 2010), none explores its intrinsic narrative quality.

MEMORY IMAGES AND WORDS AS NARRATIVES

It is said in *Ad Herennium* (III, 20-34): «Since, then, images must resemble objects, we ought ourselves to choose from all objects likenesses for our use. [...] Likenesses of matter are formed when we enlist images that present a general view of the matter with which we are dealing; likenesses of words are established when the record of each single noun or appellative is kept by an image. Often we encompass the record of an entire matter by one notation, a single image.» (Caplan 1954). The «subject-matter» has «narration» for synonym (*A.H.* I,8). Images or words, attributed to each *locus*/place correspond to an idea or an event - a 'historia' as proposed by Alberti (Spencer 1970): a narrative sequence to be individually organized within a scenario, in order to re-build a larger story/discourse.

In cyberspace information is not just stored but produced. Words and images are mnemonic devices, stimuli, and traces of events. New media (Aczél 2010): «necessarily introduces a grammar upon which a writing becomes the self (mystory), a new function of creation and storage (algorithms and databases), an alternative to the narrative (periautography, data-autography) and spaces where all other real places may intersect in ‘another time’, heterochrony».

Independently of the new divisions and classifications proposed for memory - subjective, individual, public, collective (Rose 2003); sensorial, short or long-term (Rossi 2000); relying on new technologies and practices (Van

House and Churchill 2008) resorting to a globalized database, shared worldwide (MySpace, Facebook, Blogs, Twitter), all and each of these exercises have to be referred to what has just happened - Aristotle (384-322b.C.) is still right: all memories are of things past.

«New» concepts of Narrative

The ambiguity of the concept narrative has already been discussed (Barbas and Correia 2009). Here - for future purposes - the opposition between «story» (content plane) and «discourse» (expression plane) will be summoned, as theorized by the French structuralists (Genette 1996) and linguists (Benveniste 1974); it was also explored by the Russian formalists (Todorov 1983), American scholars (Prince 1988) and game theorists (Crawford 2004, Montfort 2007).

The similarity between *Ad Herennium* (III, 20-34) and the definition proposed by Marie Laure Ryan is very interesting: «Narrative is defined as a mental image, or cognitive construct, which can be activated by various types of signs. This image consists of a world (setting) populated by intelligent agents (characters). These agents participate in actions and happenings (events, plot), which cause global changes in the narrative world.» (Ryan 1994).

TIME AND SPACE DETERMINANTS

Memory and narrative rest on two vectors - time and space. Causal and chronological dimensions are essential to all kinds of fiction. A narrative corresponds to the representation of at least one event; each event is given under the form of two propositions (sentences). The story results from the relationship established between those two sentences (by contiguity and consecution).

The temporal junction is crucial, because a shift in the order of the sentences can modify or falsify the meaning of the story (i.e.: «The baby cried. The father took the baby in his arms» vs. «The father took the baby in his arms. The baby cried.»). For this reason, in spite of some oppositions (Eskelinen 2008), order is quite relevant. With another example, Nick Montfort states that: «changing the order in which events in a given temporal sequence are related — is important to the aesthetic and rhetorical effect of more complex narratives and to ones of more literary interest.» (Montfort 2007). Yet, this goes beyond the mere literary aesthetics: it is close to the *cataphora* with its binding effect that the old rhetoric masters knew so well; it is related to the way our brain accepts (any kind of) information - the last data received being seen as the most important, and giving the tone to the previous discourse (retroaction effect); it results from a cultural practice in which all logical discourse formulations (rhetorical, philosophical, mathematical) present their conclusions/ demonstrations in the last line.

Any association of ideas or of events, besides the chronological factor, needs some kind of physical support/space. In the last instance it resorts to the adverbs «here» and «now». These are deictic (empty) words, because although their semantic meaning is fixed, their denotation varies accordingly to the moment and/or location of the enunciation. And time and space are the two fundamental categories we use to structure the world around us - being it real, imaginary or virtual.

Transposing these principles to the world of fiction, and considering memory in its first variant as history, it is necessary to differentiate the several layers operating extra and intra-textually. The distinction between «textual time and text time may be one of the crucial differences between spatial hypertext and cybertext, since the latter often aims at conflating these into one temporal dimension» (Kosima 2009). Narratological studies can still offer the means to discriminate several of the time/memory planes.

NARRATIVE EXTRINSIC TIME - READING

Although the textual worlds/settings may get fixed or objectivised through the use of formalization, their meaning is never autonomous, as signification depends on the reader/user experience - in real time and space. For measuring the speed of narration from the reader perspective, Genette invented a (much criticized) hypothetical ratio between the chronology of events and the number of pages necessary to describe them. This «time of reading» by an «imaginary reader» is exterior to the story itself. It concerns the reception of all kinds of works (Jauss 1982). And it is under this scope that all the research concerning the reader/user experience should be included: interactivity, immersion, evaluation of the affective states. Even when it correlates to the 'narrative pact'/the player's suspension of disbelief, it is always external to storytelling.

Reading time in cyberworks

As an exception, there is a reading time that can be controlled from the interior of the narrative, which is particular and exclusive to cybertexts. Hypermedia artists have been playing with this process, trying to control the user's experience in various ways, restraining or delaying her reading. The text can appear on the screen only for a limited period of time, as in *Hegirascope 1* or *2* (Moulthrop 1995). If the reader doesn't open any link, the browser auto-refreshes the active window every 30 seconds. Something similar occurs in *Grammatron* (Amerika 1997), but here the aim is to play with the speed of the exhibited lexias. In *Blue Hyacinth* (Masurel and Andrews 2007) the work does have a stronger literary purpose; the text(s) change every time it/they is/are crossed (even unintentionally) by a movement of the mouse.

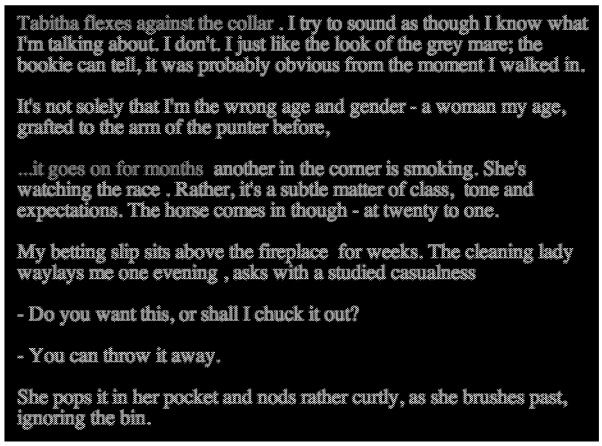


Figure 5. Pauline Masurel and Jim Andrews - *Blue Hyacinth*, 2007;

In these dynamic cybertexts the duration for each node can be pre-determined by their authors. They may intend the act of reading to always be new, as it will not be easy to return to the exact same window/story episode. This fuses real reading time with fiction time, but also destroys the performance of memory, and the possibility of recollection. Each narrative becomes a «happening» or a lonely labyrinth that cannot be shared.

NARRATIVE INTRISINC TIME(S)

Inside any account there are two levels of historical time: context and co-text (Halliday 2002). The first allows the user to engraft the narrative in her experience of the real world; it is fundamental for any historical novel, or biographies; not so imperative for psychological fictions. The latter corresponds to the historical time intrinsic to the story itself; it entails the (re)making of a fictional verisimilar microcosm that offers referents for the legibility of the plot; its limits are science-fiction experiences (or Tolkien novels - demanding a translation, a mediatisation through maps, dictionaries, encyclopaedias). Both of these spheres, corresponding to the creation of virtual worlds/scenarios (rhetorical mnemonic 'backgrounds'), are subject to verisimilitude laws, and fit within the author's responsibility.

Narratology and the use of time

Consequent from the above mentioned distinction between «story» and «discourse» the concept of «anachronies» becomes operational. This term, coined by Genette, defines the rupture of the temporal order used as a stylistic tool inside traditional fiction; it refers to the workings of history/memory inside the narrative *per se*.

Under «anachronies» the more significant are «analepsis» and «prolepsis». As we have seen (Barbas and Correia 2009), both are essential instruments for playing with time inside a narrative. The first - flashback - in order to delay the outcome of any episode, to postpone the resolution of the bifurcation, to reiterate any event already told - is the memory internal to the narrative. The latter - flash-forward - has a prophetic function that can be ominous, is the memory of the *future* inside the narrative.

In what concerns duration - «anisochronies» - they result from the hypothetical comparison between the story's chronological time and an ideal duration of the reading. This farfetched notion has been criticized and, as above mentioned, belongs to the sphere of reception.

Yet, this cannot obliterate Genette's subsequent considerations in what concerns the aspects of «duration» that affect time inside the narrative (some of them also rhetorical tropes): «ellipsis» and «parallipsis» (omission of diegetic events); «summary» (a concise digest of events); «scene» (equal time between story and discourse, equivalent to dialogues); and «pause» (the descriptions, or the Barthesian catalysis). All these are narrative strategies that can interfere with the plot, accelerate or delay the final outcome, which makes them meaningful and useful tools for storytelling.

Nevertheless, *prolepsis* and duration are print-oriented approaches that fail when applied to hyperfiction and games.

Narratives without memory

InStory (Interactive Media Group 2006). This project had the goal of implementing a platform for mobile and cinematic storytelling, information access, and gaming activities using a PDA, in «Quinta da Regaleira» (World Heritage) in Sintra, Portugal (Correia and Alves 2005). The script creation was mainly based on a linear structural narrative sequence scheme (Greimas 1996). However, as each sequence needed to be reproduced in each different spot/context, it became necessary to resort to Steve Holzman's aesthetical proposals for virtual space. This inspired the elaboration of a fractal narrative model (Holzman 1997). De par, as muses for scriptwriting were used *The Game of the Goose*, and the *Rhetorica Ad Herennium*.

The development of basic interactive content was predetermined by the environment. The user is immersed in the story - as a character/narrator/avatar - inhabits the diegesis, and can decide which way to go. The real time and space become a theatrical setting, projecting all actions to the present. *Analepsis/prolepsis* were not supported by a narrative of this kind. They could be recovered only when the user had access to her experiences recorded in the server. The user's wanderings and later recollection of the visit reproduced the procedures recommended by *Ad Herennium* for the creation of backgrounds (III,19) in a inverted order; here the *loci* were the places visited. This was a cultural heritage project (Barbas and Correia 2006), and its main objective was the conservation/promotion of a historical site in which cyberspace was used as a new alternative for a curatorial action.

In what concerns duration - «anisochronies» - all narrated episodes became «scenes» (equal time between story and discourse).

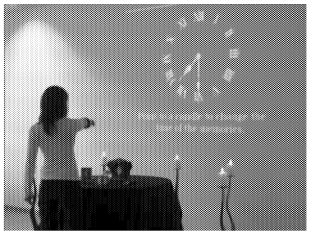
The theoretical outcome was that memory of the *future* works only inside traditional narratives. It corresponds to the existence of actions in a closed linear world (the book) that are naturally implied in and by the story – by logic or verisimilitude. Even in the case of «open narratives» (Eco 1979) there is a chronological order of events and it is possible to know where and when the story is going to finish (the last page). But not so in hyperfiction.

RESURGENCE OF MEMORY IN HYPERFICTION

Cyberart practitioners have not abdicated from the use of memory as a narrative device. They resort to it, adapting it to conform with the novel ways of storytelling, to the instruments they develop, and the stories they want to tell. Poles apart - ubiquitous computing/ ambient intelligence versus cybertext - two examples of fictional strategies that reinstate memory in the plot are shown below.

Noon - a secret told by objects (Heidecker and Martins 2007) is an interactive installation where common objects are repurposed as interfaces for telling a story. The plot is quite linear. Four members of the Novak household (father, two children, and a maid) perished in a tragic fire; Mrs. Novak survived, but had to be institutionalized. The police have no means or will to proceed with the investigation, and the case is closed. «Oddly enough, some objects were

recovered unscathed from the wreckage. It's as if Fate itself meant them to survive so they could one day tell their stories. That day is here.». The six salvaged objects are repositories of the Novak's recent past: «these hold the key to the mystery, in the form of memories».

Figures 6-7. Christina Heidecker and Tiago Martins - *Noon - A Secret told by objects,* 2007;

These physical objects are displayed on a table surrounded by five candles. The user can touch them wearing a special glove (the Gauntlet). By performing different gestures, she recalls the events previous to the arson: «Displaying of a memory is triggered when the Gauntlet detects an RFID tag, and the haptic actuator is activated to both signal a reading and suggest a flow of energy between player and object» (Heidecker and Martins 2007). The information retrieved is projected on a screen, accompanied with noises/music, so that the player «pieces together a puzzling narrative, at times actively confronting a physical manifestation of the most painful impressions in the form of poltergeist».

Although this story-game is still enacted in a dramatic present, the narrative elements are events of the past, and the plot is structured upon the existence and recovery of memories. In terms of the «Art of Memory» this project physically recreates the images supporting the «subject-matter» (*A.H.*, III, 16); and the girl with the objects (lights, tome) could be read as an update of Ripa's icon for Knowledge (Fig.4).

Dim O Gauble (Campbell 2006). This is one of the works presented in *Dreaming Methods.* The group aims at «a fusion of writing and new media exploring imaginary memories and dream-inspired states». These cyberfictions refuse labels: «to be classed as "literature" or not has ceased to be of much interest to both author(s) and reader(s)» (Campbell 2006). Each project combines all the media and one of the most interesting is *Dim O'Gauble* by Campbell himself.

Figure 8. Andy Campbell - *Dim O'Gauble*, 2006;

It is: «based on a series of the author's childhood drawings» and presents the user/reader with an intricate collection of intertwined colours and shapes, music, and several layers of perspective suggesting depth (3D).

Navigation is easy. Yellow arrows send the user to a new "scene", yet always inside the same window. There are text fragments with hyperlinks showing dates, details. Animated cut-sequences reveal additional narrative elements: «the generation of a piece may begin with one particular "scene" that is programmed/ designed/written/assembled together to a high degree, and which then acts as a gateway into a larger and more complicated narrative» (Campbell 2006). The critics/readers have tried to classify it as interactive fiction, dynamic narrative, mobile text, game, (nobody called it an 'opera'). The project has been widely acclaimed: «The text *Dim O'Gauble* is about memories, how spaces and places are so often the sites of memory. The theme of memory gives rise to the image of the failure of communication and how understanding often emerges 'after the event', when a stronger sense of (dialogic) context is available» (Barrett 2010). In rhetorical terms this could represent a not very suitable «background» (*A.H.* III, 19, 32).

The novelty in *O'Gauble* is its oniric aura, which gives it an unsuspected psychological dimension. Its reading demands the knowledge of narrative strategies, but also of several other vocabularies - painting, dramaturgy, theatre, scenarios, music. This experience also confirms the need for a new info-aesthetics (Manovich 2008).

TRANSLITERATURE, TRANSLITERACY, GAMELITERACY

Litera is the radical inspiring these new terminologies. «Transliterature» names a «new universal genre intended to unify electronic documents and media, erasing format boundaries and easing the copyright problem.» (Nelson 1960) - a genre and a software. «Transliteracy» is «the ability to read, write and interact across a range of platforms, tools and media from signing and orality through handwriting, print, TV, radio and film, to digital social networks» (Thomas 2007); the examples come from history, orality, philosophy, literature, and ethnography. «Gameliteracy» refers «one approach to addressing these new sorts of literacies that will become increasingly crucial for work, play, education, and citizenship in the coming century» (Zimmerman 2007).

Our friend from *Ad Herennium* states that: «Those who know the letters of the alphabet can thereby write out what is dictated to them and read aloud what they have written. Likewise, those who have learned mnemonics can set in backgrounds what they have heard, and from these backgrounds deliver it by memory. For the backgrounds are very much like wax tablets or papyrus, the images like letters, the arrangement and disposition of the images like the script, and the delivery is like the reading.» (III, 17). There is an apparently easy similarity in decoding words and images that has to be regained.

Humans are natural born storytellers who interlace mental tales to fit structures and shore up recollection. The research on memory (Furman 2007) attests that, although subjects may keep the meaning for short stories but tamper with the structure, people memorize information much better when it

is presented as narrative; that they do remember the content of novels, plays, and movies well. Narrative becomes essential for learning.

Conversely: «Learning to read a game system [...] points toward a specific kind of literacy connected, in part, to the ability of a player to understand how systems operate, and how they can be transformed» (Salen 2007). Game design is the planning of systems of meaning, and: «Like letters in the alphabet, objects and actions within a game gain meaning through rules that determine how all of the parts relate. A game designer is responsible for designing the rules that gives these objects meaning.» (Salen 2007).

The problem of meaning is far from being clear. From Cognitive Sciences we know that our brain works with memory, but it does not process information like a computer (Rose 2003); and that the use of memory - as well as other functions - physically alters the brain and also transforms the memories. The creation of meaning is individual and hardly sharable. It can be seen as an historical and cultural process articulated by humans in interaction with their natural, social and educational environments. These processes - or problems - call for extra responsibilities in what concerns the creation of games, and even more so for content. Even if just for play, game design cannot be haphazardly prepared, nor be subject to market single party rule, and should respect memories and tradition.

CONCLUSION

Memory and narrative are closely interrelated in recollection, and storytelling. Research in cognitive sciences and DGBLearning proves that people do remember better when content is furnished through a narrative medium - or a game. Trans-literacy is already in order and will soon be the norm.

This urgency oriented these reflexions on the functioning of memory and time in, and out, of traditional narratives. Memory, like history, operates on several levels - exterior or interior to the narrative act. The tools of traditional fiction suffer changes in cyberspace – such as the user's reading time and *prolepsis*. Other forms of cyberworks resort to technology to reinstate memory in their plots. Narratological concepts are no longer able to fully encompass the procedures and practices of hyperfiction. The assessment of narrative bifurcation and suspense is being prepared as future work.

The difficulty of finding a (trans)language to define the new practices is aggravated by the need for a terminology, and for new reading instruments. The main problem becomes one of significance. E-literacy has to go beyond the *litera*, and allow the naming of all the relationships arising from the combination of every imaginable media. Hyperfiction, games, DGBlearning are pervading our world with new systems that have to be decoded, have to be learned, have to teach and be taught. Content development is becoming crucial, and cannot be arbitrarily prepared. The Humanities, with a long term experience in these areas, can unquestionably contribute to help meet these new challenges.

REFERENCES

Aczél, P. 2010. "Mystory in Myspace - Rhetoric of Memory". *New Media in Transforming Culture in the Digital Age*, Tartu 14/16 April. 155-59.

Acquaviva,C. 1970. *Ratio Studiorum-1599*. Trans. Allan P. Farrell, S.J., Detroit University, Washington.

Amerika, M. 1997.*Grammatron*. <http://www.grammatron.com/index2.html> [27.Oct.2010]

Barbas, H. and Correia,N. 2006. "Documenting InStory – Mobile Storytelling in a Cultural Heritage Environment". *First European Workshop on Intelligent Technologies for Cultural Heritage Exploitation*, L. Bordoni, M. Zancanaro, A. Krueger (eds.), ECAI 2006 - 17th European Conference on Artificial Intelligence - Riva del Garda, Italy, 28th. August.

--- . 2009. "The Making of an Interactive Digital Narrative – Instory", Euromedia'2009 - *Fifteenth Annual Scientific Conference on Web Technology, New Media, Communications and Telematics Theory, Methods, Tools and Applications,* (Bruges 15-17th. April), Jeanne Schreurs (eds.), Eurosis-ETI (European Technologic Institute) and Hasselt University, Ghent, Belgium, pp.33-41

Barrett, J. 2010. *Augmented Reality Teaching Blog*. <http://realmixed.blogspot.com/2009/04/ digital-literature-bakhtin-and-dialogic.html> [27.Oct.2010]

Beare, J.I., 1994. Aristotle, *On Memory and Reminiscence* <http://classics.mit.edu/Aristotle/memory.html> [27.Oct.2010]

Benveniste, É. 1974. "L'Homme dans la Langue". *Problèmes de Linguistique Générale, 2*. Gall., Paris, France. 195-238.

Blunt, R. 2005. *A Framework for the Pedagogical Evaluation of Video Game-Based Learning Environments*, Ph D. Diss. Walden University.

--- 2009. "Do Serious Games Work? Results from Three Studies". *e-Learn Magazine*. <http://www.elearnmag.org/subpage.cfm?article=9-1&section=research> [27.Oct.2010]

Bolt, R. A. 1979. *Spatial Data Management*. Cambridge University Press, Cambridge, Mass. 8.

Camillo, G. 1554. *L'Idea del Teatro*.Venice. 6. <http://www.liberliber.it/biblioteca/c/camillo/l_idea_del_teatro/pdf/l_idea_p.pdf> [27.Oct.2010]

Campbel, A. 2006. "Dim O'Gauble". *Dreaming Methods* <http://www.dreamingmethods.com/> [27.Oct.2010]

Caplan, H. Ed. 1954. *Rhetorica Ad Herennium.* Loeb Classical Library, London. <http://penelope.uchicago.edu/Thayer/E/Roman/Texts/Rhetorica_ad_Herennium/home.html> [27.Oct.2010]

Correia, N. and Alves, L. et. al., 2005."InStory: A System for Mobile Information Access, Storytelling and Gaming Activities in Physical Spaces", ACE2005-ACM SIGCHI .U. Politecnica de Valencia, Spain 15 - 17 June. <http://img.di.fct.unl.pt/InStory/publications/paper-ace2005.pdf> [27.Oct.2010]

Crawford, C. 2004. *On Interactive Storytelling*. New Riders, Indianapolis.

Eco, U. 1979. *L'Œuvre Ouverte*. Seuil, Paris.

Edmonds, J. M. Ed.1924. *Lyra Graeca 2.* Harvard University Press, Cambridge, Mass. 307.

Eskelinen, M. 2008. "(Introduction to) Cybertext Narratology". *Cybertext Yearbook Series*. University of Jyväskylä, Finland.

Evelyn-White, H.G., 1914. *The Theogony of Hesiod.* <http://www.sacred-texts.com/cla/hesiod/theogony.htm> [27.Oct.2010]

Fludd, R. 1619. "Ars Memoriae". *Utriusque Cosmi Historia*. Lib. I, t. II, 2nd. part, Cap. X. John Theodore De Bry, Oppenheim 55.

Furman, O. et al. 2007. *Learning & Memory*. Cold Spring Harbor Laboratory Press.

Genette, G. 1966. "Fontières du Récit." *Communications 8*. Ed. Seuil. Paris.152-63.

Halliday, M.A. K. 2002. *An Introduction to Functional Grammar*. Edward Arnold, London (3rd.).

Heidecker, Ch. and Martins, T. 2007. *Noon-a Secret Told by Objects*. <http://tiagomartins.wordpress.com/projects/ noon-a-secret-told-by-objects/> [27.Oct.2010]

Holtzman, S. 1997. *Digital Mosaics: The Aesthetics of Cyberspace*, Simon & Schuster, N. York.

Jauss, H. R. 1982. *Toward an Aesthetic of Reception*. Trans. Timothy Bahti. University of Minnesota Press, Minneapolis.

Kosima, R. 2009. "The Experience of the Unique in Reading Digital Literature". *MIT6 - Media in Transition - Stone and Papyrus Storage and Transmission, International Conference*. (24-26 April). <http://web.mit.edu/comm-forum/mit6/papers/Koskimaa.pdf> [27.Oct.2010]

La Rocca, R. 2007. *Da Memória e Arquitectura* [Art of Memory and Architecture]. Diss. Escola de Engenharia de São Carlos, Universidade de São Paulo. <http://www.teses.usp.br/teses/disponiveis/18/18142/tde-20032008-104738/publico/RenataLaRocca_Dissertacao.pdf>

Laurel, B. 1991. *Computers as Theatre*. Addison-Wesley, Mass.

Loyola, Ignatius, 1914. *Spiritual Exercises - 1521-3*. Trans. Father Elder Mullan, S.J. P.J. Kenedy & Sons, New York Christian Classics Ethereal Library. <http://www.ccel.org/ccel/ignatius/exercises.pdf> [27.Oct.2010]

Manovich, L. 2008. *Info-Aesthetic Manifesto*. <http://www.manovich.net/IA/index.html> [27.Oct.2010]

Masurel, P. and Andrews, J. 2007. *Blue Hyacinth*. <http://www.vispo.com/StirFryTexts/bluehyacinth3.html>

Matussek, P. 1999-2004. "Computers as Theatre of Memory". <http://www.sfb-performativ.de/seiten/b7_engl.html> [27.Oct.2010]

Montfort, N. 2007. *Generating Narrative Variation in Interactive Fiction*. Ph. D. Diss., Faculties of the University of Pennsylvania. 9. <http://nickm.com/if/Generating_Narrative_Variation_in_Interactive_Fiction.pdf> [27.Oct.2010]

Moulthrop, S. 1995. *Hegirascope 1* and *2*. <http://iat.ubalt.edu/moulthrop/hypertexts/hgs/> [27.Oct.2010]

Nelson, T. 1960. *Xanadu.* <http://transliterature.org/> [27.Oct.2010]

Prince, G. 1988. *Dictionary of Narratology*. Scholar Press, Aldershot, U.K. 21.

Ripa, C. 1593. *Iconologia or Moral Emblems*. <http://emblem.libraries.psu.edu/Ripa/Images/ripa00i.htm>

Robert, M. 1972. *Roman des Origines et Origines du Roman*. Gallimard, Paris.

Rose, S. 2003. *The Making of Memory - From Molecules to Mind*. Vintage, London.

Rossi, P. 2000. *Logic and the Art of Memory - The Quest for a Universal Language*. Trans. Stephen Clucas. The University of Chicago Press, Chicago.

Ryan, M. L. 2001. *Narrative as Virtual Reality – Immersion and Interactivity in Literature and New Media*. The John Hopkins U. Press, Baltimore, U.S.A.

Salen, K. 2007. "Gaming Literacies: A Game Design Study in Action". *Jl. of Educational Multimedia and Hypermedia*, 16/3: 301-22.

Sapir, I. et al. 2006. "Narrative, Memory and the Crisis of Mimesis: The Case of Adam Elsheimer and Giordano Bruno". *The Travelling Concept of Narrative.* Helsinki Collegium for advanced Studies, Helsinki. 84-96.

Spencer, J. R. Ed. 1970. *Leon Battista Alberti - On Painting* (1435). Yale University Press, New Haven. <http://www.noteaccess.com/Texts/Alberti/index.htm> [27.Oct.2010]

Thomas, S. et al. 2007. "Transliteracy - Crossing Divides". *First Monday*, 12/12 (3 December). <http://www.uic.edu/htbin/cgiwrap/bin/ojs/index.php/fm/article/view/2060/1908> [27.Oct.2010]

Todorov, T. 1983. *Introduction to Poetics*. Trans. R. Howard. Harvester, Brighton.

Van House, N. and Churchill, E. F. 2008. "Technologies of Memory: Key Issues and Critical Perspectives". *Memory Studies*, 1/3. 295-310.

Yates, F. 2001. *The Art of Memory*. Pimlico, London (8th.).

Yilmaz, A. 2010. "Memorialization as The Art of Memory: A Method to Analyze Memorials". *Metu Journal of the Faculty of Architecture*. Middle East Technical University, Ankara. 267-80.

Zimmerman, E. 2007. *Gaming Literacy - Game design as a model for literacy in the 21st century*.

<http://www.itu.dk/~malmborg/../Zimmerman-Gaming-Literacy.doc> [27.Oct.2010]

VV.AA. 2006. *InStory*. Interactive media Group. <http://img.di.fct.unl.pt/> [27.Oct.2010]

BIOGRAPHY

HELENA BARBAS (1951) is Professor-Lecturer of the Departament of Portuguese Studies – F.C.S.H. – Universidade Nova de Lisboa; she is a researcher of IEMo and collaborator of CENTRIA. She holds a MA (1990) and a PhD (1998) in Comparative Literature – Literature and the Arts, and gained her "Habilitation" (2008) in Literature and Cyberarts.

Research interests: Digital humanities, storytelling, hyperficton, avatars, serious games. She was a member of the *InStory Project* team. She has prepared a project on serious games, *PlatoMundi*, aiming to introduce ethical issues in game playing. <http://www.helenabarbas.net>

AUTHOR LISTING

AUTHOR LISTING